UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

DESARROLLO DE UN SISTEMA DE SOFTWARE PARA LA UTILIZACIÓN DE PIZARRONES ELECTRÓNICOS PARA APOYAR LA ENSEÑANZA DE LA PROGRAMACIÓN ORIENTADA A OBJETOS CON JAVA

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA
(COMPUTACIÓN)

P R E S E N T A :

GERARDO AVILÉS ROSAS

DIRECTORA DE TESIS: DRA. AMPARO LÓPEZ GAONA

México, D.F. 2009.

Agradecimientos

En primer lugar quiero dar gracias a **Dios**, gracias porque nunca me ha dejado solo, porque todos los días me ha regalado el milagro de la vida. Gracias porque me has rodeado de seres maravillosos, ángeles que me han guiado, cuidado y protegido a lo largo de este trayecto. Gracias por darme la fuerza para seguir avanzando en las pruebas que me mandas, porque sé que son oportunidades para ser una mejor persona. Gracias por darme salud, sé que aún esperas más de mí y espero no defraudarte.

Quiero agradecer a la **Universidad Nacional Autónoma de México**, la mejor universidad del país, mi alma mater, gracias por haberme dado la oportunidad de dar un paso más en mi formación profesional y porque fué aquí donde conocí a los mejores maestros y a mis más entrañables amigos.

Gracias al **Posgrado en Ciencia e Ingeniería de la Computación**, porque en sus aulas reafirme mi pasión por la computación y porque me ha dado más armas para seguir adelante en el camino de mi formación profesional, pero sé que me falta mucho por aprender. Gracias a todo el personal de la Coordinación, porque siempre estuvieron dispuestos a apoyarnos en lo que necesitábamos.

Gracias a la **Facultad de Ingeniería** y más recientemente a la **Facultad de Ciencias**, porque en sus aulas he podido explotar otra de mis pasiones: la docencia. Gracias porque con sus estudiantes he aprendido que tengo una gran responsabilidad y compromiso. Gracias por motivarme a prepararme y a ser una persona innovadora.

Quiero agradecer de manera especial a la **Dra. Amparo López Gaona** por invitarme a formar parte de su equipo de trabajo, por creer en mí, por su experiencia y sus valiosas enseñanzas. Gracias por su invaluable apoyo, sin el cual, este trabajo no existiría. Ha sido un enorme placer trabajar con una gran mujer, una gran apasionada de la computación, una gran maestra. Gracias por brindarme su amistad. Gracias también a su hermosa familia porque, aunque pocos, he compartido inolvidables momentos.

A: la **Mtra. Lupita Ibargüengoitia**, la **Dra. Hanna Oktaba**, el **Mtro. Gustavo Márquez** y el **Dr. Fernando Gamboa**; todos miembros de mi Jurado, muchas gracias por sus valiosos comentarios, observaciones y correcciones. Gracias por sus consejos, por compartir su experiencia conmigo y por su enorme disposición para apoyarme a obtener lo mejor de este trabajo.

Al **Consejo Nacional de Ciencia y Tecnología** (CONACyT), gracias por la beca otorgada, porque ésta me permitió continuar con mis estudios de maestría y sin la cual no hubiera terminado en tiempo y forma.

Agradecimientos

Muy especialmente deseo agradecer a mis papás: **Irma** y **José Guadalupe**; quiénes me han enseñado que no me debo de rendir y luchar por cumplir mis sueños. Gracias por todo su apoyo, por sus consejos, por sus palabras de aliento. Este es un sueño más que logro y jamás la hubiera culminado sin ustedes; deben de saber que estoy muy orgulloso de ser su hijo, porque gracias a sus sacrificios he podido salir adelante y gracias a sus enseñanzas he llegado a donde estoy. Este triunfo es también de ustedes. Ojalá la vida y Dios me den la oportunidad de seguir obsequiándoles más satisfacciones y les prometo seguir siendo un hombre de bien, del que se sientan orgullosos. Los amo.

Gracias a mis hermanos: **Lety**, **Alex**, **Víc** y **Gus**; ustedes han sido las personas con quienes crecí, jugue, lloré, hice travesuras y que me conocen perfectamente. Gracias por cada segundo a su lado, sepan que siempre tendrán en mí a un amigo. Los quiero. Gracias a mi hermanita, mi princesa, gracias por soportarme, ten la seguridad de que jamás te dejaré sola.

Gracias a mis abuelitos: **Cony**, **Sol** y **Pedro†**, ustedes serán siempre mi más grande ejemplo; de ustedes he aprendido que la forma de conseguir las metas es a base de trabajo y dedicación. Gracias por su amor y comprensión. **Papá Pedrito**, hoy ya no estas con nosotros, pero sé que desde donde quiera que estes me cuidas y te sigues sintiendo orgulloso de mi. Jamás te olvidaré. A mís tíos: **Ade**, **Gloria**, **Pepe** y **Lupe**; gracias por acompañarme en cada paso que doy, gracias por hacerme sentir como su hijo y por enseñarme que hay que reirnos de todo.

De manera muy especial quiero agradecer a **Javi**, a lo largo de estos seis años, eres la persona que más me ha comprendido y más me ha apoyado, doy gracias a la vida y a Dios haberme dado la oportunidad de compartir momentos importantes contigo. Hoy nos separan unos kilómetros, pero la amistad que forjamos es un alma que habita en nuestros dos cuerpos y un corazón que habita en nuestras dos almas. Gracias por cada regaño, cada consejo, cada palabra de aliento. Eres un eje fundamental para mí y siempre podrás contar conmigo.

Gracias **Anita**(y a los dos **Víctor**), **Javier**, **Mon** y **Blanquis**, mis entrañables amigos de COPADI, gracias por dejarme compartir un pedacito de sus vidas, gracias por su invaluable amistad, su cariño y su confianza. Sé que todo lo que me brindan es desinteresado y sepan que contarán conmigo para siempre. Gracias por cada sonrisa y por enseñarme que la vida es tan difícil como la queramos ver.

A mis amigos: **Cony**, **Mil**, **Patty**, **Betty**, **Lety**, **Veny**, **Cuauh**, **Miguel**, **Victor**, **Lalo**, **Angie** y **Magda**; muchas gracias por cada sonrisa, cada abrazo, cada segundo. Sepan que en mi tendrán un amigo y alguien en quien confiar. Cada enseñanza suya la llevo grabada en el corazón. Los quiero mucho.

A MIS PAPÁS

IRMA Y JOSÉ GUADALUPE, CON TODO MI AMOR

A MIS HERMANOS

LETY, ALEX, VÍC Y GUS

A MIS ABUELITOS

CONY, SOL Y EN ESPECIAL A MI PAPÁ PEDRITO†

A ESA PERSONA ESPECIAL QUE ES CAPAZ DE REVOLUCIONAR AL MUNDO CON SU SONRISA

TODOS USTEDES SON LOS ÁNGELES QUE DIOS PUSO EN MI VIDA PARA PROTEGERME Y GUIARME POR EL CAMINO DEL BIEN. CON ESTO CUMPLO UN SUEÑO MÁS Y QUIERO DEDICARLO A USTEDES, PORQUE USTEDES HAN SIDO MI INSPIRACIÓN Y MI FORTALEZA. DICEN QUE LA GRATITUD ES LA MEMORIA DEL CORAZÓN Y MI CORAZÓN LLEVA GRABADO CON LETRAS DE ORO TODO LO QUE HAN HECHO POR MÍ, DE VERDAD, MUCHAS GRACIAS. LOS AMO Y RECUERDEN:

ALL OF YOU ARE A SONG WRITTEN BY THE HANDS OF GOD...

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

INTRODUCCIÓN

Actualmente, la programación orientada a objetos (POO) es indiscutiblemente un paradigma dominante de programación no sólo en todos los campos de las Ciencias de la Computación, sino también en el sector industrial. La comunidad de software está más o menos de acuerdo en que la enseñanza de POO es una buena idea, incorpora un elegante soporte que facilita la programación estructurada, modularización y diseño de programas. Por otra parte sus técnicas se aproximan a la solución de problemas que han surgido recientemente: programación en equipos de software, mantenimiento de una gran cantidad de sistemas y la reutilización de software. En términos generales se trata de una buena herramienta para enseñar metodologías de programación importantes.

Es importante enseñar programación orientada a objetos (POO) desde los primeros cursos de programación. Enseñar POO es más que enseñar la sintaxis y la semántica de un lenguaje de programación orientado a objetos. Dominar la programación orientada a objetos significa que el alumno debe tener conocimientos fluidos de los conceptos de orientación a objetos y ser capaz de aplicarlos efectiva y sistemáticamente cuando desarrolla programas.

Numerosos grupos de investigación alrededor del mundo dedican sus esfuerzos a proponer nuevos métodos para enseñar POO y gracias a éstos, un igual número de herramientas de apoyo han surgido. Uno de los problemas más comunes que presenta la mayoría de estas herramientas, es que no siempre es posible hacer presentaciones y trabajar con la edición y compilación de programas con la misma aplicación. Por otro lado, muchas de estas herramientas no son fáciles de aprender o utilizar, de manera que son pocos los profesores que se interesan en obtener el máximo provecho de las mismas. Lo anterior supone un gran obstáculo, pues es necesario encontrar un balance entre la enseñanza y la tecnología, de manera que los profesores y alumnos vean a ésta última como un medio para facilitar el proceso de enseñanza – aprendizaje. Por otro lado, es importante que se entienda que la tecnología por sí sola no puede sustituir a ninguno de los

actores en este proceso, por esta razón es que los profesores deben de comprometerse en comprender y entender lo mejor posible el paradigma Orientado a Objetos, de manera que el conocimiento que transmitan a sus alumnos sea más efectivo.

Hoy en día se dispone de varias herramientas que incorporan la animación de algoritmos como una técnica que permite transmitir de una forma más fácil los conceptos de la POO. De esta forma se pretende que los alumnos tengan un acercamiento menos abstracto que el que se presenta comúnmente, en donde, con la ayuda de un editor de texto y el compilador de Java se les muestra la forma en que trabajan los códigos que se programan.

En este trabajo se desarrolla y presenta una propuesta diferente, que incorpora la presentación de información junto con una biblioteca de animaciones; estas animaciones simularán la forma en cómo trabaja un código programado en Java. Para esta versión, sólo se han animado algunos ejemplos representativos, dichos ejemplos cubren los conceptos básicos de la POO: *clases, objetos, agregación* y *herencia*. Esta propuesta no sustituye el trabajo que se realiza con el compilador de Java, ya que los programas se seguirán compilando de la misma forma, sólo que antes de realizar este proceso se puede presentar a los alumnos estas animaciones, con el fin de que comprendan de una mejor forma, la forma en que trabajan los programas que codifican.

Este trabajo permitirá obtener un mejor provecho de herramientas educativas como los pizarrones electrónicos. El sistema incorpora la interactividad suficiente para que el usuario del sistema (el profesor) pueda controlar esta herramienta con los controles y características que tiene el mismo pizarrón. Una de las ventajas de esta propuesta es que el profesor no necesita utilizar otra aplicación para proyectar la información de su clase, ésta se ha incorporado en forma de diapositivas y se ha complementado con las animaciones de cada uno de los ejemplos que se plantean en el sistema; además podrá cargar cada uno de los ejemplos propuestos para mostrar a los alumnos el proceso de compilación y ejecución. Los pizarrones electrónicos incorporan un teclado virtual que permite

hacer pequeñas modificaciones a los códigos propuestos, de manera que podrán ejemplificar errores comunes de sintaxis del propio lenguaje. Aspectos similares se pueden manejar también con una TabletPC.

Para la construcción de este sistema se decidió utilizar el Proceso Unificado, ya que proporciona un marco de trabajo genérico que puede especializarse para cada clase de sistemas de software, para diferentes áreas de aplicación, tipo de organizaciones, niveles de competencia y tamaño de proyecto. Los aspectos que lo distinguen son: manejado por casos de uso, centrado en la arquitectura, además de ser iterativo e incremental. Está basado en componentes, de manera que ésta hecho a base de componentes de software interconectados a través de interfaces; además, usa el Lenguaje de Modelado Unificado (UML) que permite visualizar, especificar, construir y documentar en formatos estándar para toda la comunidad de Ingeniería de Software.

En cuanto a la construcción de la interfaz se aplicaron los Criterios Ergonómicos y se aplicaron pruebas de usabilidad para validar la implementación en un escenario real y se buscó identificar los errores más comunes, para así, asegurar un producto de mayor calidad.

El presente trabajo se ha dividido en los siguientes capítulos:

Capítulo 1. *Antecedentes*. En este capítulo se documentan los principales problemas que ha enfrentado en la enseñanza el paradigma Orientado a Objetos, las metodologías propuestas para lograr un aprendizaje más eficiente de los fundamentos de programación y las ventajas que ofrece el lenguaje de programación Java.

Capítulo 2. *Software para apoyar la enseñanza*. En este capítulo se revisan las herramientas más populares en el mercado que son utilizadas para impartir los fundamentos de programación orientada a objetos y la importancia de la animación de algoritmos y cómo ha influido en el desarrollo de herramientas que incorporan visualización.

Capítulo 3. *La tecnología como apoyo en la enseñanza*. En este capítulo se revisa el estado actual de los dispositivos que se utilizan comúnmente en la enseñanza: los pizarrones electrónicos y las TabletPC's.

Capítulo 4. *Especificación del problema y análisis de requerimientos*. En este capítulo se especifica el problema y los alcances de la solución en el contexto educativo; se describe la fase del proceso unificado de captura y análisis de requerimientos del sistema para describir la funcionalidad completa.

Capítulo 5. *Análisis y diseño del sistema*. En este capítulo se describen las fases de análisis y diseño del Proceso Unificada, aplicadas al sistema de apoyo a la enseñanza de la POO.

Capítulo 6. *Implementación y pruebas del sistema*. En este capítulo se describe la forma en que se construyó la versión final del sistema, especificada y documentada en los capítulos 4 y 5. Se describen las pruebas de usabilidad que se aplicaron para asegurar un producto de mayor calidad y que permiten validarlo en un escenario real de enseñanza (salón de clases).

CAPÍTULO I.

ANTECEDENTES

EN LOS ÚLTIMOS AÑOS, LAS COMPUTADORAS Y LAS TECNOLOGÍAS DE LA INFORMACIÓN HAN EXPERIMENTADO GRANDES AVANCES, UNA DE LAS ÁREAS EN LAS QUE HAN ALCANZADO MAYOR PENETRACIÓN ES LA EDUCACIÓN. HOY EN DÍA INSTITUCIONES EDUCATIVAS ALREDEDOR DEL MUNDO APUESTAN POR LA INCORPORACIÓN DE LA TECNOLOGÍA COMO MEDIO QUE POTENCIALICE LA CALIDAD DE LA EDUCACIÓN.

EL PARADIGMA ORIENTADO A OBJETOS (OO) ES SIN LUGAR A DUDAS, EL PARADIGMA QUE MÁS SE ESTUDIA Y SE ENSEÑA EN NUMEROSAS ESCUELAS Y UNIVERSIDADES; LOS TRABAJOS SOBRE EDUCACIÓN EN ESTE CAMPO SE CENTRAN EN DETERMINAR CUÁL ES LA MEJOR FORMA DE ENSEÑAR A LOS ESTUDIANTES LA PROGRAMACIÓN ORIENTADA A OBJETOS (POO). DE ESTOS TRABAJOS SE DESPRENDEN VARIAS METODOLOGÍAS Y APORTACIONES; SIN EMBARGO, AÚN HAY LIMITACIONES QUE SUPERAR. POR UN LADO, SON POCOS LOS PROFESORES QUE ENTIENDEN Y COMPRENDEN ESTE PARADIGMA, LO CUAL PROVOCA QUE LOS ESTUDIANTES TENGAN DEFICIENCIAS EN SU APRENDIZAJE; AUNADO A ESTO, SON POCO LOS LIBROS QUE EXPLICAN DE FORMA CLARA LOS CONCEPTOS DE LA POO. EN LA ACTUALIDAD, ES POSIBLE ENCONTRAR NUMEROSAS HERRAMIENTAS QUE PUEDEN UTILIZARSE PARA ESTABLECER AMBIENTES MÁS DIDÁCTICOS, SIN EMBARGO, ESTAS HERRAMIENTAS PUEDEN LLEGAR A DIFICULTAR EL TRABAJO DE ENSEÑANZA, YA QUE MUCHAS DE ELLAS REQUIEREN DEL APRENDIZAJE DE HABILIDADES QUE POCO O NADA TIENEN QUE VER CON LA ENSEÑANZA DE LA POO.

OTRO PROBLEMA SIGNIFICA LA ELECCIÓN ADECUADA DEL LENGUAJE OO A UTILIZAR, PUES EN LA ACTUALIDAD, ES POSIBLE ENCONTRAR UNA GRAN VARIEDAD DE LOS MISMOS. EL LENGUAJE JAVA SE HA CONVERTIDO RÁPIDAMENTE EN UNO DE LOS LENGUAJES MÁS UTILIZADOS PARA ENSEÑAR LA POO DEBIDO A SU CLARIDAD EN EL DISEÑO, SU PORTABILIDAD, SEGURIDAD Y ROBUSTEZ; CARACTERÍSTICAS QUE LO POSICIONAN COMO EL VEHÍCULO IDEAL PARA ENSEÑAR LOS FUNDAMENTOS DE LA POO.

I. ANTECEDENTES

1.1 La informática en la sociedad y la Educación

El uso de computadoras y de otras tecnologías de la información y la comunicación permite imaginar nuevos modos de enseñar y de aprender, capaces de conducir a la educación hacia caminos menos difíciles de los que atraviesa en la actualidad; sin embargo, estas tecnologías no son la panacea, es decir, por sí solas no pueden solucionar el problema de la incorporación de las tecnologías de la información en las aulas y la mejora de la educación ya que se trata de algo más que el simple hecho de asegurar la transmisión de conocimientos, la adquisición de competencias y de habilidades diversas.

La educación en sus diferentes modalidades y niveles debe adaptarse a la presencia directa o indirecta de las tecnologías de la información en casi todas las actividades, tanto en los ámbitos públicos como privados. Así, independientemente de la posesión o no de dispositivos informáticos, no deja de crecer el porcentaje de la población que de algún modo y en distinto grado de su vida cotidiana está en contacto directo con alguna computadora.

Actualmente, la programación orientada a objetos (POO) es indiscutiblemente un paradigma dominante de programación no sólo en todos los campos de las Ciencias de la Computación, sino también en el sector industrial. La comunidad de software está más o menos de acuerdo en que la enseñanza de POO es una buena idea, debido a que incorpora un elegante soporte que facilita la programación estructurada, modularización y diseño de programas. Por otra parte sus técnicas se aproximan a la solución de problemas que han surgido recientemente: programación en equipos de software, mantenimiento de una gran cantidad de sistemas y la reutilización de software. En términos generales se trata de una buena herramienta para enseñar metodologías de programación importantes [19].

Es importante enseñar POO desde los primeros cursos de programación. Enseñar POO es más que enseñar la sintaxis y la semántica de un lenguaje de programación orientado a objetos. Dominar la programación orientada a objetos significa que el alumno debe tener conocimientos fluidos de los conceptos de orientación a objetos y ser capaz de aplicarlos efectiva y sistemáticamente cuando desarrolla programas [28].

El proceso evolutivo que dió como resultado este tipo de paradigma comenzó con los lenguajes poco estructurados (BASIC, FORTRAN), siguiendo con aquellos que introducían en forma eficiente los principios de programación estructurada (PL/1, ALGOL, PASCAL). Otros tipos de lenguajes como los modulares, de programación funcional o lógica, que se usan actualmente para resolver problemas específicos, no encontraron una amplia aceptación, como se esperaba en el momento de su aparición [30].

Sin embargo, conforme crecen los requerimientos de los usuarios sobre el software, el tamaño de los programas crece también. El software escrito, usando el paradigma de programación estructurada, consume el 67% de recursos para mantenimiento del total de su costo. Además, el diseño se vuelve un trabajo difícil y la necesidad de trabajar en grupos poco práctica por la misma estructura de los programas [30].

Conocer los conceptos de orientación a objetos no es suficiente. Los estudiantes deben ser capaces de aplicar sus conocimientos para desarrollar programas del mundo real. Los programas de ejemplo que existen en muchos libros son demasiados simples. Los estudiantes raramente encuentran programas de ejemplo en los libros de texto que usen o definan más de tres clases. Pero, en los proyectos del mundo real, los programadores deben usar muchas clases de las bibliotecas y definir muchas clases por sí mismos [28].

En general, la enseñanza de POO se basa en uno de los dos enfoques presentados por McKim: *breadth-first* y *depth-first*. El primero se caracteriza por ofrecer una amplia variedad de tópicos relacionados con la POO, donde los

estudiantes al finalizar el curso son capaces de leer y entender artículos relacionados con la misma, compartir la información con los profesionales avanzados del área y reconocer los conceptos avanzados sobre POO. Sin embargo, como la idea principal es desarrollar en el estudiante una visión amplia sobre la POO, este tipo de cursos se realiza sin depender de ningún lenguaje de programación. La carencia de éste hace imposible fortalecer el curso con la parte práctica [30, 32].

El enfoque *depth-first* se basa en algún lenguaje de programación orientado a objetos y desarrolla en los estudiantes conocimientos sobre POO basándose en este lenguaje y profundizando su conocimiento. Así los estudiantes pueden aprender y también practicar muchos de los conceptos de POO. Sin embargo, la desventaja de este enfoque es que los estudiantes pueden tener una visión sesgada sobre el concepto del mismo basado en un solo lenguaje POO [30, 32].

Según distintas opiniones, el enfoque de *breadth-first* debe ser usado en los cursos más avanzados de la carrera de computación y *depth-first* al inicio de la carrera. En este último se pueden considerar diferentes formas de aprender la POO: *top-down* y *bottom-up*. El primero comienza con la visión global de lo que son los lenguajes de programación: *conceptos de objetos*, *herencia*, *polimorfismo*, etc. y después presenta los detalles de implementación. La segunda forma, *bottom-up*, adopta la pedagogía de *"aprende haciendo"*, donde primero se presenta el problema, después se proporciona la solución al mismo y se hace el análisis adecuado [30, 32].

La enseñanza de POO no es sólo un aspecto de gusto. Es necesario considerar que se trata de una propuesta en cuanto a un paradigma diferente para análisis y diseño. Cualquier cambio de este tipo encuentra resistencias. Así, algunos profesores opinan que el estudiante novato no es capaz de asimilar este concepto por ser muy abstracto; otros dicen que como los estudiantes no saben nada sobre programación pueden asimilar todo lo que se les presenta, solo si esto se hace en una forma adecuada. Lo importante es que el primer curso de programación es de suma importancia, porque es un curso formativo. Si el estudiante desarrolla malas

técnicas de programación será muy difícil corregirlas en los cursos posteriores. Esta responsabilidad está puesta en el profesor que imparte el curso [30, 32].

1.2 Dificultades en el diseño Orientado a Objetos

Aprender un lenguaje de programación no es una tarea fácil. Existen varias razones que apoyan esta afirmación: *entender la orientación a objetos, el aprender la notación, entender la notación, la adquisición de estructuras estándar y finalmente la práctica*. Cuando los profesores comienzan a enseñar a programar, es necesario que sus enseñanzas estén direccionadas a trabajar bajo estas razones, lo cual no siempre es fácil de hacer [30].

Otro aspecto que dificulta lo anterior, es que la mayoría de los libros de texto fundamentalmente han mantenido un vista *procedural* de la programación. Además, la gran mayoría de los textos introductorios tienen una sintaxis que maneja la organización del material de la siguiente forma: el tema central es el *lenguaje de programación* (que sigue una metodología *bottom-up*, donde se comienza con los constructores simples del lenguaje y avanzando con constructores más avanzados). Después de la presentación de los constructores se continúa con las técnicas de programación, el único problema es que en la mayoría de los casos no se explican todas las técnicas en los libros de texto [30].

Es necesario que se proporcione una adecuada organización del conocimiento cuando se está impartiendo un curso introductorio de POO. Las siguientes seis categorías proporcionan una guía para la organización del proceso de aprendizaje y de esta manera poder cumplir con los objetivos educativos [3]:

- ✓ **Conocimiento.** Se refiere a la capacidad de reproducir el material que ha sido aprendido. Sus niveles son: *conocimiento de los hechos, conocimiento de los caminos para manejar esos hechos, conocimiento las teorías y principios generales.*
- ✓ **Comprensión.** Es la habilidad de aplicar lo que se ha aprendido de forma que pueda ser relacionado con otro material o a través de entendimiento.

Sus niveles son: *traducción (explicar en propias palabras), interpretación (resumir lo esencial), extrapolación (implicaciones y consecuencias).*

- ✓ **Aplicación.** Se refiere a la habilidad de usar ideas generales, teorías, principios, procedimientos y métodos en situaciones nuevas o específicas.
- ✓ **Análisis.** La habilidad de descomponer y descubrir relaciones entre partes individuales. Sus niveles son: *análisis de elementos, análisis de relaciones, análisis de principios organizados*
- ✓ **Síntesis.** Se refiere a la habilidad de reensamblar el resultado de cualquier análisis en uno completamente nuevo. Sus niveles son: *crear una nueva estructura, generar un plan o planear operaciones, deducción de relaciones abstractas.*
- ✓ **Evaluación.** La habilidad de evaluar un material dado, sus niveles son: *evaluación por criterios internos y evaluación por criterios externos*.

1.3 Requerimientos para enseñar un lenguaje

Los requerimientos para los lenguajes orientados a objetos en general han sido discutidos en diversos libros y artículos de investigación. Muchos argumentos, algunos a favor y otros en contra han sido presentados para construcciones específicas del lenguaje, por ejemplo, la *herencia.* Lo que es un hecho es que los lenguajes no son buenos o malos *per se*, se puede decir que en general, son buenos o malos dependiendo de un propósito específico. Un lenguaje que es malo en un concepto puede ser excelente en otro (o viceversa). A continuación se presentan una serie de requerimientos para un lenguaje orientado a objetos en el contexto de su utilización para enseñar un lenguaje para principiantes [19]:

- ✓ **Claridad de conceptos.** Los conceptos que se quieren enseñar, deberían ser representados en un lenguaje *claro, consistente y de fácil entendimiento*. Se debería de presentar un modelo de construcción de manera que se puedan implementar variaciones del modelo. La

implementación del lenguaje debe reflejar el nivel de abstracción que se desee usar para los modelos conceptuales.

- ✓ **Orientación a objetos pura.** Se elije esta expresión para hacer la diferencia con respecto a los lenguaje híbridos (lenguajes que soportan y no el paradigma orientados a objetos, por ejemplo C++). El problema de trabajar con este tipo de lenguajes es que los estudiantes que tienen cierta experiencia con lenguajes procedurales, pocas veces se dan cuenta del cambio de estilo y pueden creer que están escribiendo programas bajo el paradigma orientado a objetos sin darse cuenta de que no están considerando sus conceptos principales.
- ✓ **Seguridad.** Se refiere a que los errores pueden ser fácilmente detectados por el compilador. Además los errores deberían ser detectados tempranamente y mensajes claros deberían explicar las causas.
- ✓ **Alto nivel.** El programador no debería necesitar concentrase en cómo trabaja la máquina internamente. Las tareas que pueden ser fácilmente manejadas por el compilador o le sistema en tiempo de ejecución no deberían ser responsabilidad del programador (por ejemplo, el manejo dinámico de la memoria).
- ✓ **Modelo de ejecución y modelo de objeto.** El modelo de ejecución debería ser simple y fácil de entender, esto incluye el modelo de objeto y el modelo de memoria.
- ✓ **Sintaxis legible.** La sintaxis utilizada debería ser legible y consistente, de esta manera, un programa legible es probablemente más correcto. Una sintaxis legible no es garantía de correctez, existen casos donde los errores son descubiertos en programas que están presentes sólo porque un programador no entendió otra parte del código. Un lenguaje legible tiene ventajas para alumnos y profesores. Los profesores muchas veces no están dispuestos a aprender un nuevo lenguaje de programación. Programas que están escritos en un lenguaje bien diseñado pueden ser fácilmente leídos por cualquiera que tenga experiencia en otro lenguaje con similares

conceptos. Otro aspecto de la legibilidad es la consistencia. Para programadores principiantes y avanzados, la legibilidad tiene claras ventajas, pues hace que sea más fácil aprender el lenguaje y reducir el número de errores.

- ✓ **No redundante.** Significa que sólo haga lo que puede hacer. La redundancia es ocasiones es asociada con la flexibilidad y la eficiencia; teniendo diferentes constructores, se puede lograr la misma tarea y en ocasiones se usa para optimizar código. Para programadores principiantes, esto, lejos de ser una ventaja puede causarles problemas, pues al tener distintos mecanismos para hacer la misma tarea, ellos pueden no conocer los efectos de sus decisiones.
- ✓ **Pequeño.** El lenguaje debería ser tan pequeño como sea posible pero incluyendo todas las características importantes. En ocasiones los profesores deciden enseñar sólo un subconjunto del lenguaje, sin embargo no es la solución adecuada, pues los alumnos no conocen toda la potencia del lenguaje que están aprendiendo. Cuando se trabaja con lenguajes pequeños, los alumnos conocen la totalidad de los constructores.
- ✓ **Fácil transición a otros lenguajes**
- ✓ **Soporte para asegurar la correctez**
- ✓ **Ambiente apropiado.** El lenguaje debe tener una buena interfaz gráfica integrada. Una de las dificultades que encontramos cuando se enseña POO a programadores principiantes es la complejidad del ambiente de trabajo. Sería apropiado contar con un ambiente que, primeramente se concentre en el lenguaje de programación, más que en el sistema operativo y que soporte el paradigma orientado a objetos. El entorno de trabajo es uno de los más importantes puntos de esta lista de requerimientos. Un lenguaje apropiado puede no ser utilizado para enseñanza porque no cuenta con un adecuado ambiente de trabajo.

Como se puede observar, hay conceptos tales como la eficiencia, los cuales son en ocasiones considerados extremadamente importantes para la generación de lenguajes de programación, tiene poca importancia para la enseñanza de un lenguaje; esto por ejemplo, sólo se requerirá cuando sea necesario proporcionar un ambiente de enseñanza con un tiempo razonable de respuesta. De manera similar no es importante que el lenguaje sea lo bastante flexible para desarrollar aplicaciones del *mundo real*, pues jamás será utilizado para este propósito [19].

1.4 Requerimientos para un ambiente de enseñanza

Un ambiente adecuado de programación es crucial para el éxito de un curso de programación introductorio. La gran mayoría de los profesores que enseñan programación orientada a objetos tienen problemas con el ambiente usado para la enseñanza [20].

Los ambientes para enseñanza y sus requerimientos no han sido tan discutidos en la literatura de los lenguajes de programación. Esto significa que cuando los académicos discuten sobre un lenguaje de programación particular, sólo se centran en examinar las características del lenguaje y comparativamente en menor grado, en descubrir los beneficios o desventajas de los ambientes de programación. La razón de esto parece ser histórica: cuando surgieron los primeros lenguajes de programación, no existían diferencias substanciales en cuanto a sus ambientes de desarrollo. Por ejemplo, si algún programador usaba una máquina con sistema operativo UNIX, era claro que su ambiente de desarrollo debería ser un shell de UNIX [20].

Sin embargo, han surgido cambios considerables en la última década. La principal razón es que, diferentes ambientes de desarrollo ya están disponibles. Este desarrollo se debe en gran medida a la integración de ambientes gráficos en las computadoras. Hoy en día el mismo lenguaje puede ser usado desde la *línea de comandos* o dentro de un *ambiente gráfico integrado*, creando de esta manera una gran variedad de experiencias en programación.

En segundo lugar tenemos que las crecientes necesidades de los usuarios han provocado la creación de mejores ambientes de desarrollo. En un inicio, los cursos estaban centrados en el desarrollo de algoritmos en lenguajes procedurales o funcionales y para lograr este objetivo, un editor y un compilador eran todo lo que se necesitaba para la parte práctica del trabajo. Los modernos cursos de programación orientada a objetos incluyen *pruebas, corrección de errores y reutilización de código*. Esto ha creado la necesidad de tratar con múltiples archivos fuente y múltiples herramientas de desarrollo. Para proporcionarle al estudiante principiante la oportunidad de trabajar con este incremento en la complejidad es necesario contar con un buen ambiente de desarrollo [20].

Los lenguajes orientados a objetos se han convertido en los más populares, múltiples investigaciones se desarrollan en estos ambientes y nuevos lenguajes orientados a objetos con ambientes de desarrollo integrado han surgido. A continuación se resumen los requerimientos para tener un adecuado ambiente de enseñanza. Dos aspectos cobran particular importancia: *el soporte para orientación a objetos y la conveniencia para la enseñanza* [20]:

- ✓ **Facilidad de uso.** Se trata de uno de los más importantes factores que nos ayudan a decidir la conveniencia de enseñar en un ambiente de desarrollo de software. El ambiente debe ser lo bastante fácil como para que lo puedan manejar estudiantes no experimentados y ser usado para programar tareas después de un corto tiempo. Lo anterior implica que el ambiente de trabajo debe tener una interfaz gráfica de usuario. Las tareas que tiene que ser manejadas por el ambiente de trabajo son muy complejas: *administrar archivos, edición, compilación, administrar las dependencias de compilación, depuración, búsqueda, prueba, ejecución, etc.* Por otra parte, esta interfaz debe de tener una curva de aprendizaje empinada, en especial para usuarios no experimentados.

- ✓ **Herramientas integradas.** Este requerimiento es un resultado directo del requerimiento de facilidad de uso. Las integración de herramientas al ambiente de trabajo tiene muchos beneficios:
  - *Interfaz unificada*. Sí la interfaz de todo los componentes aparece en el mismo lugar, entonces los componentes adicionales son fáciles de entender.
  - *Interfaces pequeñas*. En ocasiones, una herramienta puede hacer uso de una interfaz que es proporcionada por otra herramienta, por ejemplo un depurador. Esto resulta en pequeños componentes de la interfaz para que el usuario aprenda.
  - *Incremento de la productividad*. La integración puede fácilmente proporcionar *atajos* para el ciclo *edición – compilación – ejecución*.
  - *Mejor funcionalidad*. En ocasiones una herramienta puede proporcionar funcionalidad adicional para hacer uso de información que fue generada por otra herramienta.
- ✓ **Soporte para objetos.** Existen muchos ambientes de trabajo, los cuales en un principio fueron desarrollados para ambientes que no soportaban los lenguajes orientados a objetos y posteriormente adaptados para proporcionar el soporte orientado a objetos. Esta adaptación, sin embargo, usualmente falla en la explotación de las posibilidades que trae consigo la orientación a objetos. Aunque existen numerosos ambientes para lenguajes orientados a objetos, sólo unos pocos de ellos son *ambientes orientados a objetos*.
- ✓ **Soporte para la reutilización de código.** La facilidad de reutilización de código es una de las principales motivaciones de la programación orientada a objetos. Los ambientes de desarrollo deben proporcionar un buscador de clases que les permita encontrar una biblioteca de clases. Además debe de tener la habilidad de construir nuevas bibliotecas a partir de las que el estudiante escribió. Esto puede ser reusado posteriormente por él mismo o por otros estudiantes.

- **Soporte para la enseñanza.** El ambiente de trabajo debería proporcionar algunas técnicas que apoyen el aprendizaje de los conceptos de programación: *interacción/Experimentación y visualización*. Con este último se pretende que los estudiantes puedan ver como se materializan las clases, siendo una de las complicaciones más grandes a las que se enfrentan los profesores, pues no es fácil pensar en términos de clases.

✓ **Soporte para el trabajo en equipo.** Esta característica de programación es muy importante, pues en el mundo real es necesario trabajar en equipos.

✓ **Disponibilidad.** Algunos de los requerimientos mencionados anteriormente, son implementados en una herramienta CASE (*Computer Assisted Software Engineering*), sobre todo, existen algunos problemas con algunas de estas herramientas, el costo. Es por esto necesario que el sistema deba tener un precio razonable y que sea capaz de ejecutarse en el hardware disponible para los alumnos.

Ninguno de los requerimientos nombrados arriba es realmente nuevo. Cada uno de ellos ha sido implementado en algunos sistemas existentes.

1.5 El lenguaje de Programación Java

Java se ha llegado a convertir rápidamente en uno de los más populares lenguajes orientados a objetos en el mercado. Cuatro aspectos son los que han apoyado el éxito de Java: *el hecho de que fue el primer lenguaje que produjo applets para ejecutarse dentro de los navegadores web, el soporte que brinda su máquina virtual, el que permita trabajar en una plataforma independiente y su mercadotecnia.* Java es considerado como un lenguaje de programación bien diseñado, pero no es la solución a todos los problemas [19].

Muchos de los conceptos más importantes de la orientación a objetos son presentados de una forma bastante sencilla. Un aspecto positivo es la posibilidad de manejar herencia múltiple: Java soporta tipos separados de herencia a través

de un concepto llamado *interfaz*. Esto permite trabajar con una sola clase de herencia pero múltiples tipos de herencia. La mayoría de los operadores en Java se trabajan a alto nivel, pero hay una omisión que destacar: la generalidad. La falta de generalidad provoca que en algunos casos se emplee un tipo *cast*. Otro problema que se encuentra con Java es la legibilidad en sus sintaxis, además de que no es posible asegurar la correctez [19].

A pesar de las debilidades tratadas anteriormente, Java es el lenguaje más apropiado para mostrar los fundamentos de la programación orientada a objetos, no solo por su relevancia en el mundo de la Computación, sino también por su claridad de diseño. Los diseñadores del lenguaje Java adoptaron una aproximación *minimalista*. Incluyeron únicamente aquellas características indispensables y eliminaron las características que consideraban excesivas o redundantes. Esta aproximación *minimalista* hace que Java sea un lenguaje mucho más sencillo de aprender que otros lenguajes de programación orientada a objetos. Java es un vehículo ideal para la enseñanza de los fundamentos de la programación orientada a objetos [28].

CAPÍTULO II.

SOFTWARE PARA APOYAR LA ENSEÑANZA

ACTUALMENTE MUCHOS TRABAJOS DE INVESTIGACIÓN SOBRE EDUCACIÓN EN COMPUTACIÓN, SE ESTÁN ENFOCANDO AL DESARROLLO Y UTILIZACIÓN DE HERRAMIENTAS DE CÓMPUTO QUE FACILITEN LA ENSEÑANZA DE LOS FUNDAMENTOS DE PROGRAMACIÓN, SIN QUE SU USO DISTRAIGA DEL OBJETIVO FUNDAMENTAL, QUE ES EL APRENDIZAJE DE ALGÚN PARADIGMA DE PROGRAMACIÓN.

INVESTIGACIONES RECIENTES HAN DEMOSTRADO QUE LOS ESTUDIANTES APRENDEN MÁS Y DE MEJOR FORMA, CUANDO LOS PROFESORES EMPLEAN AMBIENTES DE ENSEÑANZA QUE INCORPORAN ELEMENTOS MULTIMEDIA (AUDIO, SONIDO, VIDEO) Y ANIMACIONES, TODO ELLO EN UNA HERRAMIENTA QUE SE INTERACTIVA Y DIDÁCTICA. DESAFORTUNADAMENTE SON POCAS LAS HERRAMIENTAS QUE PUEDEN CONTENER TODOS ESTOS ASPECTOS, PUES LA MAYORÍA DE ELLAS ADOLECEN DE UNA O MÁS DE ESTAS CARACTERÍSTICAS; POR EJEMPLO, UNA HERRAMIENTA QUE INCORPORE MULTIMEDIA, POCAS VECES PERMITIRÁ LA EDICIÓN Y COMPILACIÓN DE UN PROGRAMA, Y SI LO PERMITE, SE TRATA DE LENGUAJES MUY ESPECÍFICOS. LA GRAN MAYORÍA DE LAS HERRAMIENTAS QUE EXISTEN EN EL MERCADO PARA APOYAR LA ENSEÑANZA, DIFICULTAN LA EDICIÓN Y COMPILACIÓN DE UN PROGRAMA, SOBRE TODO, CUANDO SE DESEA REALIZAR ALGUNA DE ESTAS OPERACIONES AL MISMO TIEMPO QUE SE ESTÁ REALIZANDO UNA PRESENTACIÓN. POR OTRO LADO, MUCHAS DE ESTAS HERRAMIENTAS TIENEN UNA CURVA DE APRENDIZAJE QUE DIFICULTA LA RÁPIDA INCORPORACIÓN DE LAS MISMAS AL SALÓN DE CLASES, RAZÓN POR LA CUAL, SON POCOS LOS PROFESORES QUE SE AVENTURAN A UTILIZARLAS Y CUANDO LO HACEN, SON POCOS TAMBIÉN QUIENES LO HACEN COMPLETAMENTE.

LA ANIMACIÓN DE ALGORITMOS, ES UNA TÉCNICA QUE ESTÁ DEMOSTRANDO SER UN MEDIO EFICAZ PARA LA ENSEÑANZA DE LOS FUNDAMENTOS DE PROGRAMACIÓN; EN ESTE SENTIDO DESTACAN TRES HERRAMIENTAS: BLUEJ, JELIOT Y ALICE; CADA UNO DE ELLOS ENFOCADO A UN DOMINIO PARTICULAR, EL CUAL, SIN EMBARGO, AYUDA A LOS ESTUDIANTES A ADQUIRIR VARIAS ESTRATEGIAS PARA SOLUCIONAR PROBLEMAS Y VISUALIZAR SUS SOLUCIONES. LOS CONCEPTOS DEJAN DE SER ABSTRACTOS PARA CONVERTIRSE EN ELEMENTOS CON LOS CUALES DE ALGUNA FORMA ESTÁN MÁS FAMILIARIZADOS, FACILITANDO EL APRENDIZAJE DE ALGÚN LENGUAJE DE PROGRAMACIÓN.

II. SOFTWARE PARA APOYAR LA ENSEÑANZA

2.1 Animación de Algoritmos

Una decisión importante que deben tomar los profesores que imparten cursos de programación introductorios tiene que ver con la selección de una herramienta que soporte la escritura de programas y el desarrollo de software. Usualmente esto involucra seleccionar un entorno integrado de desarrollo (IDE). Esta decisión era bastante simple cuando sólo se requería un editor de textos y una línea de comandos, sin embargo, el panorama cambio cuando se comenzaron a desarrollar las IDE's con lo cual, los lenguajes de programación también cambiaron [44].

La creciente popularidad de la visualización de algoritmos y programas ha llevado al desarrollo de diversas herramientas, muchas de las cuales están diseñadas para tratar una cierta área de la visualización. La ventaja de implementar esas herramientas específicas es que pueden ser altamente especializadas y así ofrecer altas prestaciones. Sin embargo, normalmente estos sistemas resultan inútiles fuera de su campo específico [37].

Una de las áreas que ha mostrado un creciente interés por la visualización es la enseñanza. Un buen número de profesores se ha interesado por utilizar en sus clases, animaciones de algoritmos previamente generadas, en vez de tener que efectuar las operaciones ellos mismos en el pizarrón. Sin embargo, si el sistema usado para la visualización es muy específico, se tiene que usar alternativamente la herramienta seleccionada y la herramienta usual para presentación. Esto resulta bastante incómodo en conferencias, donde los expositores que desean mostrar el uso de un sistema de este tipo tienen que alternar entre la herramienta de visualización y la elegida para la presentación [37].

Disponer de una herramienta que sea lo bastante genérica para mostrar la presentación y manejar el contenido presentado, permite ahorrar el tiempo de intercambio entre dichas herramientas y lo que es más importante, el presentador se ahorra el aprendizaje de distintas herramientas. Sin embargo, el sistema ha de

ser lo bastante genérico como para cubrir los requisitos típicos de las presentaciones y las operaciones propias del campo de aplicación. Es difícil que un solo sistema pueda ofrecer apoyo para visualizar cualquier clase de software. Por consiguiente, es beneficioso disponer de una herramienta que sea extensible dinámicamente por los usuarios para atender sus necesidades específicas sin tener que leer y modificar largos fragmentos de código subyacente [37].

Normalmente, en las herramientas clásicas para elaborar presentaciones se espera que el usuario dé todos los contenidos para elaborar la visualización de forma manual a través de una interfaz gráfica de usuario. La ventaja de este enfoque es que el usuario tiene un control absoluto sobre todos los elementos, al menos dentro de las posibilidades de la herramienta. Sin embargo, al estar obligados a hacerlo todo manualmente, la generación tampoco se puede automatizar. Si hiciera falta producir una segunda visualización sobre el mismo tema pero con valores diferentes, el usuario tendría de partir de cero. Más aún, los paquetes de presentaciones están a menudo mejor preparados para generar excelentes imágenes visuales en presentaciones cortas, lo cual se dificulta cuando se tiene por ejemplo, la necesidad de presentar procesos dinámicos, como algoritmos o estructuras de datos. Estos paquetes, ofrecen una variedad razonable de modos de añadir elementos nuevos, pero pueden estar limitados a la hora de modificar los elementos visibles. Más todavía, el concepto subyacente a la presentación es una secuencia de dispositivas; esto supone que la siguiente diapositiva está inicialmente vacía, sin poder reflejar el contenido de la última, lo cual es inapropiado para visualizar algoritmos que efectúan diversas operaciones a lo largo de una colección de “diapositivas”. En resumen, aunque las herramientas de presentaciones son muy útiles para presentar diapositivas pero no son efectivas cuando se quiere mostrar el comportamiento de sistemas o algoritmos [37].

Los sistemas de visualización basados en el código fuente, como DDD, Jeliot y WinHIPE, presentan automáticamente una visualización de dicho código subyacente. Esto también supone que es muy fácil generar automáticamente

visualizaciones: el usuario sólo tiene que proporcionar el código fuente o cambiar los valores de los parámetros según sus necesidades específicas, sin embargo, estos sistemas tienen algunas limitaciones: el lenguaje de programación que se puede usar es fijo para el sistema. Por ejemplo, DDD [37] trabaja con programas en código de máquina generado normalmente a partir de C o C++, mientras que Jeliot [2, 16, 37] trabaja con código fuente en Java, y WinHIPE [37, 34] está restringido a la programación funcional. Si la herramienta no soporta el lenguaje de programación usado en el contexto de la presentación, no puede usarse. Por otra parte, como la visualización se genera automáticamente, las posibilidades del usuario para cambiar las imágenes son limitadas. Por ejemplo, es poco probable que el sistema permita deducir cual representación de una estructura de datos es la más apropiada semánticamente a partir del código subyacente: por ejemplo, una pila implementada con una lista enlazada podría no ser mostrada como una pila y no como una lista. Por otra parte, los profesores pueden desear añadir texto explicativo, o adoptar un punto de vista más abstracto del algoritmo, como en el pseudocódigo, para evitar confundir a los usuarios con los detalles de implementación. Ninguna de estas operaciones es normalmente soportada por estos tipos de herramientas. Finalmente, si el tema de la presentación no está disponible como un algoritmo, las herramientas no pueden usarse en absoluto [2,37].

Herramientas como JAL [37], de Silicon Graphics, se basan en llamadas a métodos de una biblioteca de programación específica (API) para generar la visualización. La facilidad de uso de la biblioteca, así como el punto hasta el que son soportadas las operaciones, depende del estado de la biblioteca. Se pueden señalar dos exigencias efectivas: el usuario tiene que ser lo bastante experto en programación para escribir un programa que invoca correctamente los métodos de la biblioteca, y el lenguaje en que la biblioteca está implementada debe encajar con el usado para el código subyacente. Si se dan ambos requisitos en una API, puede ser muy útil para generar visualizaciones; de lo contrario, la biblioteca es inadecuada y por tanto inaplicable [37].

Los sistemas que admiten datos de entrada en un formato específico, como un lenguaje de script, ofrecen una solución parcial al problema de adecuación. La herramienta no se ocupa de cómo se genera la entrada de datos (manualmente por el usuario, o automáticamente por el código subyacente). Así pues, el usuario no tiene que ser un experto en programación. Sin embargo, los algoritmos implementados en cualquier lenguaje pueden ser modificados para generar los scripts apropiados. Por esta razón, las herramientas que admiten un script como entrada se han popularizado en los últimos años: JSamba, JAWAA y sistemas como JellRep y JFlap [37], específicos para contenidos de construcción de compiladores. El lenguaje de script subyacente a JSamba [37] es genérico, mientras que muchas otras herramientas ofrecen un lenguaje orientado a un contexto específico. Por ejemplo, JAWAA [37] está orientado a algoritmos y estructuras de datos, y así ofrece elementos como árboles y grafos; sin embargo, es menos útil para presentaciones porque no soporta otras estructuras como listas de elementos. Por otra parte, con frecuencia los sistemas no permiten retroceder uno o más pasos en la visualización [37].

En general, los usuarios deben elegir el tipo de herramienta más apropiada para sus intenciones. Es necesario reflexionar sobre las posibilidades y limitaciones de cada enfoque, especialmente si también desean reutilizar la herramienta en otras aplicaciones menos específicas.

2.2 Estado actual de las herramientas más importantes de visualización

Existen muchos problemas cuando se está enseñando un lenguaje de programación, ya que muchos estudiantes tienen muchas dificultades cuando están desarrollando sus algoritmos, la mayoría de ellos no encuentran una forma sencilla de aplicar las técnicas de solución de problemas que han estudiado en sus programas y peor aún, se les dificulta el utilizar las estructuras de programación comunes. Profesores experimentados en cursos de introducción a ciencias de la computación y cursos introductorios de programación están conscientes que los estudiantes que ingresan a estos cursos, generalmente no cuentan con antecedentes necesarios mínimos; de ahí que sea tema de debate el

decidir el nivel con el que comenzarán a impartir sus clases, es decir, comenzar con el nivel de los estudiantes principiantes provocaría que los estudiantes más experimentados se aburran; el enfoque contrario provocaría que los estudiantes novatos se sientan frustrados y confundidos al sentir que no están comprendiendo ningún concepto [8].

Otro problema reportado en múltiples trabajos tiene que ver con el hecho de que los estudiantes no saben cómo resolver un problema, ellos están acostumbrados a resolver problemas matemáticos, en los cuales, de cierta forma el algoritmo ya esta descrito; no así, cuando ellos tienen que plantear su propio algoritmo. Es por eso importante que se enseñe a los alumnos a desarrollar e incrementar sus niveles de competencia en el diseño de algoritmos, para que puedan resolver los problemas que se les plantean satisfactoriamente; por otro lado es importante que se les enseñe cómo utilizar sentencias de programación específica para que puedan alcanzar sus objetivos. Sí los alumnos no poseen estos antecedentes, será más complicado que sepan cómo resolver un problema y más cuando esta solución tienen que plasmarla en un programa de computadora. Se ha encontrado también que los estudiantes no aprenden satisfactoriamente a escribir, probar y depurar un programa si no saben cómo es que el programa de computadora resuelve el problema; se ha observado que a muchos estudiantes se les dificulta visualizar los pasos que ocurren en la ejecución de un programa y como resultado, es difícil que puedan corregir los errores de sus programas, cuando éstos no trabajan adecuadamente [8].

Es necesario que se les pueda proporcionar a los estudiantes un ambiente en el cual ellos puedan aprender varios tipos de estrategias para solucionar problemas y los conceptos y habilidades necesarias para crear programas de computadora. Animar la ejecución de un programa puede ayudar a los estudiantes a “colocar las piezas del rompecabezas” juntas, la visualización es un enfoque que puede ayudar a los estudiantes a comprender cuál es la tarea que cada “pieza” deberá tener y como todas las “piezas” trabajan juntas para resolver y efectuar una tarea completa y así, resolver el problema [8].

Los lenguajes orientados a objetos se enseñan en las universidades desde hace ya algunos años. Sin embargo, los aspectos pedagógicos relacionados con la incorporación de tales lenguajes aún no están comprendidos propiamente. Los enfoques para incorporar un lenguaje orientado a objetos en un programa de enseñanza varían enormemente. Algunos enfoques dan un mayor énfasis a los aspectos procedurales de un lenguaje orientado a objetos. Otros enfoques se centran en desarrollar una clase poniendo especial atención en la utilización de interfaces gráficas de usuario y applets para enseñar estos conceptos a los estudiantes [39, 21].

Las animaciones han sido utilizadas por algunos autores, para enseñar a estudiantes de ciencias de la computación, algoritmos complicados de forma que la animación puede ayudar a los estudiantes a entender el comportamiento dinámico de un algoritmo, sin embargo no siempre se han tenido resultados exitosos ya que el diseño de las animaciones ha sido más apropiado para usuarios expertos que para principiantes, pues éstos últimos encuentran grandes dificultades para hacer el mapeo de los elementos gráficos de la animación al algoritmo [2].

Los experimentos que se han hecho con aprendizaje a través del uso de multimedia han demostrado en la mayoría de los casos que las visualizaciones deben de estar acompañadas de explicaciones (simultáneas) para que en verdad puedan ser efectivas.; por otro lado, las animaciones presentadas ante un grupo de estudiantes presentan más motivación y satisfacción. Se ha demostrado que los despliegues gráficos no son lo bastante claros si éstos no son estudiados y aprendidos [2].

El aprendizaje basado en multimedia juega dos roles importantes: *guía a los estudiantes y los ayuda a crear las conexiones entre el texto y los conceptos*. El uso de multimedia ayuda a los estudiantes a crear múltiples representaciones del problema [2].

2.2.1 Jeliot

Jeliot (nombre corto para designar *Java* – Eliot) es una familia de sistemas que permite generar animaciones de algoritmos (programas) escritos en el lenguaje de programación Java mediante la visualización de estructuras de datos como objetos gráficos. Ha sido exitosamente utilizado para mejorar la enseñanza en los cursos de introducción a la programación pues proporciona un lenguaje concreto en el cual es posible explicar estructuras de programación y conceptos [28, 33].

El diseño y la operación del marco de trabajo de Jeliot son similares a Eliot, el primer generador de animaciones que trabajaba en un ambiente X Windows y utilizaba animaciones *primitivas* basadas en una biblioteca llamada Polka. Este marco de trabajo está escrito en Java e implementa el modelo Eliot en un ambiente *Wolrd Wide Web*. Jeliot 3 conserva una interfaz gráfica de usuario orientada a usuarios novatos y una ventana de animación, mismos que ya se habían utilizado desde su versión anterior, Jeliot 2000. Ambas versiones visualizan de forma automática la ejecución de programas Java escritos por los usuarios [28, 33].

Tradicionalmente las animaciones para un algoritmo se construyen a partir de la inserción de llamadas a animaciones *primitivas* en varias partes del código del algoritmo, comúnmente denominados *eventos clave*. En Jeliot las animaciones son controladas por la operación de tipos de datos, de manera que el usuario no tiene la necesidad de escribir ninguna llamada adicional. Si el programa utiliza los tipos de datos animados proporcionados por Jeliot, la animación del programa será automática debido a la que la animación se encuentra embebida en la implementación de los tipos de datos [28].

Esta herramienta fue pensada para principiantes, la característica más interesante, es la máquina de animación. Con este mecanismo, los alumnos pueden ver los objetos del programa, la llamada de mensajes, la evaluación de expresiones, entre otras aplicaciones, todas esas actividades son muy útiles para la comprensión de programas. Las animaciones muestran el comportamiento interno del sistema.

Jeliot 3 es una herramienta simple si la comparamos con otras pero las animaciones son difíciles de desarrollar y generalmente, no se utiliza en aplicaciones más complejas porque requiere de aspectos más específicos del dominio del problema de estudio y de la implementación [5, 15].

Jeliot 3 es un sistema extensible y con la capacidad de añadir nuevas características. La interfaz del sistema fue reemplazada con el intérprete *DinamicJava*, el cual fue instrumentado de tal forma que produce un código intermedio llamado MCode, este código intermedio le brinda al usuario la posibilidad de trazar el tiempo de ejecución del programa y es interpretado por la etapa gráfica de Jeliot [33].

Los programas pueden ser creados desde el interior del Jeliot o pueden ser modificados de un programa previamente almacenado. Dicho programa será mostrado visualmente y no necesita ningún tipo de llamada adicional, toda la visualización es automáticamente generada. Esfuerzos adicionales están encaminados a animar características orientadas a objetos, tales como la herencia. Jeliot 3.0 pertenece a una familia de 4 programas de animación que están basados en un paradigma de auto-animación, los cuales son: Eliot, Jeliot I, Jeliot 2000 y Jeliot 3. Las versiones previas de Jeliot permitían la animación de variables, expresiones, I/O y llamadas a métodos estáticos ; Jeliot 3 es además capaz de crear animaciones para los conceptos de la programación orientada a objetos : objetos, herencia de clases, constructores, llamadas a métodos, instanciación y referencia semántica de arreglos y objetos [28,33, 34].

La característica clave de la familia de programas de Jeliot es la visualización semi-automática de los flujos de control y de los datos de un programa Java. El desarrollo de estos programas ha tomado más de diez años en los diferentes estilos de programación. Varias versiones de este software han sido desarrolladas en algunas universidades. La nueva versión de Jeliot es un software que es totalmente gratuito al igual que editor de código BlueJ y el kit de desarrollo de java j2sdk , lo cual lo hace muy fácil de conseguir; y así poder introducirse en los fundamentos de la programación orientada a objetos y del lenguaje Java. Este

programa es de mucha utilidad para que los alumnos comprendan como se ejecuta un programa en Java [28, 34].

Es una herramienta simple que puede ser utilizada en diferentes escenarios de aprendizaje y puede ser utilizada para mostrar y enseñar los conceptos básicos de programación. Los estudiantes podrían usar Jeliot después de sus clases para complementar y entender los conceptos estudiados, aunque también puede ser utilizado como una herramienta para lograr sesiones interactivas en un laboratorio. El despliegue visual del programa puede ser usado para mostrar de una forma sencilla la corrección de errores. Esta herramienta además puede apoyar la impartición de cursos virtuales, pues en los cursos no presenciales en donde Jeliot puede demostrar su potencia como una herramienta que puede ayudar a los estudiantes cuando la ayuda externa no está disponible [33].

El diseño de Jeliot se muestra en la figura 1: un *generador de animaciones* que consiste de una *interfaz de usuario* y un *interpretador visual*. El interpretador transforma los datos de objeto de un algoritmo en sus contrapartes visuales en la animación. El generador de animaciones supone una *semántica visual* para el lenguaje animado. Para facilitar las diversas visualizaciones, la semántica visual no debe ser fija pero si ajustable. La interfaz de usuario (figura 2) permite definir la apariencia visual de la animación (por ejemplo, es posible ajustar los parámetros de la semántica). La presentación de la animación en Jeliot está basada en una *metáfora de teatro*, la cual guía el diseño y la implementación. Es posible considerar la animación completa como una *obra de teatro*. El guión de esta metáfora es el algoritmo a ser visualizado ya que éste determina que es lo que pasará y en qué orden. Por otro lado es necesario contar con una etapa en la que la animación sea mostrada. Es posible que se cuenten con múltiples etapas que muestren lo mismo en el mismo tiempo, pero estas *vistas* serán ligeramente diferentes en cada etapa. La visualización está conformada por *actores*, los cuales son entidades que poseen un conjunto de atributos visuales tales como *tamaño, color y posición*. Cada una de estas entidades tiene un papel, el cual corresponde

a los datos del objeto del algoritmo. La apariencia de cada entidad es determinada por el usuario, es decir, la persona que diseña la animación [28].

Figura 1. Estructura de Jeliot [28]

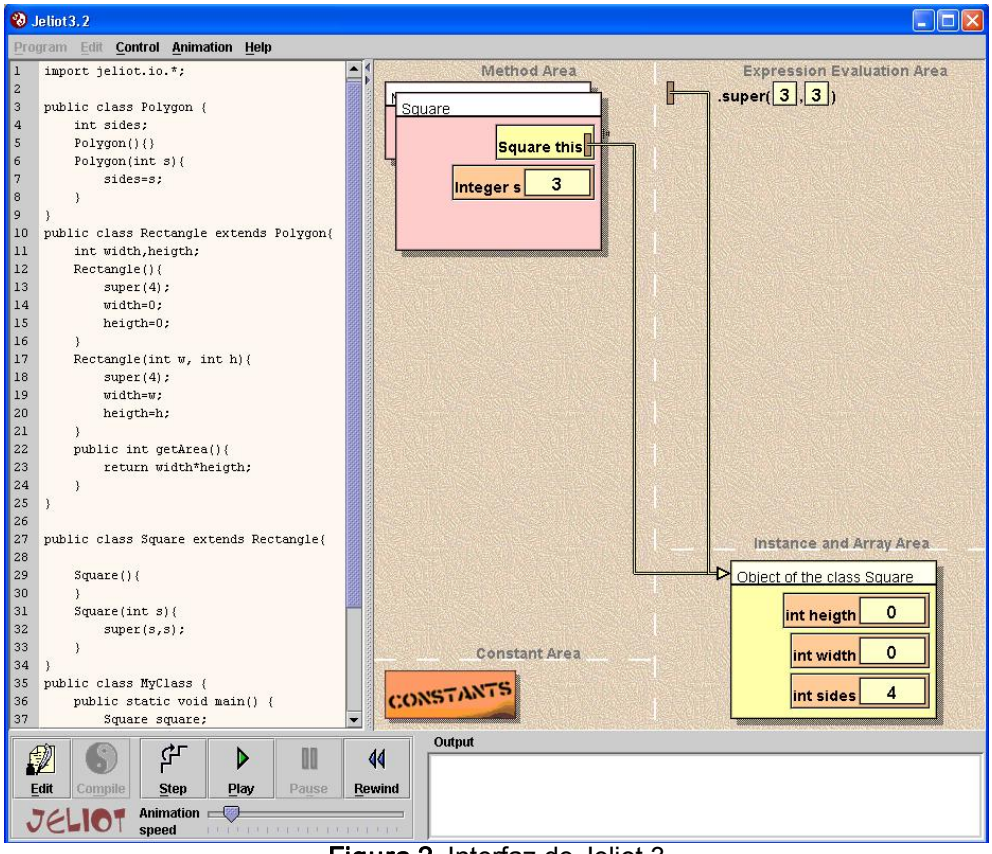


Figura 2. Interfaz de Jeliot 3

Jeliot 3 está disponible para descarga bajo una licencia GPL a través de **http://www.joensuu.fi/jeliot/**. Los desarrolladores, intentan crear una comunidad donde profesores, estudiantes y desarrolladores puedan proponer ideas y soluciones que ayuden a mejorar tanto la herramienta como la enseñanza de la programación para lo cual se dispone de un foro Web. Existen además planes para próximos desarrollos, por ejemplo, una nueva herramienta llamada JeCo (Jelliot Colaborativo), cuyo objetivo es de integrar junto con Jeliot un ambiente de coautoría que permita que los estudiantes puedan desarrollar y visualizar programas escritos en conjunto [33].

2.2.2 Alice

El uso de las animaciones para mostrar cómo se ejecuta un programa no es una idea nueva, a lo largo de varias décadas se han venido desarrollando diversas herramientas que añaden estas ideas, ya sea herramientas que incorporan a la visualización como su filosofía de operación o bien paquetes y bibliotecas de animaciones que en conjunto con algunos lenguajes de programación, permiten la introducción de aspectos visuales dentro de los programas. Sin embargo, a pesar de estos esfuerzos, no siempre es fácil utilizar estas herramientas y menos cuando se les presentan a estudiantes principiantes. Una de las más exitosas herramientas usada para visualizar programas ha sido El Robot Karel J (KJR), un software que se ha utilizado para introducir a los estudiantes a la programación a través de animaciones y utilizado en múltiples cursos desde 1980. Este software inicialmente se utilizó para enseñar Pascal, C y C++; sin embargo, no ha tenido el mismo éxito para la enseñanza de la programación orientada a objetos, pues con ella aumentó también el nivel de complejidad de sus códigos, lo que dificultó la enseñanza de estos conceptos. Entonces es que surge Alice, una herramienta que permite a los estudiantes a construir sus propios mundos virtuales, lo cual la posiciona como una herramienta más realista, además, estos mundos pueden ser visualizados en una página Web [4, 8].

Alice (**http://www.alice.org**) es un ambiente gráfico de programación interactiva, inicialmente desarrollado para ejecutarse en plataformas Windows, sin embargo,

hoy en día es posible encontrar una versión para ejecutarse en plataformas Linux. Este ambiente fue desarrollado en el Carnegie Melon. El objetivo del proyecto Alice es facilitar a los estudiantes novatos el desarrollo de ambientes en tercera dimensión bastante interesantes. [8, 9, 15].

Los objetos populares en Alice se componen de modelos en 3-D (por ejemplo, animales, vehículos, personas, etc.), estos objetos encapsulan sus propios datos (alto, ancho y posición) y tienen sus propios métodos. Escribiendo pequeños scripts, los usuarios de Alice pueden controlar la apariencia y comportamiento de un objeto; durante la ejecución de estos scripts, el objeto responde a las entradas del usuario, las cuales pueden ser hechas a través del mouse o del teclado. Cada acción programada es animada suavemente durante un cierto tiempo especificado en el mismo script (este mecanismo reemplaza la idea en la que el desarrollador prepara una serie de fotogramas para poder visualizar la animación una vez que se reproduce dichos fotogramas a una cierta velocidad). Alice está basado en Python (**www.python.org**) y utiliza muchas características de este lenguaje de programación. Una vez que el mundo virtual es inicializado, el código del programa es creado usando un pequeño editor en el que basta con “arrastrar y soltar” las acciones que queremos añadir a los objetos desde la biblioteca. Este enfoque permite que los estudiantes se centren en los conceptos de objeto y encapsulación y no se preocupen por la puntuación (comas, puntos, paréntesis, etc.); esto permite reducir la complejidad en la escritura de los programas [8, 9].

Alice funciona como un buen lenguaje de programación para los programadores novatos, los estudiantes son capaces inmediatamente de ver cómo sus programas se ejecutan a través de una animación. Por otro lado, la alta capacidad de retroalimentación visual que tiene Alice permite al estudiante saber qué está haciendo el objeto durante animación y saber cuándo se comente un error [8]. El entorno de Alice se muestra en la figura 3:

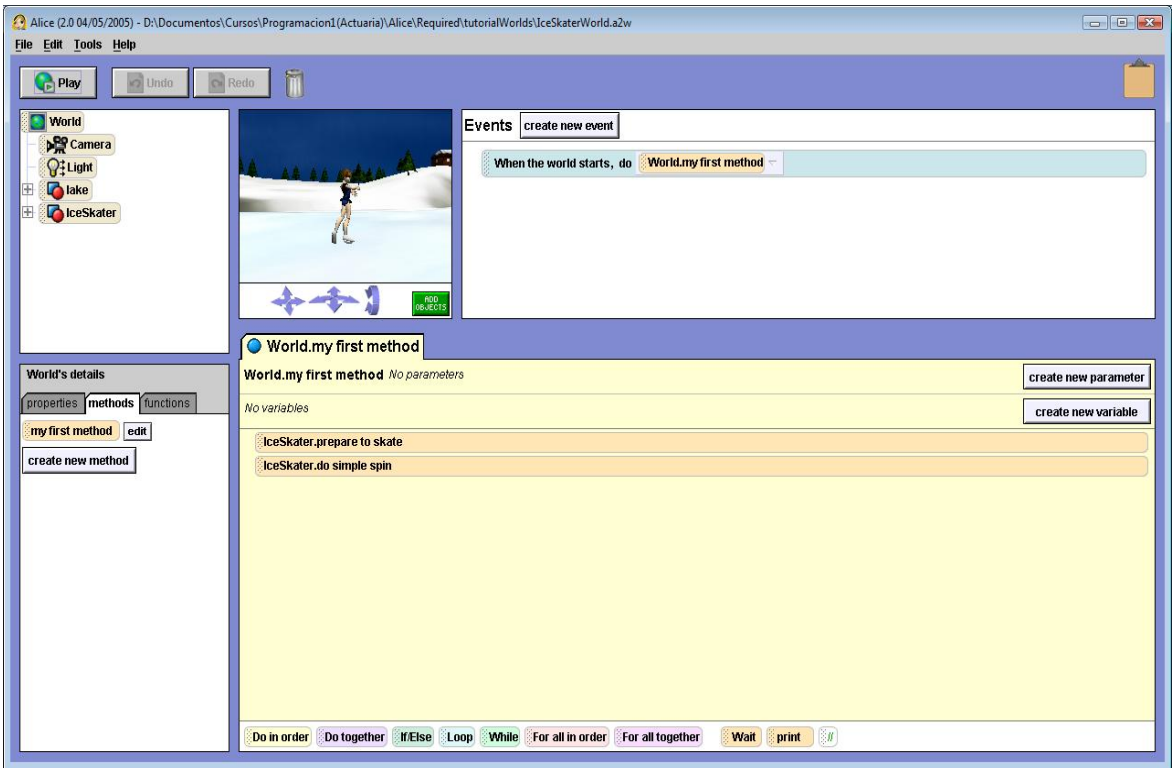


Figura 3. El entorno de Alice

Alice proporciona una gran cantidad de comandos incorporados (acciones); en general, las acciones pueden ser subdivididas en dos categorías: aquellas que le dicen al objeto cómo realizar el movimiento y aquellas que permiten cambiar la naturaleza física del objeto. Alice permite darle nombre de forma más intuitiva a una secuencia de instrucciones, este concepto es similar al de procedimiento, muy popular en otros lenguajes de programación (también se pueden encontrar como funciones). Las funciones y decisiones en Alice están soportadas a través del lenguaje Python; las primeras son generalmente utilizadas para implementar: recursiones, iteraciones y cálculos [8].

Se han realizado numerosas pruebas tomando a Alice como herramienta para la enseñanza de POO y se ha observado que los estudiantes han desarrollado las siguientes habilidades [9, 15]:

- Un fuerte sentido de diseño, se ha observado que después de las primeras clases, los estudiantes tienden a plasmar sus diseños en papel, antes de llevarlos a la computadora.

- Una contextualización para conceptos de la programación orientada a objetos, como objetos, clases y métodos; esto se debe a que todo lo que los estudiantes desarrollan en sus mundos virtuales es un objeto, que pertenece a una clase y que utiliza los métodos para comunicarse con otros objetos.
- Ejemplos claros de prueba y error; los profesores pueden en todo momento hacer cambios sensibles en sus programas y los estudiantes pueden verlos inmediatamente, por otra parte, los estudiantes sienten curiosidad por explorar ellos mismos, que otras "cosas saben hacer" los objetos con los que están trabajando.
- Un enfoque de construcción incremental; los estudiantes no desarrollan el programa completo en un solo paso, sino que van incorporando poco a poco nuevos objetos y probando cada una de sus piezas en todo momento.
- Un firme sentido del trabajo con objetos.
- Una buena intuición sobre el concepto de encapsulación; los estudiantes están conscientes de que a través de ciertos métodos pueden modificar el estado de los objetos, pero saben que hay información de los objetos que no siempre es posible modificar.
- Comprenden el concepto de objeto pues están conscientes de que para lograr que un objeto realice una tarea en particular es a través del envío de mensajes.
- Un fuerte sentido del concepto de herencia; los estudiantes escriben códigos que les permiten crear clases más potentes.
- Comprenden la forma de trabajo del tipo de dato Booleano.
- Desarrollan un sentido más claro de lo que significa el estado de un programa.

Sin embargo, como sucede con muchos lenguajes de programación, no es la panacea y presenta muchos problemas. Mientras el estudiante trabaje con las funciones básicas que trae incorporadas el lenguaje, no encontrará ningún problema, sin embargo, para realizar animaciones más complicadas se enfrentará

con un gran problema: necesitará conocer más a fondo el lenguaje Python, ya que necesitará hacer scripts más específicos. Por otro lado, el tiempo representará también un problema en algunos escenarios, principalmente aquellos que tengan que ver con recursiones. Por otro lado, al crear programas con tan solo "arrastrar y soltar" acciones hace perder a los estudiantes la noción de sintaxis de un lenguaje de programación pues no se les da la oportunidad de experimentar con errores de este tipo [8, 9, 15].

En general se puede decir que Alice proporciona un ambiente para animar programas en tercera dimensión y que hasta cierto punto permite resolver algunos problemas particulares que se presentan en la enseñanza de la programación orientada a objetos. Al utilizar esta herramienta se ha observado que los estudiantes se sienten más a gusto utilizando objetos que le son más cercanos a la vida real y que de cierta manera entiende cómo es que funcionan, esto además, permite que los estudiantes sean capaces de corregir de manera rápida y fácil los errores que se presentan a lo largo del desarrollo de sus mundos virtuales [8, 9].

2.2.3 BlueJ

BlueJ es un entorno de desarrollo relativamente nuevo que incorpora características que pueden ser usadas para crear un curso introductorio de Java que involucre el enfoque de la enseñanza de objetos como primer tema. Es un entorno que presenta una interfaz que ofrece un estilo de interacción que es diferente de otros ambientes de desarrollo disponibles hoy en día. Este entorno ha sido diseñado e implementado por el equipo BlueJ en la Deakin University, Melbourne, Australia, y la University of Kent, en Canterbury, Reino Unido; fue dado a conocer al público en el año de 1998, siendo hoy en día uno de los entornos más ampliamente utilizado en muchas instituciones alrededor del mundo. BlueJ está basado en un lenguaje y un entorno desarrollado preliminarmente llamado Blue, se trata de un entorno escrito completamente en Java. Proporciona un soporte gráfico para el diseño orientado a objetos e integra un entorno que soporta el diseño, edición, compilación y prueba. Adicionalmente BlueJ soporta la creación interactiva de objetos y la llamada interactivas de los métodos de los objetos.

BlueJ fue cuidadosamente diseñado teniendo en mente tres objetivos fundamentales: visualización, interacción y simplicidad. El concepto de visualización tiene el objetivo de hacer que las principales abstracciones de la orientación a objetos estén visibles en la pantalla: objetos y clases [4, 21].

Las clases son visualizadas a través de diagramas similares a los diagramas UML (figura 4), los cuales brindan una visión general de los proyectos y muestran dependencias. La parte interactiva permite a los usuarios interactuar directamente con las entidades conceptuales en el ambiente de desarrollo. Las clases pueden ser instanciadas interactivamente y los métodos públicos pueden ser invocados sobre los objetos. La razón fundamental de esto es permitir que los estudiantes experimenten con las clases y objetos sin la necesidad de escribir ningún código. Esto permite apoyar un enfoque exploratorio que permita comprender la orientación a objetos y que al mismo tiempo involucre a los estudiantes mucho más que otros entornos de desarrollo tradicionales. Estas características proporcionan una forma simple de experimentar y probar la funcionalidad de una clase sin la necesidad de tener un método main [4, 44].

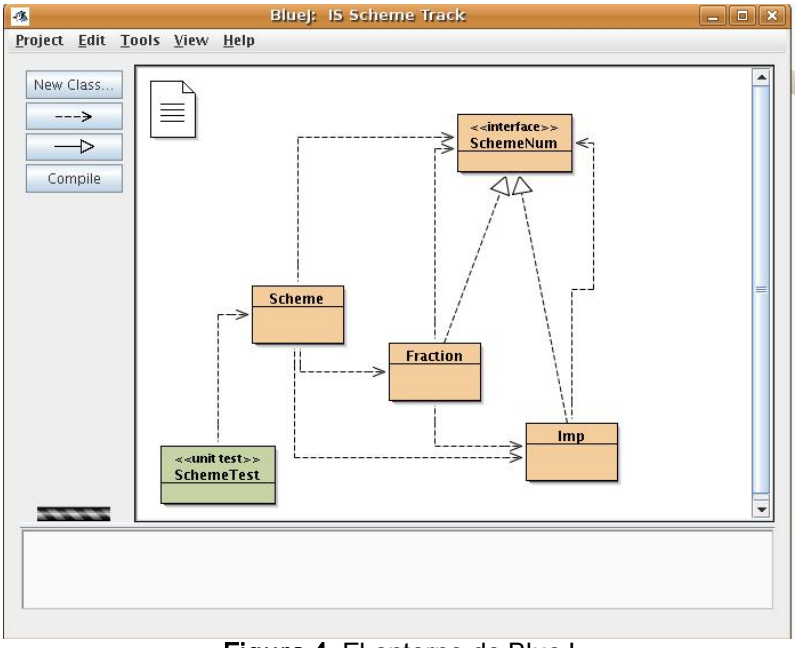


Figura 4. El entorno de BlueJ

La simplicidad es el objetivo clave en el diseño de la interfaz de BlueJ. El entorno de desarrollo por sí mismo no puede llegar a convertirse en un obstáculo en el aprendizaje de los principios de programación. Desafortunadamente, los entornos diseñados por desarrolladores profesionales poseen un grado de complejidad mayor que a la larga distraen del aprendizaje de la programación en sí. BlueJ es una herramienta desarrollada para el uso de estudiantes [4].

Además de los mecanismos generales de interacción con objetos, BlueJ ofrece un soporte integrado para una gran variedad de herramientas educativas relacionadas, la más importante es que integra un depurador de fácil uso, la facilidad de generar documentación a través de la herramienta *javadoc*, un soporte integrado para JUnit y soporta el desarrollo de aplicaciones y applets en Java [4, 21].

Una de las principales diferencias entre BlueJ y otros entornos es que incorpora un enfoque distinto en el contexto de enseñanza ya que combina herramientas poderosas con una interfaz fácil de usar, evitando la complejidad que presentan otras herramientas utilizadas en el salón de clases [21].

CAPÍTULO III.

LA TECNOLOGÍA COMO APOYO EN LA ENSEÑANZA

EL SALÓN DE CLASES SE HA TRANSFORMADO, LOS CLÁSICOS PIZARRONES TRADICIONALES ESTÁN SIENDO ABANDONADOS, PARA DAR PASO A NUEVAS TECNOLOGÍAS: LOS PIZARRONES ELECTRÓNICOS Y LAS TABLETPC. HOY EN DÍA, LOS AVANCES EN LA TECNOLOGÍA HAN PERMITIDO QUE SE ABARATEN ESTE TIPO DE HERRAMIENTAS, DE MANERA QUE SON MÁS RÁPIDAMENTE INCORPORADAS A LAS AULAS.

DIVERSAS INSTITUCIONES EDUCATIVAS ESTÁN APOSTANDO POR ACERCAR LA TECNOLOGÍA A LOS PROFESORES Y ALUMNOS, SIN EMBARGO, A PESAR DE ESTOS ESFUERZOS POR MODERNIZAR LAS AULAS, ÉSTAS NO SE APROVECHAN Y SE EXPLOTAN AL MÁXIMO. LA RAZÓN ES QUE SON POCOS LOS CURSOS DE CAPACITACIÓN QUE SE BRINDA AL PROFESOR Y TAMBIÉN A QUE SON POCOS LOS PROFESORES INTERESADOS EN OBTENER EL MAYOR BENEFICIO DE ESTOS DISPOSITIVOS; COMO PODER INTERACTUAR COMPLETAMENTE CON LO QUE SE ESTÁ PROYECTANDO, CONTROLAR LA COMPUTADORA A TRAVÉS DE LA PANTALLA DE PROYECCIÓN Y FACILITAR LA DISTRIBUCIÓN DEL MATERIAL ELABORADO.

POR OTRO LADO, LOS PIZARRONES ELECTRÓNICOS TIENEN LA DESVENTAJA DE QUE LA SUPERFICIE DE PRESENTACIÓN ESTÁ LIMITADA, ADEMÁS DE QUE AL TRATARSE DE UNA SUPERFICIE PARA PROYECCIÓN, SE DEBE DE TENER CONTROL DE LA ILUMINACIÓN, LO CUAL PUEDE DIFICULTAR LA VISUALIZACIÓN DEL MATERIAL. ADEMÁS, ESTOS DISPOSITIVOS SON COMPLETAMENTE DEPENDIENTES DE LA PLATAFORMA.

UNA SOLUCIÓN VIABLE Y DE BAJO COSTO LA BRINDAN LAS TABLETPC, DISPOSITIVOS QUE PUEDEN CONECTARSE A UN VIDEOPROYECTOR Y TIENEN LA CAPACIDAD DE TRABAJAR CON TINTA DIGITAL, LO CUAL LAS CONVIERTE EN UNA ESPECIE DE PIZARRÓN ELECTRÓNICO PORTÁTIL.

ES IMPORTANTE TENER EN CUENTA, ESTAS HERRAMIENTAS SE DEBEN CONVERTIR SÓLO EN UN MEDIO QUE FACILITE LA IMPARTICIÓN DE UNA CLASE Y NO EN UN DISPOSITIVO QUE SUSTITUYA LA FUNCIÓN DEL PROFESOR, QUE ES LA DE ENSEÑAR.

III. LA TECNOLOGÍA COMO APOYO EN LA ENSEÑANZA

Actualmente el uso de la computadora e Internet en la educación han transformado la relación entre los actores del proceso enseñanza-aprendizaje, se ha estimulado la construcción de nuevos conceptos e interpretaciones del trabajo educativo, siendo necesario tomar en cuenta la forma en que se debe realizar la nueva organización educativa a partir de las condiciones que estas dos tecnologías imponen; por otro lado, influyen fuertemente en la presentación de nuevas propuestas que enriquecen el proceso educativo y el diseño de nuevos productos que giran en torno a ellas, tal es el caso del *pizarrón electrónico* interactivo y la TabletPC [43]. Las tecnologías de información ofrecen un sinfín de recursos que pueden servir de apoyo para lograr un proceso de aprendizaje activo, dinámico e interactivo [36].

3.1 Pizarrones Electrónicos

La influencia de las nuevas tecnologías, como las computadoras, está presente cada vez más en todos los ámbitos de la vida cotidiana y la educación por supuesto no está exenta de esta influencia. Las nuevas tecnologías han venido a revolucionar muchos aspectos fundamentales en materia educativa, ya que *las nuevas tecnologías abren la posibilidad de mayor participación del estudiante en la construcción y desarrollo del proceso de enseñanza – aprendizaje*. Actualmente el uso de la computadora y de Internet en la educación, están reduciendo las distancias, manteniendo la comunicación en línea y la interacción en tiempo real. La convergencia de estas dos tecnologías exige reconocer el impacto y transformación que ocasionan en la educación, de tal forma que se puedan aprovechar para lograr un mayor y mejor proceso educativo, afrontando los retos y problemas que dicha fusión ocasiona; pues es necesario que se planteen soluciones creativas para su uso en la educación. Hoy en día, existe una gran diversidad de propuestas para que las nuevas tecnologías puedan ser utilizadas con fines educativos, sin embargo, para poder implementarlas en proyectos

educativo deben ser minuciosamente seleccionadas, de tal manera que se logre una conjunción perfecta con el aspecto pedagógico de dicho proyecto [43].

Un pizarrón electrónico interactivo (PEI) es una superficie destinada a presentar información mediante un videoproyector conectado a una computadora, que proyecta la imagen de la pantalla sobre una superficie, desde la que se puede controlar la computadora, hacer anotaciones manuscritas sobre cualquier imagen proyectada, así como guardarlas, imprimirlas, enviarlas por correo electrónico y exportarlas a diversos formatos (figura 5). Tiene la característica de interactivo por que brinda a los usuarios la posibilidad de manipular la información a través de una superficie sensible al tacto o una pluma electrónica. También se conoce como *Pizarrón Digital Interactivo* (PDI) [45, 31].

El PEI en apariencia tiene la forma de un pizarrón blanco, de los utilizados para escribir con marcadores, con la diferencia que puede ser conectado a una computadora y puede ser utilizado en combinación con alguna aplicación que permita trabajar en red posibilitando impartir clases a alumnos en la modalidad a distancia, coincidiendo en el tiempo pero no en espacio (sincronía temporal) [43].

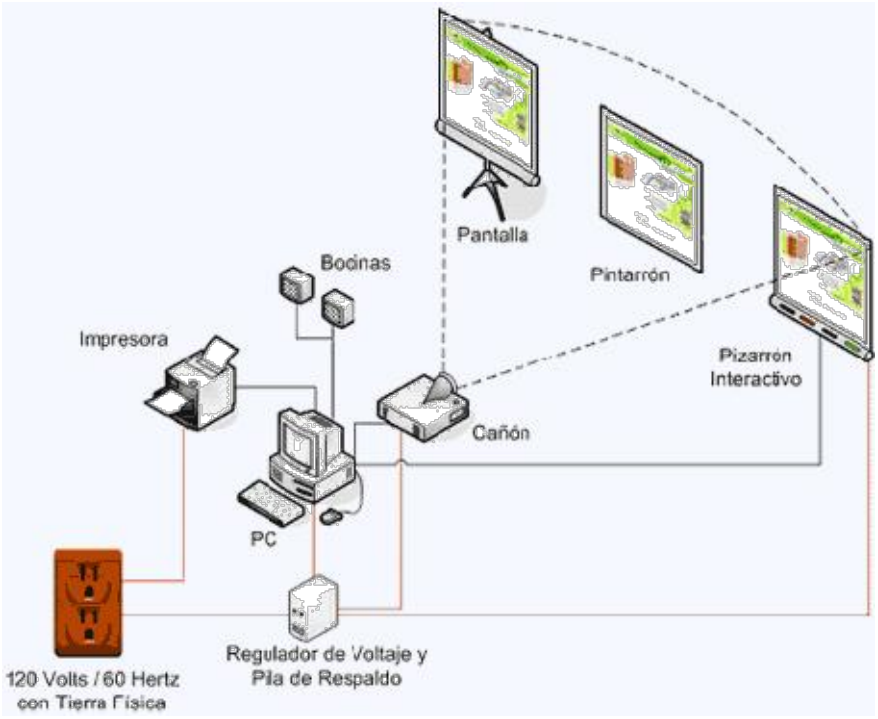


Figura 5. Pizarrón electrónico [28]

El tamaño del PEI permite a los alumnos visualizar la información de la computadora y facilita la interacción con las herramientas propias del recurso; en este sentido se pueden destacar tres funciones principales: sirve como un pizarrón común, puede ser usado como proyector o pantalla y posibilita la interactividad [31].

Según el mecanismo que tenga el PEI para manipular la información, es posible distinguir dos tipos de pizarrones [31]:

- **Hard boards:** tienen una superficie magnética a lo largo de la pantalla y necesitan plumones especiales para actuar sobre ellos.
- **Soft boards:** poseen una membrana "táctil" en toda su superficie que es sensible tanto a las plumas electrónicas como al tacto de los dedos.

Una de las principales características del PEI es el acceso a una amplia variedad de recursos, que corresponden al software del pizarrón y otros más del equipo de cómputo (figura 6). Para optimizar el uso de los recursos, medios y herramientas, se deben considerar dos elementos [31]:

- **Interactividad:** consiste en la posibilidad de responder a los estímulos escritos, visuales o auditivos que muestra la computadora.
- **Multimedia:** es la combinación simultánea de recursos en la computadora, gráficos, sonidos, videos animaciones, etc.

Figura 6. El pizarrón electrónico en el salón de clases

Estos dos elementos pueden contribuir a la construcción del aprendizaje significativo, siempre y cuando el docente maneje adecuadamente los enfoques de enseñanza y realice una buena planeación de la clase. La implementación de los PEI´s en el aula enfrenta a los docentes a nuevos retos: la incorporación de recursos multimedia y para utilizar este tipo de recursos, se tiene que poner en juego las competencias profesionales desarrolladas en la práctica docente, como la elaboración de material didáctico, donde se aprecie el uso de los PEI´s y los tres principales momentos de la planeación: inicio, desarrollo y cierre [31].

El PEI tiene la ventaja que se escribe directamente sobre el propio pizarrón, de la misma forma que se hace sobre cualquier pizarrón convencional, lo que lo hace especialmente sencillo de utilizar por un profesor desde el primer minuto. Otra característica, es que puede ser sensible al tacto, la cual permite controlar la exposición directamente desde la pantalla como si se estuviera utilizando el ratón o el teclado; dicha característica puede utilizarse para acceder, desplegar información y ejecutar programas de aplicación contenidos en la computadora y trasmitirlos a través de una red local [43, 45].

Es importante mencionar que el uso del PEI en los diferentes momentos de la clase puede propiciar el exceso de estímulos visuales y auditivos, sin dejar espacio al análisis y reflexión de los contenidos o temas trabajados con los alumnos. Para ello es recomendable el PEI sea utilizado a partir de las necesidades del grupo con que se trabaja, lo cual significa que se deben buscar los espacios y los recursos mejor adaptados a los propósitos y contenidos abordados en clase. Dependiendo de la intención que el docente le dé, el PEI puede convertirse en una herramienta que [31]:

- Promueva el trabajo en grupo.
- Fortalezca los espacios de discusión y análisis de la información.
- Coadyuve al diseño materiales didácticos.
- Favorezca el trabajo colaborativo.
- Permita compartir ideas y guardarlas en archivos para después usarlas.

La incorporación de los PEI´s de manera integral a las actividades del aula permite que, a partir de la observación de algunos recursos, los alumnos platiquen o comenten en grupo lo que piensan, opinan o saben en relación con lo visto en clase. Ningún PEI o programa computacional puede cambiar las prácticas tradicionales, ni generar por si solo situaciones de aprendizaje significativas y pertinentes para los alumnos. El docente es punto clave para el éxito de la incorporación de las herramientas de cualquier tipo al aula.

Se debe tener claro que no basta con proyectar una imagen, una presentación, un texto o un video, si no se genera el análisis y reflexión del contenido por los alumnos. Los docentes ubicados la línea de profesionalización de su tarea pueden descubrir en el PEI una interesante herramienta para la generación de una enseñanza participativa y responsable para sus alumnos.

Las principales características que se pueden enlistar del PEI son [43]:

- ✓ Captura instantáneamente lo que se escriba en el pizarrón y lo muestra en todas las computadoras que se encuentren conectadas en la sesión.
- ✓ Permite compartir las imágenes del pizarrón, en tiempo real, sobre una sesión permitiendo el control a las computadoras remotas.
- ✓ Crear respaldos permanentes de las anotaciones que se hagan en clase sobre él, las cuales además de la posibilidad de imprimirse pueden enviarse por correo electrónico o publicarse como páginas HTML.

Las características mencionadas del PEI, las características de las computadoras y la combinación con Internet, ofrecen al sistema educativo la posibilidad de crear un ambiente de aprendizaje interactivo a distancia, en donde se puede ejecutar una gran variedad de programas, desde los más sencillos hasta aplicaciones multimedia, que representan sin duda alguna un enriquecimiento en el proceso enseñanza – aprendizaje [43].

Dentro de este contexto de la educación a distancia, surge un nuevo concepto al que se le denomina aula virtual, en la cual, a diferencia del aula tradicional, ya no

es necesario la presencia físicamente de los profesores y alumnos; el encuentro ahora es a través de Internet, y puede ser en forma síncrona (coincidencia en el tiempo) o asíncrona (no coincidencia en tiempo ni espacio) [43].

Dentro del concepto de aula virtual, el PEI puede ser un elemento enriquecedor si tomamos en cuenta las siguientes ventajas [43]:

- ✓ Crea un ambiente de aprendizaje interactivo al permitir el control a todos los participantes de la sesión.
- ✓ Maximiza la efectividad de la sesión a distancia ya que se envía la aplicación y no la imagen del expositor.
- ✓ Mejora la impartición de la lección proporcionando las herramientas para realzar temas clave y escribir anotaciones sobre imágenes de computadora.
- ✓ Crea un registro de la clase.

Sin embargo como sucede con otras tecnologías, también se tienen limitantes que resultarían de poca importancia si se utiliza todo el potencial que ofrece para la educación. Dentro de las limitantes que se tienen para la utilización del PEI se pueden mencionar las siguientes [43].

- ✓ Equipo de precios no muy accesibles. (Desde $499 dólares).
- ✓ Se tiene que combinar con otras aplicaciones tales como NetMeeting.
- ✓ Sólo se puede intercambiar audio y vídeo con el primer participante si se utiliza NetMeeting. Se necesita una MCU (Multipoint Control Unit) para conmutar entre varios participantes.
- ✓ El número máximo de participantes que admite depende de la topología de la red y del sistema operativo implicado.

La conexión del pizarrón electrónico a la computadora se realiza de la siguiente manera [43]:

- Se instala la aplicación del pizarrón electrónico en la computadora designada para su utilización.

- Se conecta el cable de comunicación al puerto COM de la computadora.
- Posteriormente, se conecta el proyector multimedia a la computadora y se calibra el pizarrón electrónico para que sea sensible al tacto y manejar la computadora por medio de él.
- Por último, se ejecuta la aplicación del pizarrón electrónico, instalada previamente, al hacerlo se escucha un sonido que indica que la comunicación entre la computadora y el pizarrón electrónico se ha iniciado correctamente.

Además, La búsqueda de la efectividad de la tecnología en la educación no es una tarea fácil, es importante tomar en cuenta [36]:

- ✓ Qué tipo de aprendizaje se pretende desarrollar en los estudiantes
- ✓ Qué estrategias educativas se emplearán en conjunto con la tecnología
- ✓ Preguntarse si realmente la tecnología de información satisface las necesidades de enseñanza y aprendizaje de nuestro salón de clases.

Por otra parte, es importante preguntarnos en qué medida, los planes de estudio que se imparten responden a las necesidades, demandas y características de la sociedad tecnológica en nuestro país. Una vez que se ha tomado en cuenta lo anterior, es posible entender y estar conscientes de que la tecnología es el medio y no el fin; es una herramienta que puede ser de gran ayuda en la enseñanza siempre y cuando se aplique de una forma adecuada. Por lo tanto, se considera que dentro del diseño curricular de los cursos, los docentes deben buscar la efectividad de la aplicación de tecnología en el proceso de enseñanza cuidando no caer en el abuso y sobresaturación del uso de esta herramienta [36].

Además, es de suma importancia la capacitación para los profesores en el uso adecuado de la tecnología, esto representa en definitiva un factor clave para obtener éxito en el proceso de aprendizaje de los alumnos. Un aspecto importante a tomar en cuenta como eje principal, es instruir al profesorado en el uso de las tecnologías de la información, ya que la inversión que se hace en tecnología no

será redituable mientras los docentes carezcan de las habilidades necesarias para utilizar efectivamente estos recursos tecnológicos [36].

Una vez que el docente cuente con las habilidades necesarias para el buen manejo de la tecnología se deben enfocar esfuerzos en el desarrollo de actividades que realmente dejen un aprendizaje significativo en el alumno, ya que esta es la tarea fundamental del docente y no de la tecnología. Aunado a los dos aspectos anteriores, también es importante que los educadores estén atentos y con juicio crítico para discernir entre el uso o abuso de las tecnologías de información como herramientas para el aprendizaje. Se debe destacar la importancia de tener bien definido cuándo es apropiado utilizar la tecnología como parte de las estrategias educativas y cuándo no lo es. No se debe dejar llevar completamente por la oleada de las tecnologías de información y la sociedad del conocimiento. Como formadores se debe tener como tarea fundamental el compromiso revisar contenidos y actividades, así como el perfil de los estudiantes para determinar si realmente la tecnología será un apoyo o una barrera dentro de la instrucción. Con el PEI se pretenderá entonces apoyar al proceso de enseñanza de forma individualizada, con un enfoque al trabajo colaborativo de los estudiantes, buscando promover la interacción entre los diferentes recursos y actividades de manera tal que el alumno sea el propio constructor de su conocimiento [36].

3.2 TabletPC

Cuando se piensa en un salón de clases, inmediatamente viene a la mente un lugar en donde se encuentra instalado un pizarrón convencional. El pizarrón brinda al profesor la capacidad de plasmar sus ideas o material al instante, sin embargo esta forma de trabajar tiene dos defectos: todo el material debe ser puesto sobre el pizarrón durante la clase y una vez que el espacio del pizarrón está lleno, parte de este material debe ser borrado para de esta manera, tener espacio para colocar más material; bajo este esquema, los estudiantes deben copiar el material del pizarrón sí es que desean tener un conjunto de notas de clase [40].

Un proyector puede ser utilizado para reemplazar o complementar el material que se coloca en el pizarrón, en este caso se utilizan hojas preparadas antes o durante la clase; este material puede ser almacenado para futuras referencias o ser reutilizadas, pero en general es difícil distribuir este material a los estudiantes a menos que se cuente con alguna forma de fotocopiar dicho material [40].

El uso de una Tablet PC como ayuda en la preparación e impartición de clases y cursos ofrece lo mejor de estos dos mundos: la facilidad de preparar clases, conferencias y cursos con gráficos bien planeados, y la habilidad de hacer anotaciones y desarrollar una clase en tiempo real como podría hacerse con un pizarrón convencional [40].

Un pizarrón blanco podría ser utilizado de manera similar un pizarrón convencional, también se necesita una computadora que tenga almacenada toda la información que se va a mostrar, lo cual facilita el uso posterior, sin embargo, el desarrollo de notas futuras se dificulta, ya que toda la información se encuentra desarrollado con algún software en particular y para añadir nuevo materia, se tiene que utilizar el software que lo generó. Ahora bien, las clases se pueden desarrollar utilizando software específico o directamente en una página Web con presentaciones espectaculares preparadas con anticipación y proyectadas durante la clase, de esta forma se facilita el intercambio del material, pues las copias pueden ser fácilmente distribuidas por medios electrónicos o bien descargadas desde un sitio Web que permita el almacenamiento de archivos. El material desarrollado bajo esta plataforma puede ser tan complejo como se desee, se pueden añadir un sin número de recursos audiovisuales multimedia; sin embargo, la complejidad de dichas presentaciones está limitada por el tiempo y el talento que posea el profesor, esto se debe a que no es fácil desarrollar una presentación detallada durante la clase, el material debe ser desarrollado antes de ser presentado [40].

Las presentaciones que se hacen para impartir un curso, basadas en computadora, tiende a ser comúnmente menos flexibles e interactivas. Las clases generalmente se preparan usando alguna herramienta para editar diapositivas; sin

embargo, en este tipo de entornos, la habilidad para cambiar la dirección, poner anotaciones o ilustrar en tiempo real se vuelve una tarea difícil. En el caso de las matemáticas, las ciencias y las ingenierías, la habilidad de ilustrar ciertos puntos es absolutamente crítica. Generalmente, cuando se utiliza algún recurso para hacer más "interactiva" la clase, se utiliza una computadora y un videoproyector, al utilizar estas herramientas el salón de clases debe ser oscurecido para poder permitir las capacidades de proyección, sí en algún momento es necesario ilustrar algún concepto con un ejemplo, entonces se vuelve problemático usar el pizarrón convencional sin interrumpir completamente el flujo de la presentación [40].

Desafortunadamente esta forma de trabajo impide que la clase se desarrolle de manera interactiva, pues mientras que durante la clase se pueden hacer comentarios, no se pueden hacer anotaciones tan fácilmente sobre el material, y si el material presentado necesita alguna explicación adicional o se tienen preguntas adicionales, esto se tiene que hacer sobre el pizarrón convencional, o bien, en un pizarrón electrónico, sin embargo, en este caso es necesario utilizar alguna otra aplicación de computadora, de esta manera dos o más tecnologías que no tienen nada que ver con la clase particular, tienen participación y de esta forma, el resultado es menos óptimo [40].

Planteando una situación típica se tiene por ejemplo a un profesor que proyecta una presentación usando una laptop con algún software que presenta diapositivas para dar una clase, incluso con la mejor de las preparaciones, las preguntas de parte de los estudiantes pueden surgir. Las respuestas podrían involucrar fórmulas, operaciones, diagramas, gráficas, etc., lo cual significa que el profesor necesitará utilizar algo más que la computadora, usualmente el pizarrón es la única alternativa a la mano. En muchos salones de clases o de conferencias, la luz en el frente del salón es atenuada para permitir que las proyecciones sean claramente vistas, de manera que el pizarrón no puede ser usado si cambiar la intensidad de la luz. Por otra parte, la pantalla donde se proyecta podría estar bloqueando una buena porción del pizarrón. En salones de clase pequeños (40-50 estudiantes), solo el 20% del pizarrón está disponible; en el caso contrario,

se podría presentar el caso de que la pantalla sea muy pequeña y sólo una pequeña cantidad de información puede ser mostrada; bajo estas condiciones, en muchas ocasiones es será necesario apagar el videoproyector para poder ocupar el pizarrón completo, con lo cual habrá pérdidas de tiempo ya que habrá que borrar el pizarrón o la información será cubierta con la pantalla de proyección [40].

Otro gran problema es que sólo es posible tener disponible la información de la presentación que se haya preparado en la computadora, sin embargo, el material que el profesor desarrolle en el pizarrón no estará disponible para futuras referencias. Existen la posibilidad de utilizar cámaras de video, pero esto es realmente impráctico. Los pizarrones electrónicos interactivos podrían ser utilizados, pero las oportunidades de que cada salón de clases este equipado con uno, son escasas, por otro lado, son poco portables, de manera que esto dificulta que sean llevados de un salón a otro [40].

Al utilizar una TabletPC con conectividad inalámbrica y un videoproyector, un profesor puede simplemente cambiar las aplicaciones y utilizar la computadora como un bloc de notas digital y proyectar directamente sus anotaciones a la clase sin la necesidad de utilizar el pizarrón convencional. De esta manera, el profesor puede moverse libremente en todo el salón de clases si así lo desea y como elemento adicional, todas las ilustraciones y anotaciones hechas en las presentaciones pueden ser publicadas directamente en algún medio electrónico y pueden ser distribuidas por medio de Internet [40].

Una TabletPC es una computadora portátil tradicional con la habilidad de procesar tinta digital y que en la actualidad, se ha convertido rápidamente en una poderosa herramienta de **eLearning** en educación superior (figura 7). Su alto uso educativo permite a los usuarios crear material utilizando la tinta digital, dicho material puede ser distribuido a los estudiantes para revisiones posteriores. La tinta digital permite que las clases se vuelvan más dinámicas e interactivas [16,37]

Las TabletPCs tienen un gran potencial para la enseñanza en los salones de clases, pues sus usos han sido explorados en numerosas publicaciones y han sido

usadas, principalmente como apoyo en la enseñanza de materias de educación primaria y secundaria; sin embargo, el desarrollo de nuevos sistemas que apoyen la enseñanza han sido desarrollados principalmente en los departamentos de computación de las universidades [17].

Figura 7. Tablet PC estándar

Muchos profesores han comenzado a tomar nota de las ventajas de utilizar TabletPcs en conjunto con proyecciones sobre pizarrones blancos, utilizando simplemente transparencias o bien presentaciones hechas con algún software. Una imagen proyectada es más fácil de ver sobre un pizarrón blanco que sobre un pizarrón tradicional (verde), especialmente en grandes salones de clases. Comparados con los proyectores de acetatos, las presentaciones hechas usando TabletPCs son más fáciles de corregir, guardar y enviar (vía correo electrónico), sin embargo, las presentaciones de diapositivas carecen aún de los elementos que le permitan al profesor interactuar con su presentación es decir, aún se trata de presentaciones estáticas. Los profesores han encontrado una solución a este problema y utilizan las herramientas para hacer anotaciones que brindan las

interfaces de los pizarrones electrónicos y de esta manera poder interactuar con sus presentaciones [17].

Por otro lado, estudiosos en el área, afirman que en un futuro próximo, la gran mayoría de los salones de clases estarán equipados con computadoras conectadas en red a través de las cuales los profesores y los alumnos tendrán más interacción sobre un ambiente de "presentaciones" y de esta manera facilitar una enseñanza y retroalimentación en dos direcciones, con lo cual las características pedagógicas se verán incrementadas [17].

Las TabletPCs representan el más reciente intento de la industria de la computación de colocar en el mercado una computadora basada en un sistema de tinta digital. Fueron introducidas como un prototipo de Microsoft en 2001 y fueron presentadas como la siguiente generación, evolución de las tradicionales laptop. En el año 2002 se liberó el prototipo y se presentó al público, al principio Microsoft se centró en tener una TabletPC que fuera usada como una computadora con características de escritura. Las TabletPCs poco a poco han ido ganando terreno y un buen número de fabricantes (Compaq, Gateway, Mobile Computing, etc.) ofrecen actualmente alrededor de 15 diferentes equipos. Las TabletPC se encuentran en varios ambientes: viajes, reuniones de negocios y salones de clases pues rápidamente se ha convertido en una herramienta útil para la enseñanza. Las TabletPc se encuentran en la actualidad en su tercera generación y están dotadas de suficiente poder de cómputo, el cual les permite tener un rendimiento similar al de una computadora de escritorio relativamente poderosa [17, 12].

3.2.1 Las TabletPCs en la educación

Una TabletPC es una computadora portable con características que la llevan más allá de una tradicional laptop. Su principal ventaja es que utilizan un estilo diferente para la entrada de información. Adicionalmente, existen varios productos de software hechos específicamente para las funcionalidades de una TabletPC [12].

La TabletPC ha encontrado y creado un nicho importante dentro del sistema educativo, como un mecanismo útil de colaboración y su utilidad en el campo de la educación ha sido documentada, tanto dentro como fuera del aula; pueden ser usadas sólo por los profesores, sólo por los alumnos o bien por ambos. Sin embargo, diferentes instituciones educativas han utilizado las TabletPCs de diferentes formas [12].

El material de las clases se prepara tradicionalmente en papel, el cual puede consistir de apuntes o diapositivas impresas; este material se distribuye entre los estudiantes, sin embargo la cantidad de tiempo que se requiere para preparar este material es mucho, lo mismo que la cantidad de recursos que se utilizan. A través de una TabletPC el profesor puede de una forma sencilla desarrollar su material y distribuirlo de forma electrónica con sus alumnos, por otra parte, su capacidad para manejar la tinta electrónica le brinda la facilidad de poder retroalimentar a sus alumnos de forma electrónica, sin que haya un gasto excesivo de recursos. Como una herramienta dentro del salón de clases, las TabletPCs pueden proyectar información en una pantalla a través de una conexión alambica o inalámbrica. El profesor puede hacer anotaciones, ampliar las explicaciones sobre un tema en particular y guardar esas anotaciones para distribuírselas a sus alumnos posteriormente [12].

Las TabletPC son usadas típicamente en el salón de clases como dispositivos que permiten hacer presentaciones, lo cual les permite fácilmente tomar el lugar de los pizarrones tradicionales, la salida de video se muestra en una pantalla utilizando un videoproyector, como se muestra en la figura 8 [16, 37]:

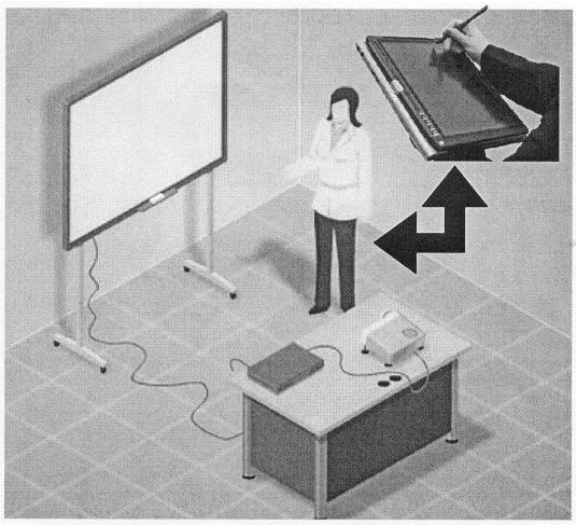


Figura 8. La TabletPC en el salón de clases

Las TabletPCs permiten a los miembros del personal docente la creación de material digital para sus clases de forma rápida mediante el uso de las características de tinta digital y una gran variedad de paquetes de software. Un número importante de universidades de todo el mundo han adoptado proyectos piloto que les permiten estudiar el impacto y la eficacia de las TabletPC como apoyo para la enseñanza. Por ejemplo las universidades de *Notre Dame* y *Seton Hall* han implantado programas a través de los cuales ponen en manos de su personal docente TabletPCs y de esta manera examinan el impacto que tienen como herramientas de enseñanza. El *Colegio Bentley* ha implantado un programa en el cual distribuye TabletPCs a sus estudiantes. Estudiantes del *MIT* han usado las TabletPCs para diseño creativo y colaborativo de robots. La *Universidad de Washingtong* ha desarrollado un proyecto denominado "Classroom Presenter" basado en TabletPCs; el sistema permite que el profesor imparta su clase desde una TabletPC que se comunica de forma inalámbrica con un video-proyector, esto permite que el profesor camine libremente por el salón de clases, por otro lado, los alumnos pueden escribir comentarios que son visibles para todos los presentes en el salón de clases. Las encuestas aplicadas para evaluar este sistema indican que los estudiantes prestan más atención, comprenden mejor el material, facilitan y permiten un nivel más alto de interacción creativa y movilidad, por otra parte, ha animado a todo el cuerpo docente a incorporar las TabletPCs como herramientas de enseñanza en el trabajo cotidiano [16, 37].

CAPÍTULO IV.

ESPECIFICACIÓN DEL PROBLEMA Y ANÁLISIS DE REQUERIMIENTOS

EN LOS CAPÍTULOS ANTERIORES SE ESTABLECIÓ EL MARCO TEÓRICO Y LAS PROBLEMÁTICAS ASOCIADAS A LA ENSEÑANZA DEL PARADIGMA ORIENTADO A OBJETOS, ASÍ COMO LAS DISTINTAS METODOLOGÍAS, ESCENARIOS Y HERRAMIENTAS QUE SE HAN DESARROLLADO PARA LOGRAR UN APRENDIZAJE MÁS EFICAZ DE LOS FUNDAMENTOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS. EN ESTE CAPÍTULO SE ESPECIFICA LA PROBLEMÁTICA A LA QUE SE DARÁ SOLUCIÓN, SITUÁNDOLA EN EL CONTEXTO ACTUAL.

PARA LOGRAR UN MEJOR ENTENDIMIENTO DEL PROBLEMA SE EFECTUARÁ LA CAPTURA Y ANÁLISIS DE REQUERIMIENTOS, CON EL OBJETIVO DE DESCRIBIR LAS NECESIDADES QUE DEBERÁ DE CUBRIR EL SOFTWARE Y SE HARÁ UNA LISTA DE LOS REQUERIMIENTOS FUNCIONALES QUE TENDRÁ CUBRIR EL SISTEMA. COMO PARTE DEL ENTENDIMIENTO DEL PROBLEMA SE MODELARÁ EL NEGOCIO A TRAVÉS DE LA CONSTRUCCIÓN DEL MODELO DE CASOS DE USO DEL ENTORNO PARA DOCUMENTARLO; LOS PROCESOS DEL NEGOCIO SE DESCRIBIRÁN EN TÉRMINOS DEL ACTORES Y CASOS DE USO. LOS ACTORES SERÁN ENTIDADES QUE INTERACCIONAN CON EL SISTEMA Y LOS CASOS DE USO MODELARÁN LAS FUNCIONALIDADES COMPLETAS DEL SISTEMA.

ADICIONALMENTE SE CONSTRUIRÁ EL PROTOTIPO DE INTERFAZ DE USUARIO, QUE INCLUYE LAS VENTANAS Y OPCIONES DE MENÚ PRINCIPALMENTE. LA INTERFAZ REAL SE CREARÁ EN PARALELO CON EL SISTEMA MÁS ADELANTE.

IV. ESPECIFICACIÓN DEL PROBLEMA Y ANÁLISIS DE REQUERIMIENTOS

4.1 Especificación del problema

En la actualidad el paradigma orientado a objetos (OO) está ampliamente aceptado y la mayoría de los planes de estudio consideran necesario impartir una o más materias relacionadas con el análisis, diseño y programación orientada a objetos (POO), el *Computing Curricula 2001* de *ACM* e *IEEE Computer Society* señala que la POO significa un cambio relevante en la enseñanza de la computación [14].

Por otro lado, existe un debate mundial en lo que se refiere a la forma en cómo acercar dicho paradigma a los estudiantes de manera que puedan comprender sus conceptos de una forma sencilla, clara y concisa, pues hoy en día el paradigma OO es la mejor opción disponible para el desarrollo de software, pues se trata del paradigma más adecuado para mejorar la calidad del software y favorece aspectos como la reutilización y extensibilidad del código; sin embargo, como ya se ha visto, enseñar POO no es una tarea fácil, de los principales problemas destacan:

- ✓ La POO no puede ser “agregada” a las enseñanzas de otros paradigmas, pues reemplaza la estructura fundamental de la programación.
- ✓ La POO no es una simple extensión de la programación estructurada, sino que se trata de un paradigma completamente diferente.
- ✓ Enseñar POO es más que enseñar la sintaxis y la semántica de un lenguaje de programación orientado a objetos, se trata más bien de enseñar a los alumnos a pensar en clases y objetos tal y como lo hacen en su vida cotidiana.
- ✓ Muchos profesores no comprenden perfectamente el paradigma OO, de manera que en sus clases combinan el enfoque procedural con el OO, lo cual dificulta que los alumnos comprendan las bondades de la POO.
- ✓ Es complicado enseñar a los alumnos a que utilicen su capacidad de abstracción, sin esto, es para ellos difícil pensar en las clases, objetos y

mensajes de la POO, pues se trata de entes abstractos con los que no están acostumbrados a trabajar.

Es evidente la importancia que la POO ha adquirido tanto en la educación universitaria como en la propia industria. Sin embargo, los conceptos clave de la programación y el diseño de este paradigma, como *clases*, *objetos*, *atributos*, *métodos*, *encapsulación*, *herencia*, *polimorfismo*, etc., no son fáciles de entender recurriendo únicamente a la explicación teórica (Macías, 2002). Esto provoca que los alumnos sufran problemas de falta de motivación y vean a la POO como un paradigma muy complicado de entender, comprender y aplicar. Una de las razones de esta falta de motivación es atribuible a los programas que se presentan para ejemplificar algún concepto de la POO pues las más de las veces son ejemplos poco realistas o muy forzados, o programas demasiado complejos, o aquellos que consisten en una entrada simple seguida de un proceso tras el cual se obtiene un solo número o mensaje en la pantalla en la pantalla.

Como se vió en capítulos anteriores, existen en el mercado una buena cantidad de productos que nos brindan algunas soluciones para acercar a los alumnos de una manera más amigable al paradigma OO, sin embargo, esto no es suficiente, es también importante que la labor docente se vuelva más dinámica en cuanto a la enseñanza de la POO, para ello resulta imprescindible la utilización de las herramientas de cómputo, los pizarrones electrónicos y/o TabletPCs, de manera que los alumnos puedan comprender de una forma mucho más fácil y amena cómo es que se ejecuta un programa escrito en un lenguaje OO como Java. Existen algunas Facultades en la UNAM, que cuentan con estas herramientas, por ejemplo, la Facultad de Ingeniería y la Facultad de Ciencias de la UNAM ha emprendido un importante esfuerzo en esta materia. Sin embargo, muchas veces no se obtiene el máximo aprovechamiento de estas tecnologías y en la actualidad se encuentran sub-aprovechadas, ya que son pocos los profesores que se atreven a impartir sus clases con estas herramientas y así tener un mejor aprovechamiento. Muchos de ellos sólo las ocupan para proyectar sus presentaciones y nada más; otra razón por lo que en ocasiones no se aprovechan

estas tecnologías de la información es que no siempre se puede contar con el software adecuado o es muy caro pagar las licencias o no hay la suficiente capacitación. Con este sistema de software se pretende desarrollar una herramienta que apoye a los profesores y que permita utilizar los pizarrones electrónicos para enseñar POO. Se planea incorporar este sistema con los pizarrones electrónicos de la Facultad de Ciencias y la Facultad de Ingeniería en algunos cursos semestrales e intersemestrales (por ejemplo para el curso de Introducción Ciencias de la Computación I para el semestre 2009 – 2 y algunos cursos de introducción a la POO con Java en la Facultad de Ingeniería durante el intersemestre 2009 – 1), de manera que tanto los profesores como los alumnos conozcan y puedan aprovechar esta tecnología y así ofrecer una solución que apoye y facilite la enseñanza y el aprendizaje de la metodología orientada a objetos.

4.1.1 Objetivo

Desarrollar un sistema de software para utilizar pizarrones electrónicos que apoye la enseñanza de la programación orientada a objetos, mostrando a los alumnos la interpretación y ejecución de un programa en lenguaje Java a través de una herramienta visual, que les permita comprender mejor los fundamentos de la metodología orientada a objetos.

4.1.2 Metas

- ✓ Contribuir con una herramienta que apoye a los profesores en la enseñanza de la POO.
- ✓ Aprovechar las capacidades que brindan los pizarrones electrónicos, como un medio para incentivar y facilitar la enseñanza de la metodología orientada a objetos.
- ✓ Utilizar la animación de algoritmos en conjunto con un diseño de calidad y la ejecución en tiempo real de los algoritmos para ayudar a los estudiantes a comprender mejor los conceptos de la POO.

4.2 Análisis de requerimientos

4.2.1 Definición del problema

Se requiere diseñar un sistema para apoyar a un profesor a enseñar la POO. Este sistema ayudará a los profesores a mostrar los conceptos de *creación y uso de objetos, de clases, agregación, herencia y polimorfismo*, temas que corresponden y que cubren los conceptos básicos de la POO. La información podrá ser capturada y presentada en cualquier editor de presentaciones de diapositivas; mismas que posteriormente serán convertidas mediante la herramienta adecuada para publicarlas en un formato que podrá desplegarse dentro del mismo sistema. Mediante un paquete de software que permita la elaboración de animaciones, se creará una biblioteca de clips de película, cuya función será mostrar mediante una animación algún programa en donde se simule el proceso de creación de un objeto a partir de una clase y posteriormente cómo es que estos interactúan con objetos de otras clases para el paso de mensajes y la solución de problemas. Cada uno de los programas que se presentan, están diseñados con el objetivo de mostrar de forma gradual los conceptos de la POO. Adicionalmente el sistema integrará un entorno (que ocupe el entorno Java) para permitirle al profesor escribir un programa desde cero o bien cargar uno predeterminado para mostrar la compilación del mismo y estudiar aspectos de sintaxis y de lógica de los programas, para que en conjunto con las animaciones se presente a los alumnos un enfoque global de lo que ocurre con un programa en particular.

El acceso al sistema de apoyo de la POO podrá ser consultado desde **Internet** o bien desde una máquina local y contará con una **interfaz gráfica** clara y fácil de utilizar.

4.2.2 Requerimientos funcionales para el sistema

a) El sistema permitirá añadir, modificar, eliminar temas o conceptos de una forma fácil. Los temas se podrán escribir utilizando algún editor de diapositivas
b) El sistema permitirá añadir, modificar o eliminar animaciones de forma fácil.
c) Para la parte de animaciones se utilizará un software que proporciona herramientas que permiten realizarlas de forma rápida y completamente configurables.
d) El acceso al material podrá ser por Internet o bien, si se desea, se puede cargar el sistema en una máquina de manera local.
e) El sistema se construirá tomando únicamente como plantilla para el montaje de la película principal, que es la que contendrá las acciones necesarias para cargar el material del sistema.
f) Se diseñará un menú pull – down (desplegable) que facilitará la ubicación de la información para llegar a ella con un solo clic.
g) En una primera versión, los temas que cubrirán este sistema serán extraídos del libro **Introducción al desarrollo de programas con Java**, de la Dra. Amparo López Gaona y editado y publicado por las Prensas de Ciencias.
h) En su primera versión, los ejemplos que se animaran también corresponderán a los ejercicios utilizados por la Dra. Amparo López Gaona en el libro mencionado en el inciso g).
i) El sistema permitirá al usuario utilizar un entorno que le permita escribir, compilar y ejecutar sin necesidad de hacerlo desde la línea de comandos.
j) Para que el entorno que incorpora el sistema pueda hacer la compilación y ejecución de los programas, se deberá tener correctamente instalado Java en la computadora donde se vaya a cargar o bien desde la computadora donde se visualizará.
k) El sistema utilizará un paquete que facilite la publicación del material que los profesores elaboren con el editor de diapositivas de su preferencia.
l) La visualización del sistema será independiente de la plataforma.
m) El sistema deberá incorporar los **Criterios Ergonómicos** [49] para contar con una interfaz efectiva.

4.2.3 Diagrama general de casos de uso

Un diagrama de casos de uso contiene elementos de modelo para el sistema, los actores y los casos de uso y muestra las diferentes relaciones tales como generalización, asociación y dependencia entre estos elementos. El diagrama de caso de uso debe ser fácil de entender por el usuario final. Un caso de uso representa la funcionalidad completa tal y como la percibe un actor. Un caso de uso en UML es definido como un conjunto de secuencias de acciones que un sistema ejecuta y que permite un resultado observable de valores para un actor en particular (figura 9).

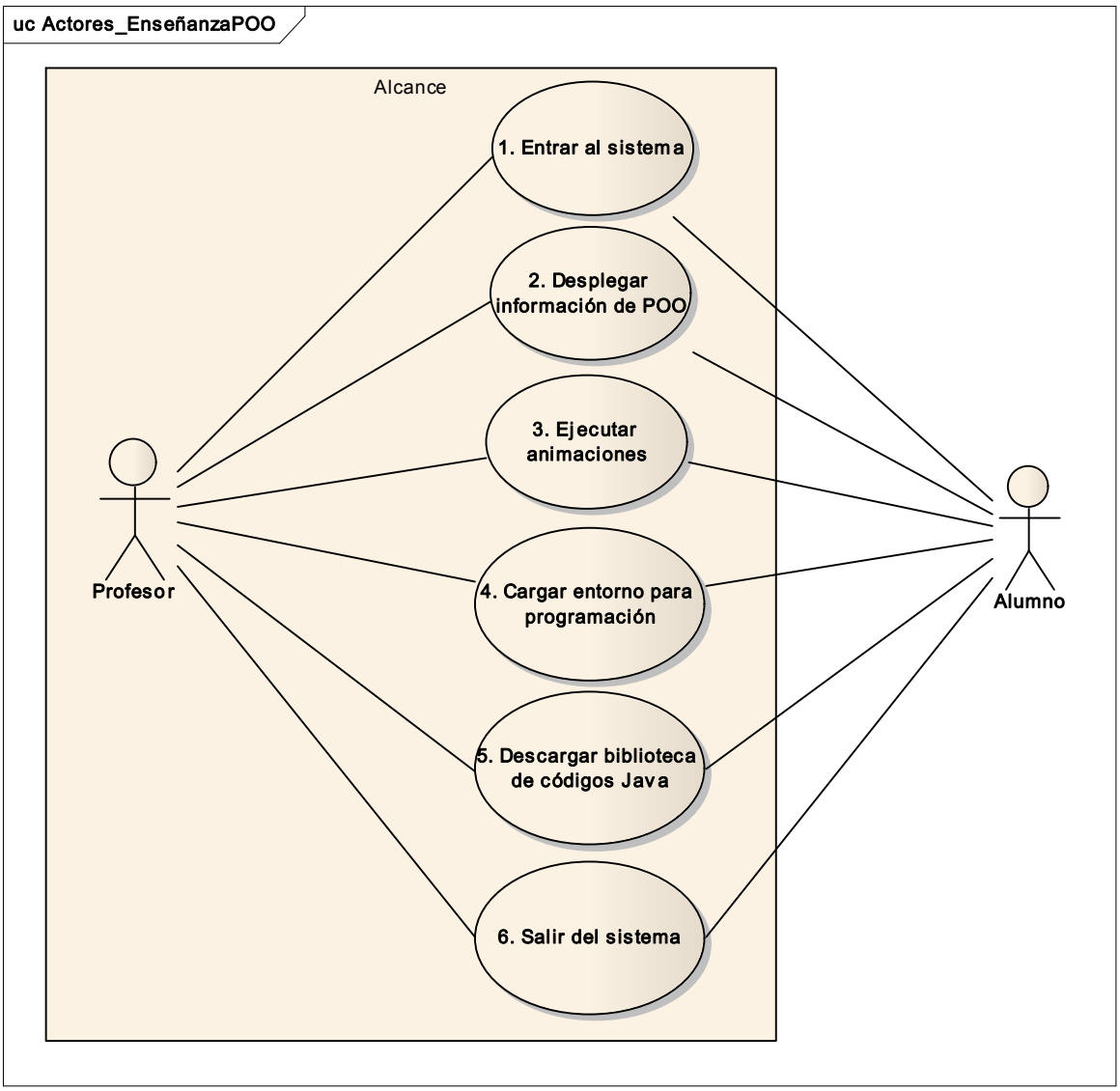


Figura 9. Diagrama general de casos de uso del sistema

4.2.4 Prototipo y detalle de casos de uso

En este apartado se hará el diseño preliminar del conjunto de interfaces que formarán el sistema de software que apoyará la enseñanza de la programación orientada a objetos. Cada una de ellas va acompañada del detalle de casos de uso detectado en el capítulo anterior. No se incluyó la interfaz para salir del sistema, porque, sólo basta con teclear una nueva página en la barra de direcciones o cerrar el navegador.

Caso de uso:	1. Entrar al sistema
Actor:	Profesor

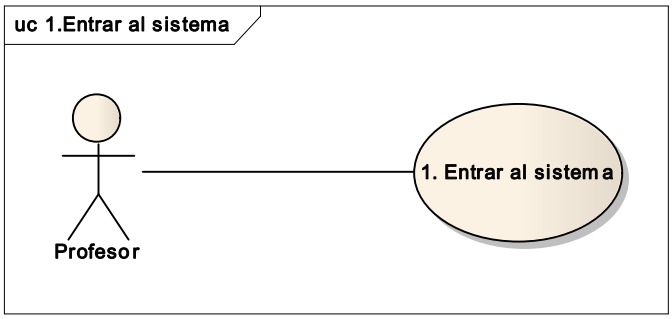


Figura 10. Caso de uso *Entrar al sistema*

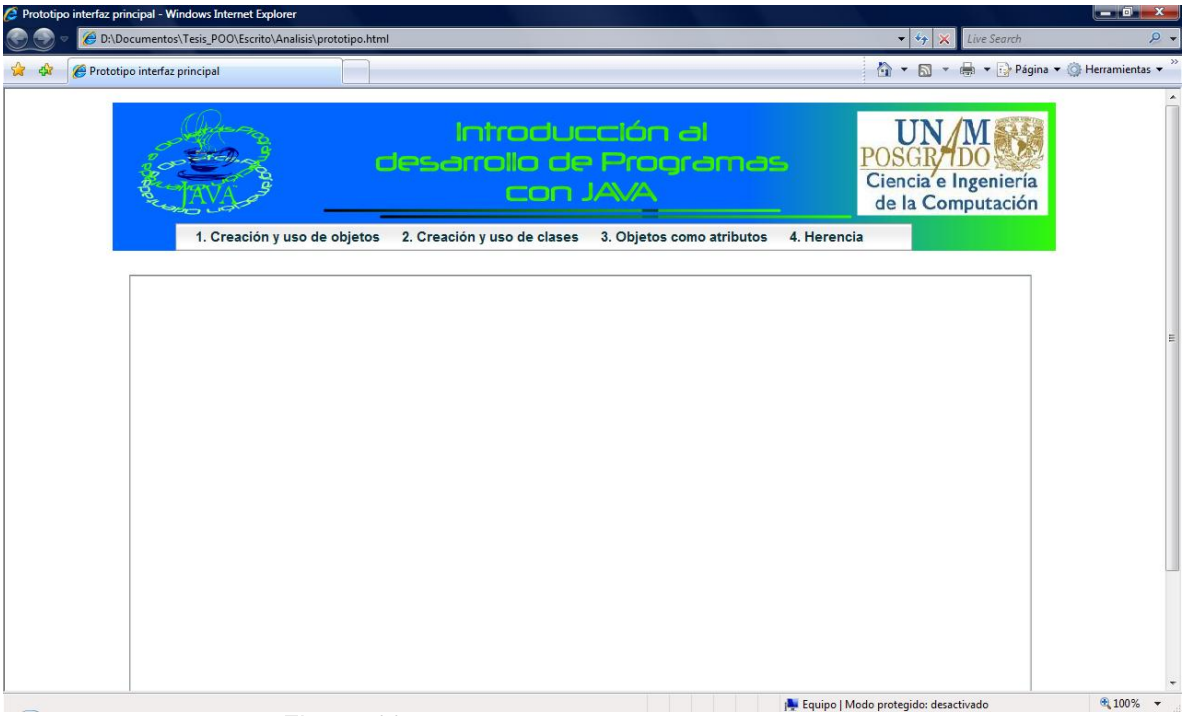


Figura 11. Interfaz de inicio para el usuario *Profesor*

Descripción:	El usuario deberá teclear la dirección de la página del sistema para poder ingresar a la página principal del sistema y comenzar a utilizarlo.

Precondiciones:	✓ El usuario deberá tener una computadora con acceso a Internet. ✓ La computadora deberá contar con el software necesario para ingresar a la página, en este caso, tener algún explorador de Internet (Internet Explorer o Netscape). ✓ El usuario necesita impartir una clase de POO ante un grupo de alumnos. ✓ El usuario deberá de contar con la dirección correcta.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	Abre una página del navegador.			
	2	Teclea en su navegador la dirección de la página del sistema	3	Despliega la interfaz principal del sistema con un menú desplegable, que contiene los vínculos disponibles a cada uno de los temas y subtemas en la aplicación.	E1

	ID	Nombre	Acción
Excepciones:	E1	Dirección incorrecta	La página no se despliega en el navegador y el sistema no hace nada.

Postcondiciones:	✓ El sistema ha desplegado la pantalla principal, la cual contiene un conjunto de menús desplegables con la información disponible sobre POO.

Caso de uso:	2. Desplegar información de POO
Actor:	Profesor

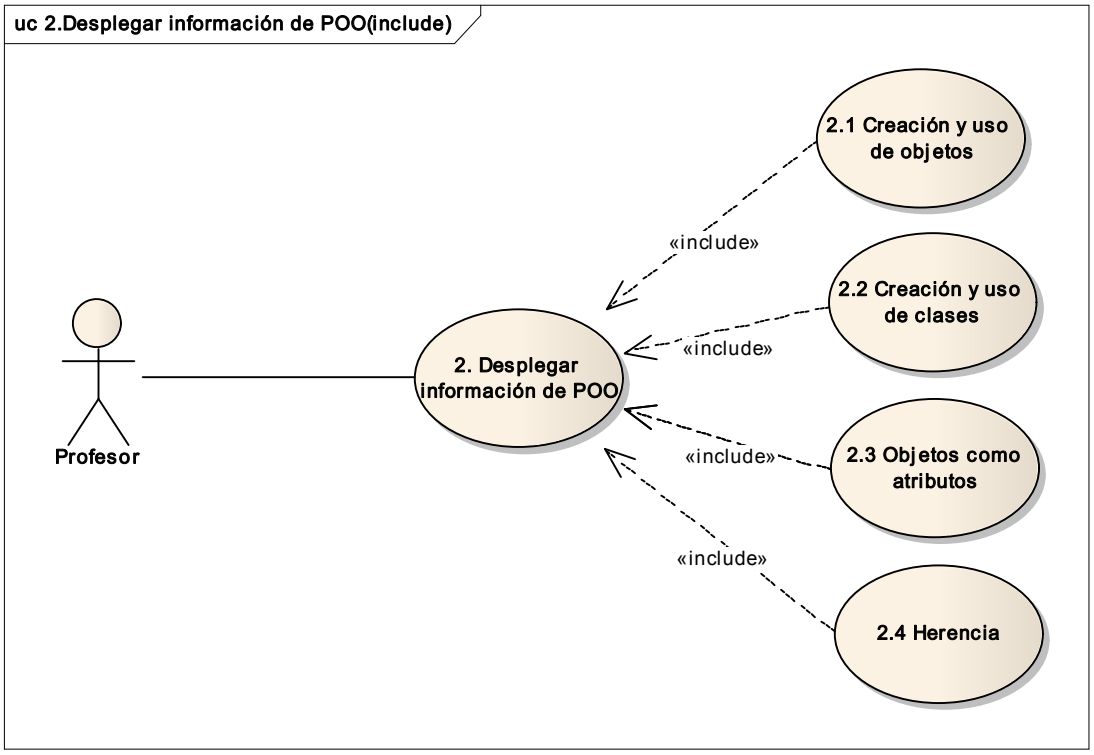


Figura 12. Caso de uso *Desplegar información de POO*

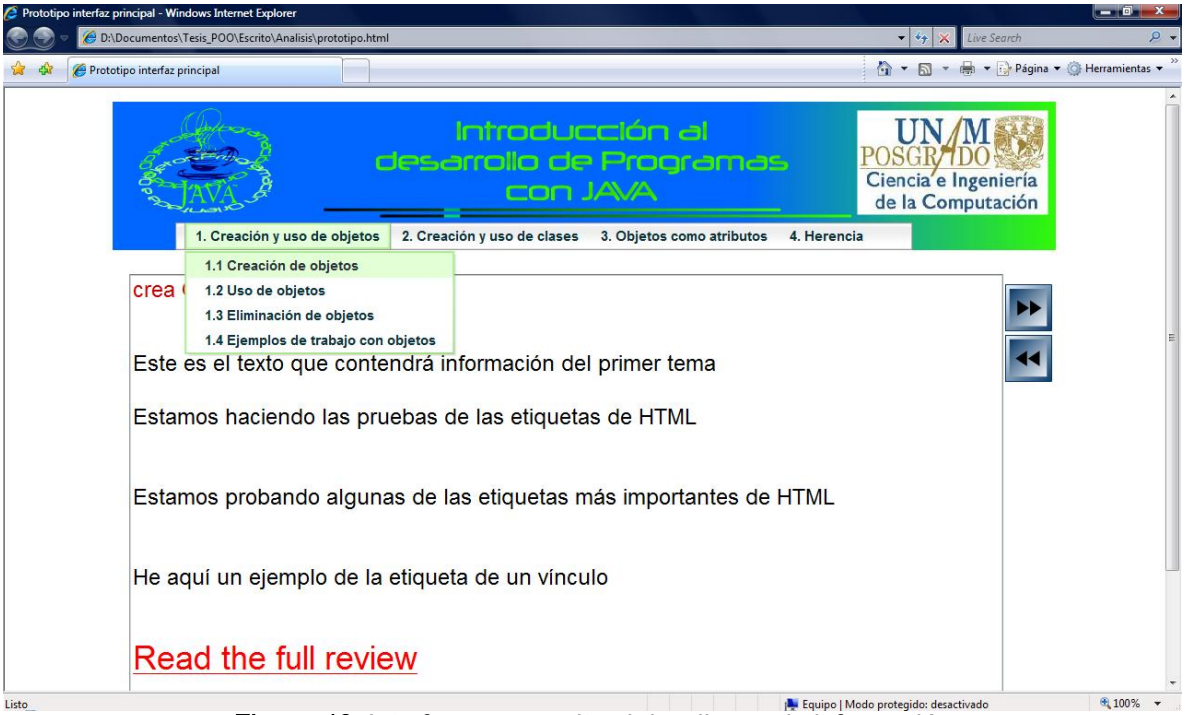


Figura 13. Interfaz que permite el despliegue de información

Descripción:	El usuario puede llevar a cabo el despliegue de la información contenida en la aplicación; además, la aplicación le permite controlar el avance y retroceso de las dispositivas mostradas y la posibilidad de cambiar y/o alternar entre los temas que se encuentran disponibles.

Precondiciones:	✓ El usuario desea mostrar a sus alumnos la información disponible en la aplicación (*Creación y uso de objetos; Creación y uso de clases; Objetos como atributos y Herencia*). ✓ El usuario desea, una vez que ha desplegado la información de un capítulo poder navegar (**avanzar o retroceder**) dentro de la información mostrada para dicho capítulo. ✓ El usuario desea cambiar de capítulo y explorar otro tema.

Flujos:

Actor		Sistema		
Paso	Acción	Paso	Acción	Excepción
1	Para poder desplegar algún tema, el usuario selecciona alguna de las opciones disponibles: **"Creación y uso de objetos", "Creación y uso de clases", "Objetos como atributos"** y **"Herencia".**	2	El sistema despliega la interfaz con el tema correspondiente de acuerdo a la opción seleccionada.	E1, E2, E3, E4
3	Una vez que se despliega la información de un tema, se habilitan los botones que permiten avanzar o retroceder en la información mostrada.	4	El sistema despliega la dispositiva anterior o siguiente, dependiendo de la opción de navegación seleccionada.	E1, E2, E3, E4
5	Para hacer un nuevo despliegue de información el usuario da clic en menú principal y selecciona otro tema disponible.	6	El sistema despliega a través de su menú las opciones de consulta disponibles.	E2, E3, E4

	ID	Nombre	Acción
Excepciones:	E1	Opción equivocada	El sistema despliega la interfaz que contiene la información de acuerdo al tema seleccionado por error.
	E2	No hay información para el tema seleccionado	El sistema muestra una pantalla del explorador para indicar que la información no está disponible o posiblemente fue borrada.
	E3	Página no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E4	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El sistema muestra la pantalla con la información que arroja el tema seleccionado. ✓ El sistema muestra datos con una consulta equivocada, de manera que el usuario debe de seleccionar de nueva cuenta el tema que desea desplegar. ✓ El sistema está listo para mostrar un nuevo tema. ✓ Accidentalmente el usuario salió de la interfaz para desplegar la información y el sistema se posicionó en la interfaz de la opción seleccionada.

A continuación se describirán las opciones que incluye el despliegue de información sobre POO:

Caso de uso: **2.1. Creación y uso de objetos**
Actor: **Profesor**

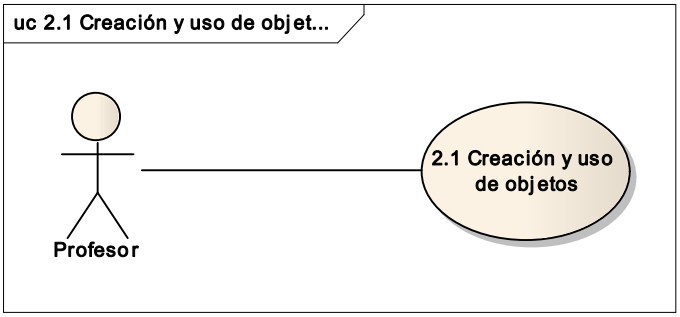


Figura 14. Caso de uso *Creación y uso de objetos*

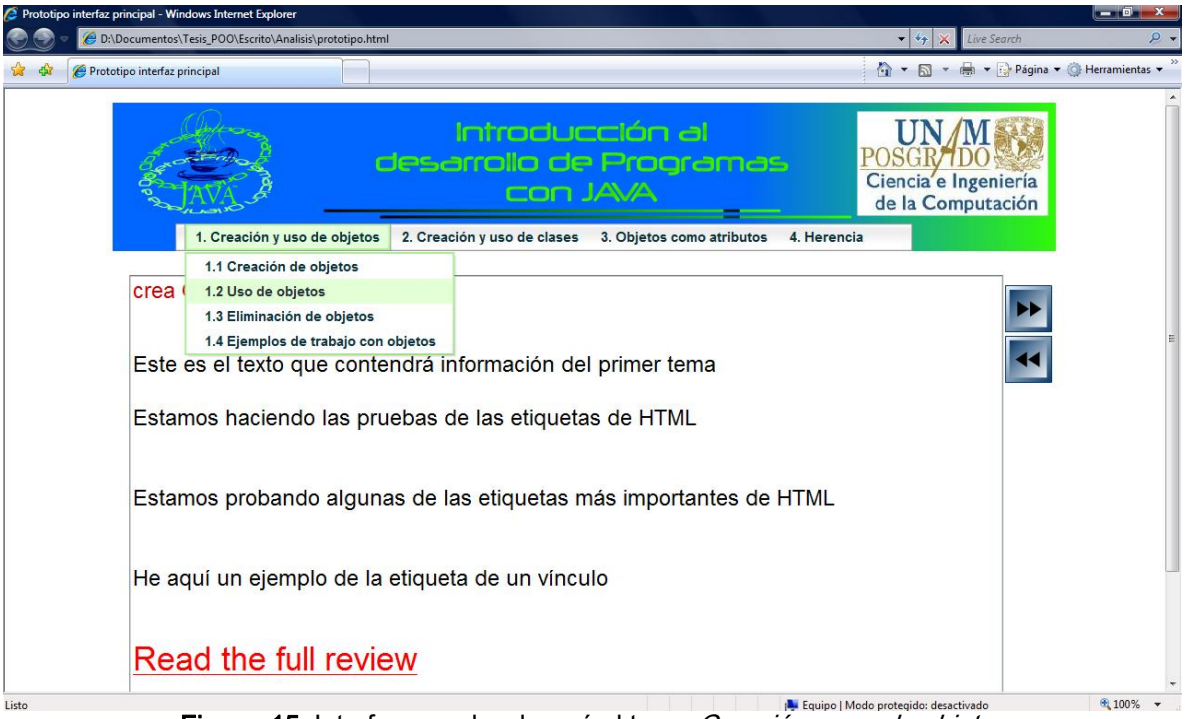


Figura 15. Interfaz que desplegará el tema *Creación y uso de objetos*

Descripción:	El usuario desea consultar y mostrar el tema **Creación y uso de objetos.** Para hacer esto, el usuario selecciona la opción correspondiente del menú desplegable.

Precondiciones:	✓ El usuario quiere desplegar la información disponible para el tema en donde se habla de la creación y uso de objetos. ✓ Le interesan solo aquellos subtemas que contenga el primer capítulo.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	En la interfaz donde se encuentra el menú principal, el usuario selecciona la opción **"Creación y uso de objetos"**.	2	El sistema despliega todos los subtemas disponibles para este tema.	
	3	Una vez que el usuario elige el subtema a revisar da clic en la opción diseñada para tal fin.	4	El sistema muestra una pantalla, donde despliega la información disponible para el subtema seleccionado por el usuario.	E1, E2, E3

Flujos:	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
	5	El usuario desea visitar alguna otra sección del sistema y para ello va al menú principal.	6	El sistema se posiciona en la pantalla correspondiente a la sección seleccionada.	E1, E2, E3

Excepciones:	ID	Nombre	Acción
	E1	Nuevo tema o subtema seleccionado de manera accidental	El sistema despliega la interfaz que contiene los apartados disponibles para el tema o subtema seleccionado.
	E2	Página no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E3	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El sistema muestra la pantalla con la información del tema o subtema seleccionado. ✓ El sistema muestra la pantalla con la información del tema o subtema seleccionado por accidente. ✓ El sistema está listo para que el usuario consulte un nuevo tema. ✓ El sistema muestra una pantalla indicando que no hay información para el tema o subtema seleccionado. ✓ Accidentalmente el usuario salió de la interfaz para desplegar la información de POO y el sistema se posicionó en la interfaz de la opción seleccionada.

Caso de uso: **2.2. Creación y uso de clases**
Actor: **Profesor**

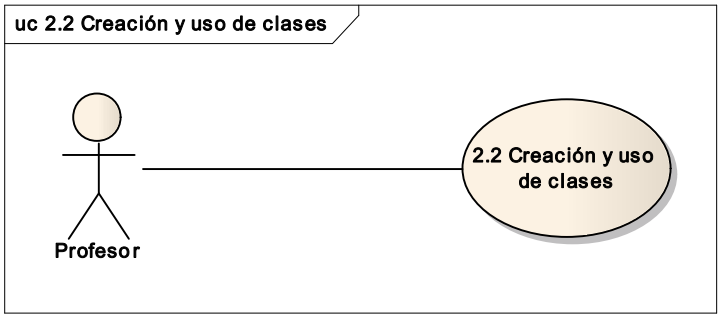


Figura 16. Caso de uso *Creación y uso de clases*

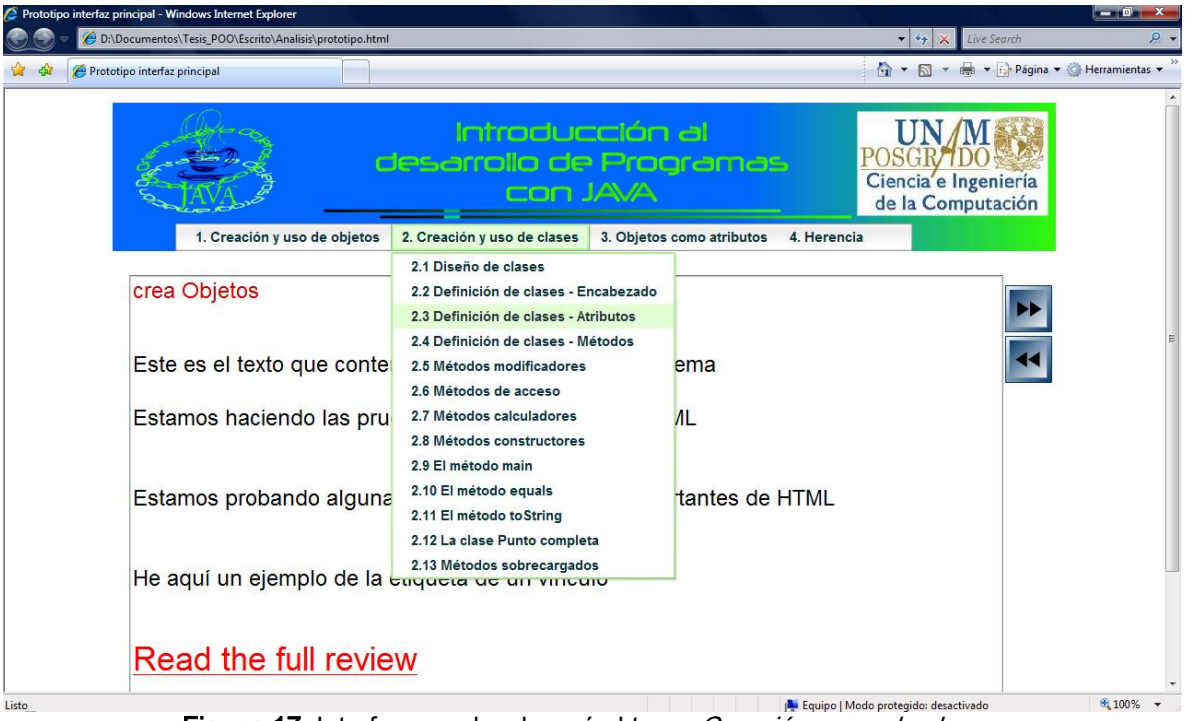


Figura 17. Interfaz que desplegará el tema *Creación y uso de clases.*

Descripción:	El usuario desea consultar y mostrar el tema **Creación y uso de clases.** Para hacer esto, el usuario selecciona la opción correspondiente del menú desplegable.

Precondiciones:	✓ El usuario quiere desplegar la información disponible para el tema en donde se habla de la creación y uso de objetos. ✓ Le interesan solo aquellos subtemas que contenga el segundo capítulo.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	En la interfaz donde se encuentra el menú principal, el usuario selecciona la opción **"Creación y uso de clases"**.	2	El sistema despliega todos los subtemas disponibles para este tema.	
	3	Una vez que el usuario elige el subtema a revisar da clic en la opción diseñada para tal fin.	4	El sistema muestra una pantalla, donde despliega la información disponible para el subtema seleccionado por el usuario.	E1, E2, E3

Flujos:	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
	5	El usuario desea visitar alguna otra sección del sistema y para ello va al menú principal.	6	El sistema se posiciona en la pantalla correspondiente a la sección seleccionada.	E1, E2, E3

Excepciones:	ID	Nombre	Acción
	E1	Nueva tema o subtema seleccionado de manera accidental	El sistema despliega la interfaz que contiene los apartados disponibles para el tema o subtema seleccionado.
	E2	Página no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E3	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El sistema muestra la pantalla con la información del tema o subtema seleccionado. ✓ El sistema muestra la pantalla con la información del tema o subtema seleccionado por accidente. ✓ El sistema está listo para que el usuario consulte un nuevo tema. ✓ El sistema muestra una pantalla indicando que no hay información para el tema o subtema seleccionado. ✓ Accidentalmente el usuario salió de la interfaz para desplegar la información de POO y el sistema se posicionó en la interfaz de la opción seleccionada.

Caso de uso: **2.3. Objetos como atributos**
Actor: **Profesor**

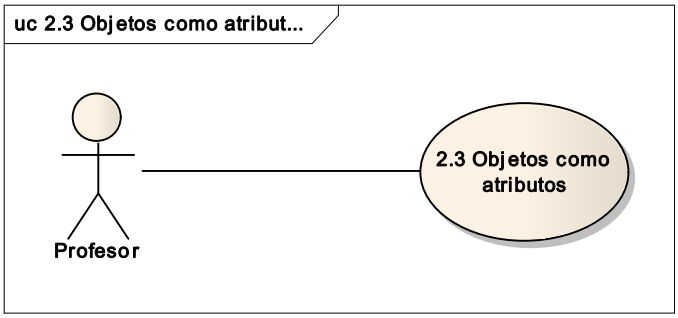


Figura 18. Caso de uso *Objetos como atributos*

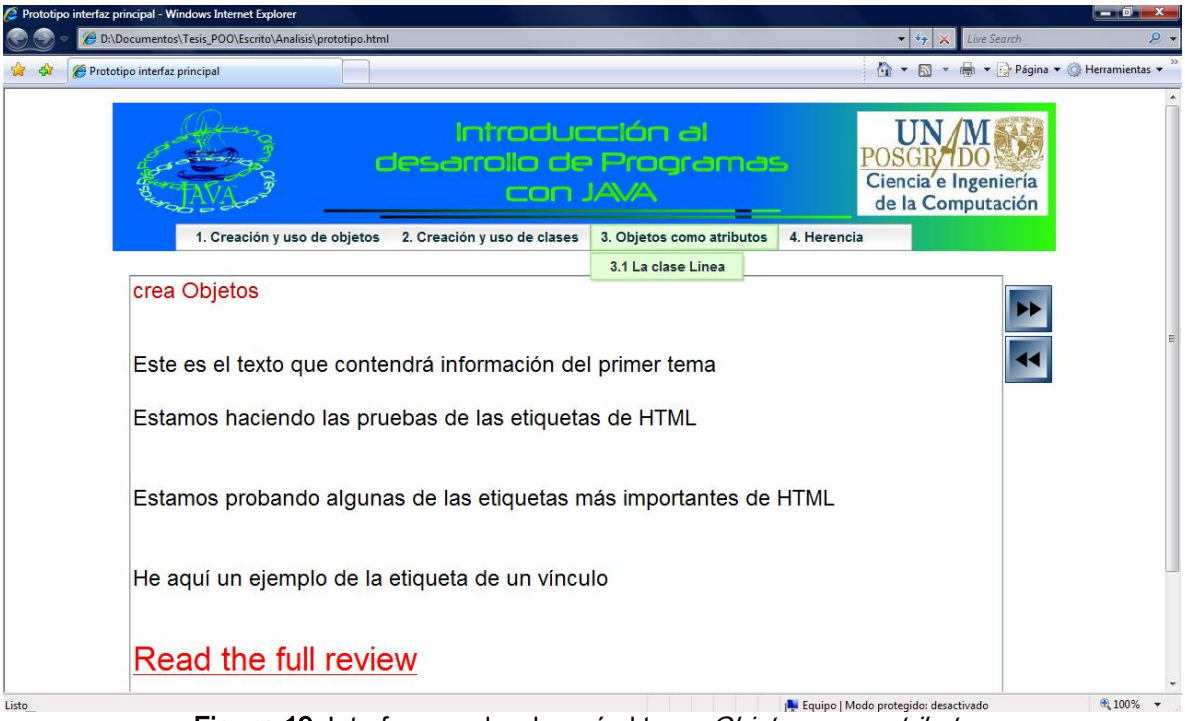


Figura 19. Interfaz que desplegará el tema *Objetos como atributos.*

Descripción:	El usuario desea consultar y mostrar el tema **Objetos como atributos.** Para hacer esto, el usuario selecciona la opción correspondiente del menú desplegable.

Precondiciones:	✓ El usuario quiere desplegar la información disponible para el tema en donde se habla del concepto de agregación en los objetos. ✓ Le interesan solo aquellos subtemas que contenga el tercer capítulo.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	En la interfaz donde se encuentra el menú principal, el usuario selecciona la opción **"Objetos como atributos"**.	2	El sistema despliega todos los subtemas disponibles para este tema.	
	3	Una vez que el usuario elige el subtema a revisar da clic en la opción diseñada para tal fin.	4	El sistema muestra una pantalla, donde despliega la información disponible para el subtema seleccionado.	E1, E2, E3

Flujos:	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
	5	El usuario desea visitar alguna otra sección del sistema y para ello va al menú principal.	6	El sistema se posiciona en la pantalla correspondiente a la sección seleccionada.	E1, E2, E3

Excepciones:	ID	Nombre	Acción
	E1	Nueva tema o subtema seleccionado de manera accidental	El sistema despliega la interfaz que contiene los apartados disponibles para el tema o subtema seleccionado.
	E2	Página no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E3	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El sistema muestra la pantalla con la información del tema o subtema seleccionado. ✓ El sistema muestra la pantalla con la información del tema o subtema seleccionado por accidente. ✓ El sistema está listo para que el usuario consulte un nuevo tema. ✓ El sistema muestra una pantalla indicando que no hay información para el tema o subtema seleccionado. ✓ Accidentalmente el usuario salió de la interfaz para desplegar la información de POO y el sistema se posicionó en la interfaz de la opción seleccionada.

Caso de uso: **2.4. Herencia**
Actor: **Profesor**

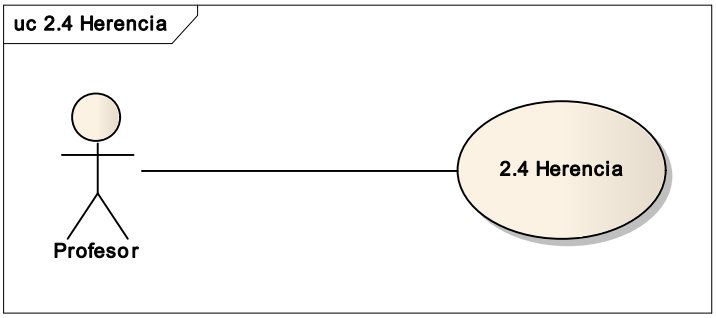


Figura 20. Caso de uso *Herencia*

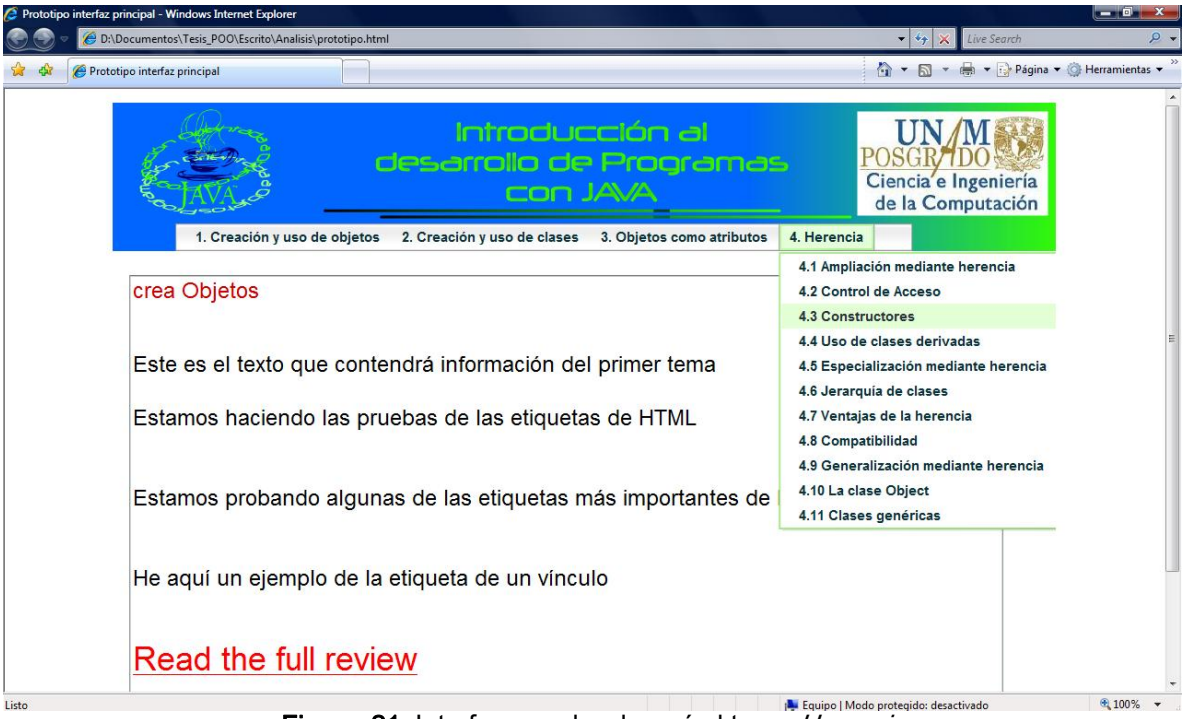


Figura 21. Interfaz que desplegará el tema *Herencia*

Descripción:	El usuario desea consultar y mostrar el tema **Herencia.** Para hacer esto, el usuario selecciona la opción correspondiente del menú desplegable.

Precondiciones:	✓ El usuario quiere desplegar la información disponible para el tema en donde se habla del concepto de herencia de clases. ✓ Le interesan solo aquellos subtemas que contenga el cuarto capítulo.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	En la interfaz donde se encuentra el menú principal, el usuario selecciona la opción **"Herencia"**.	2	El sistema despliega todos los subtemas disponibles para este tema.	
	3	Una vez que el usuario elige el subtema a revisar da clic en la opción diseñada para tal fin.	4	El sistema muestra una pantalla, donde despliega la información disponible para el subtema seleccionado.	E1, E2, E3

Flujos:	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
	5	El usuario desea visitar alguna otra sección del sistema y para ello va al menú principal.	6	El sistema se posiciona en la pantalla correspondiente a la sección seleccionada.	E1, E2, E3

Excepciones:	ID	Nombre	Acción
	E1	Nueva tema o subtema seleccionado de manera accidental	El sistema despliega la interfaz que contiene los apartados disponibles para el tema o subtema seleccionado.
	E2	Página no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E3	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El sistema muestra la pantalla con la información del tema o subtema seleccionado. ✓ El sistema muestra la pantalla con la información del tema o subtema seleccionado por accidente. ✓ El sistema está listo para que el usuario consulte un nuevo tema. ✓ El sistema muestra una pantalla indicando que no hay información para el tema o subtema seleccionado. ✓ Accidentalmente el usuario salió de la interfaz para desplegar la información de POO y el sistema se posicionó en la interfaz de la opción seleccionada.

Caso de uso: **3. Ejecutar animaciones**
Actor: **Profesor**

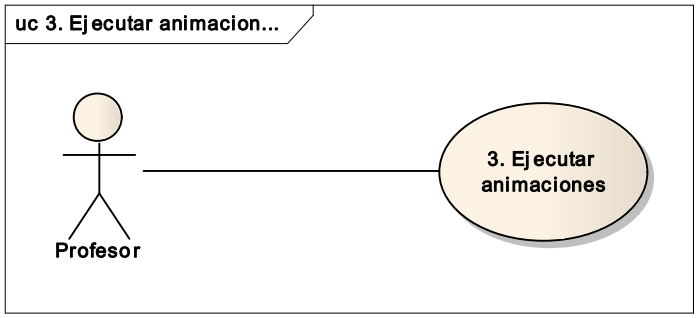


Figura 22. Caso de uso *Ejecutar animaciones*

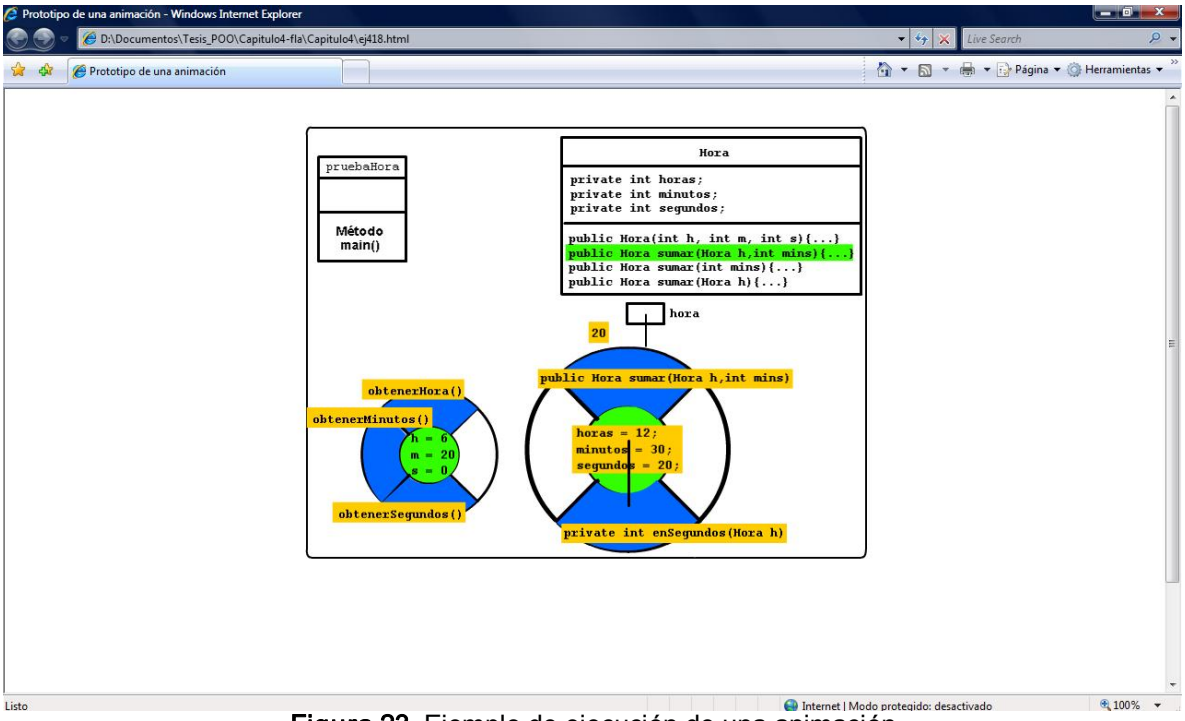


Figura 23. Ejemplo de ejecución de una animación

Descripción:	El usuario desea ejecutar las diversas animaciones que se encuentran a lo largo de los capítulos y que sirven como apoyo para entender conceptos críticos de la POO. El usuario desea poder ejecutarlas en una ventana independiente de la aplicación, de manera que pueda moverla y cerrarla con libertad. Para hacer esto, el usuario da clic en los botones diseñados para tal fin.

Precondiciones:	✓ El usuario ejecutar una animación de las incluidas, para ilustrar un concepto o ver la ejecución de un programa. ✓ Le interesa ejecutar solo aquellas animaciones contenidas a lo largo de los capítulos del sistema.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	En la interfaz donde se despliega la información disponible de cada tema, el usuario encuentra botones que le permiten abrir y/o ejecutar las animaciones para ilustrar un concepto o un programa.	2	El sistema muestra la animación en una ventana independiente de la interfaz donde se despliega la información, con los botones necesarios para ejecutar o repetir la animación.	

Flujos:	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
	3	Una vez desplegada la animación, el usuario da clic en los botones incluidos en las animaciones para ejecutarlas o bien repetir la ejecución de las mismas.	4	El sistema muestra la ejecución de la animación. Las animaciones se detienen cuando se necesita del concurso del usuario para continuar con la ejecución o bien para repetir una ejecución.	E1, E2, E3, E4, E5
	5	El usuario desea continuar con la ejecución de una animación y da clic en el botón diseñado para tal fin.	6	El sistema se continúa con la ejecución de la animación, mientras no se necesite al usuario para continuar o mientras la animación no haya terminado.	E1, E2, E3, E4, E5
	7	El usuario desea repetir la ejecución de una animación para reforzar la visualización del concepto o la ejecución de un programa.	8	El sistema vuelve a reproducir la animación.	E1, E2, E3, E4, E5

Excepciones:	ID	Nombre	Acción
	E1	Opción equivocada	La animación realiza la acción que el usuario por accidente haya seleccionado.
	E2	Animación no disponible	El sistema indica que no encuentra la animación especificada, ya sea porque se borró o no esta correcto el vínculo.
	E3	Animación errónea	El sistema despliega una animación que no corresponde con el tema o concepto visto.

Excepciones:	ID	Nombre	Acción
	E4	Página no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E5	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El sistema muestra una ventana con la animación de un tema o subtema determinado. ✓ El sistema muestra una ventana con la animación del tema o subtema seleccionado por accidente. ✓ El sistema está listo para que el usuario ejecute una nueva animación. ✓ El sistema muestra una pantalla indicando que no hay animación disponible para el tema o subtema seleccionado. ✓ Accidentalmente el usuario salió de la interfaz para desplegar la información de POO y el sistema se posiciono en la interfaz de la opción seleccionada.

Caso de uso: **4. Cargar entorno para programación**
Actor: **Profesor**

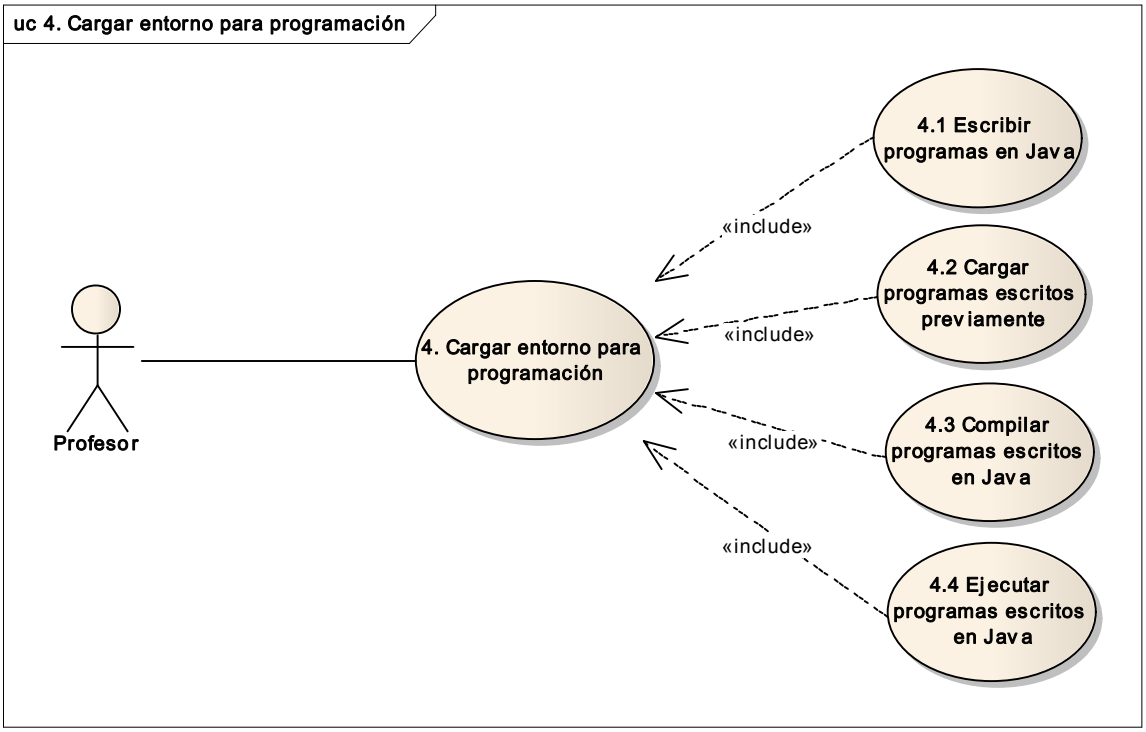


Figura 24. Caso de uso *Cargar entorno para programación*

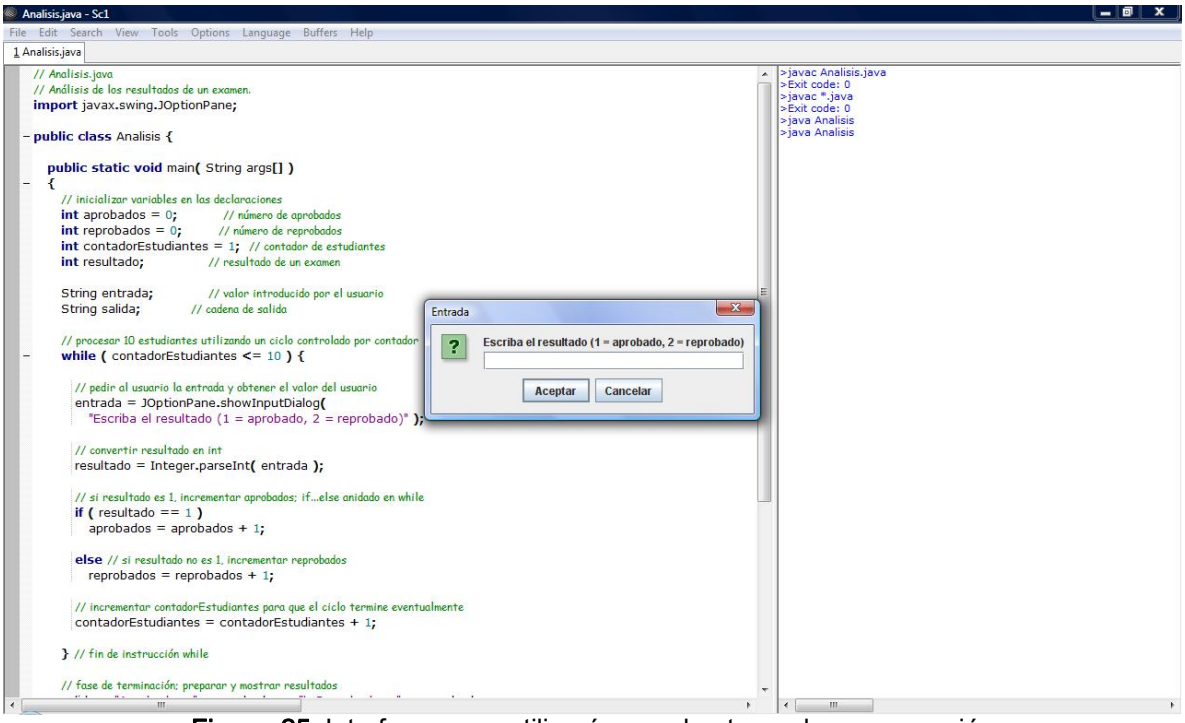


Figura 25. Interfaz que se utilizará para el entorno de programación

Descripción:	El usuario puede cargar un entorno ligero que le servirá para poner en práctica los conceptos vistos en a lo largo de la aplicación. Dicho entorno le permitirá: escribir y cargar programas Java, compilarlos y ejecutarlos.

Precondiciones:	✓ El usuario desea mostrar a sus alumnos la aplicación de los conceptos vistos en la aplicación: *Creación y uso de objetos; Creación y uso de clases; Objetos como atributos y Herencia*. ✓ El usuario desea mostrar diversos escenarios que se pueden presentar al programar utilizando el lenguaje orientado a objetos Java. ✓ El usuario desea mostrar los principales errores que se cometen cuando se escribe un programa en Java. ✓ El usuario desea mostrar cómo se compila y ejecuta un programa en Java. ✓ El usuario debe contar con una computadora con el kit de de desarrollo Java (JDK) instalado. ✓ El usuario debe contar con una computadora que tenga las variables de entorno Java correctamente configuradas.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	Para poder cargar el entorno de programación el usuario deberá dar clic en el botón creado para tal fin en la interfaz principal.	2	El sistema carga el entorno de programación y lo deja listo para que el usuario comience a programar.	
	3	El usuario puede escribir o cargar un programa y compilarlo.	4	El entorno muestra las opciones para cargar o guardar el programa escrito y compilarlo.	E1, E2, E3, E5, E6,
	5	El usuario puede ejecutar su programa para ver los resultados.	6	El entorno muestra los resultados de la ejecución del programa.	E1, E4, E5, E6, E7

	ID	Nombre	Acción
Excepciones:	E1	Archivo no encontrado	El entorno le indica al usuario que el archivo que quiere cargar no se encuentra disponible.
	E2	Archivo almacenado con el mismo nombre	El entorno le indica al usuario que ya existe otro archivo con ese nombre y si desea reemplazarlo
	E3	Errores de compilación.	El entorno le indica al usuario si su programa tiene errores, de que tipo y donde ubicarlos.
	E4	Errores de ejecución	El programa se ejecuta con ciertas inconsistencias.
	E5	Variables de entorno de Java no configuradas.	El entorno le indica al usuario que no puede encontrar los comandos para compilación y ejecución.
	E6	JDK no instalado	El entorno le indica al usuario que el JDK de Java no está instalado.
	E7	Clase no encontrada	El entorno le indica al usuario cuando necesita de alguna clase para la correcta ejecución de un programa.

Postcondiciones:	✓ El sistema muestra el entorno ligero de programación. ✓ El entorno está listo para que el usuario escriba sus programas. ✓ El entorno está listo para cargar un archivo existente. ✓ El entorno está listo para compilar y ejecutar un programa. ✓ El entorno muestra los errores de compilación si los hay ✓ El entorno indica que no hay JDK y/o no están bien configuradas las variables de entorno.

A continuación se describirán las opciones que incluye la carga del entorno para programación:

Caso de uso: **4.1. Escribir programas en Java**
Actor: **Profesor**

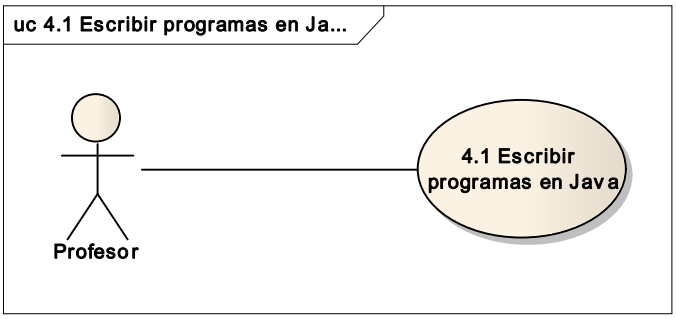


Figura 26. Caso de uso *Escribir programas en Java*

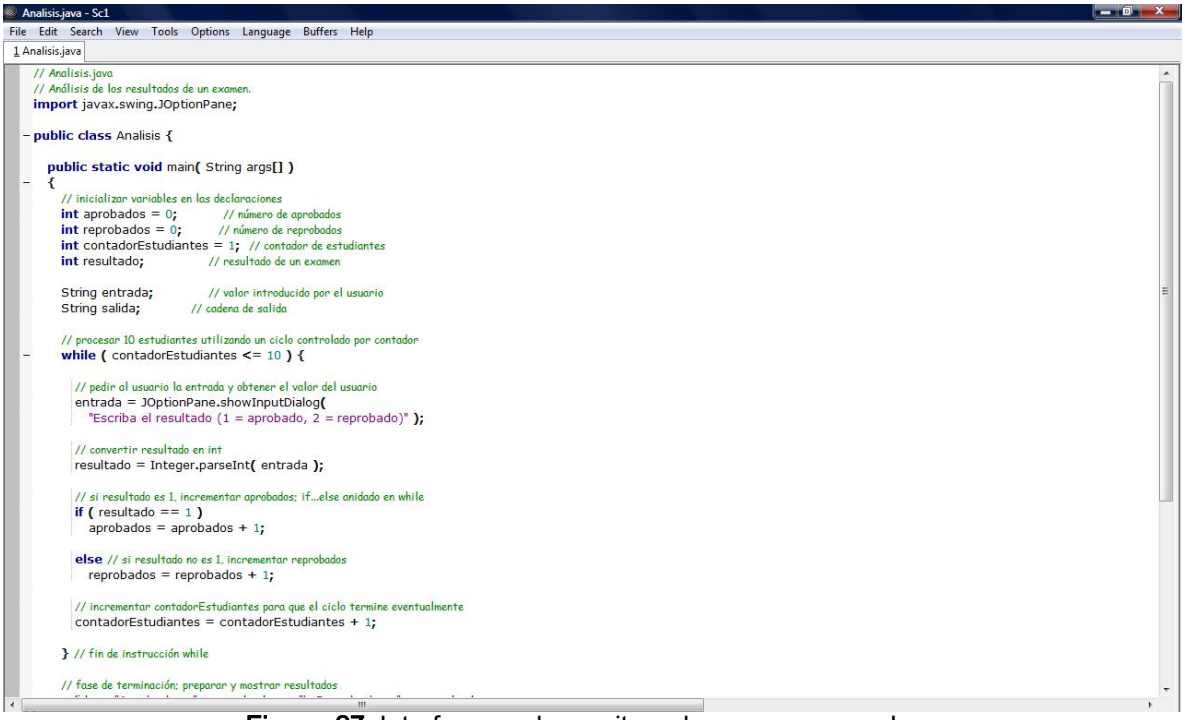


Figura 27. Interfaz para la escritura de programas en Java

Descripción:	El usuario desea mostrar cómo se escribe un programa en leguaje Java desde cero. Para hacer esto, el usuario comienza a teclear su programa en el área de trabajo destinada para ello.

Precondiciones:	✓ El usuario escribir un programa que refuerce los conceptos vistos a lo largo de la aplicación. ✓ Le interesa mostrar los escenarios a los que se puede enfrentar un alumno cuando desea escribir sus programas. ✓ El usuario desea mostrar los principales puntos de la sintaxis de Java.

Flujos:

Actor		Sistema		
Paso	Acción	Paso	Acción	Excepción
1	El usuario comienza a teclear su programa en el área de trabajo del entorno.			
2	Una vez que terminó de teclear su programa, lo guarda con algún nombre.	3	El sistema despliega una interfaz que le permite guardar su programa con algún nombre en particular.	E2
4	El usuario desea visitar alguna otra sección del sistema y para ello minimiza el entorno.	5	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E3
5	El usuario desea visitar alguna otra sección del sistema y para ello cierra el entorno.	6	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E3

	ID	Nombre	Acción
Excepciones:	E1	Archivo almacenado con el mismo nombre	El entorno le indica al usuario que ya existe otro archivo con ese nombre y si desea reemplazarlo
	E2	Entorno no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E3	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El entorno muestra el programa que el usuario tecleo. ✓ El sistema muestra la pantalla en la que el usuario se encontraba justo antes de cargar el entorno, en caso de haber minimizado o cerrado el entorno.

Caso de uso: **4.2. Cargar programas escritos previamente**
Actor: **Profesor**

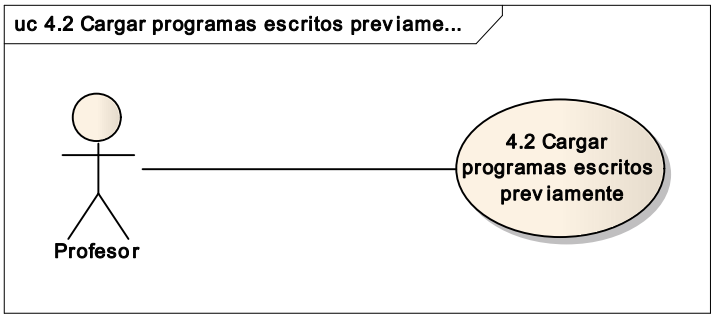


Figura 28. Caso de uso *Cargar programas escritos previamente*

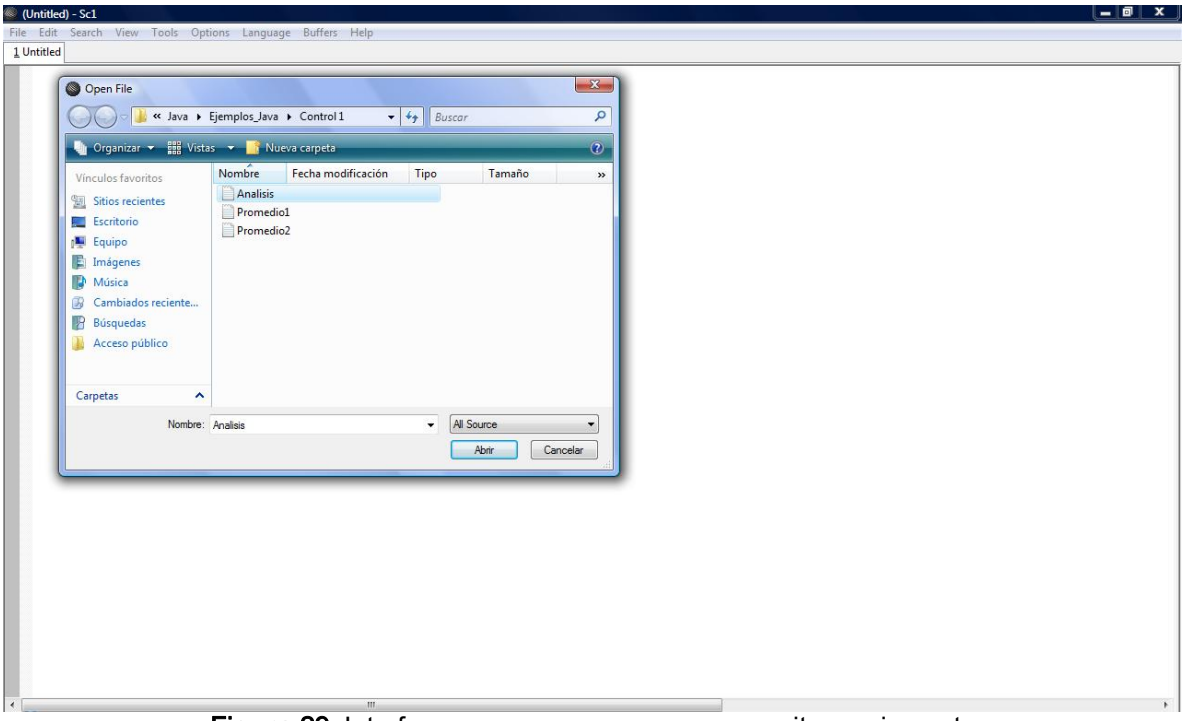


Figura 29. Interfaz para cargar un programa escrito previamente

Descripción:	El usuario ya cuenta con algunos programas escritos previamente y desea cargarlos en el entorno para ilustrar algunos conceptos. Para hacer esto, el usuario selecciona la opción de abrir archivo.

Precondiciones:	✓ El usuario desea ahorrar tiempo y prefiere cargar programas escritos previamente. ✓ El usuario desea mostrar los principales puntos de la sintaxis de Java.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	El usuario selecciona la opción de abrir archivo para cargar un archivo cargado previamente.	2	El entorno muestra un interfaz que permite navegar en las diferentes ubicaciones en la computadora o en un medio de almacenamiento extraíble.	
			3	El sistema despliega el programa.	E1

Flujos:	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
	4	El usuario desea visitar alguna otra sección del sistema y para ello minimiza el entorno.	5	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E3
	5	El usuario desea visitar alguna otra sección del sistema y para ello cierra el entorno.	6	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E3

Excepciones:	ID	Nombre	Acción
	E1	Archivo no encontrado	El entorno le indica al usuario que el archivo que quiere cargar no se encuentra disponible.
	E2	Entorno no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E3	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El entorno muestra el programa que el usuario cargó. ✓ El sistema muestra la pantalla en la que el usuario se encontraba justo antes de cargar el entorno, en caso de haber minimizado o cerrado el entorno.

Caso de uso: **4.3. Compilar programas escritos en Java**
Actor: **Profesor**

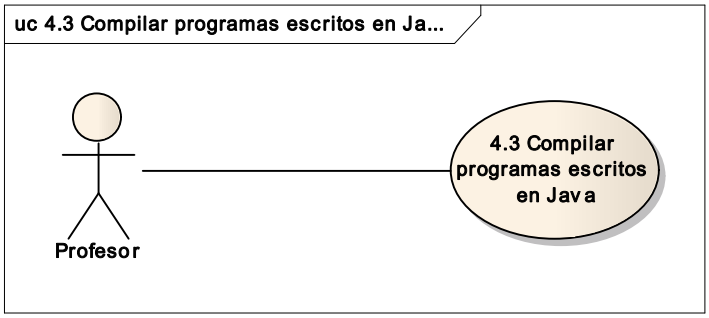


Figura 30. Caso de uso *Compilar programas escritos en Java*

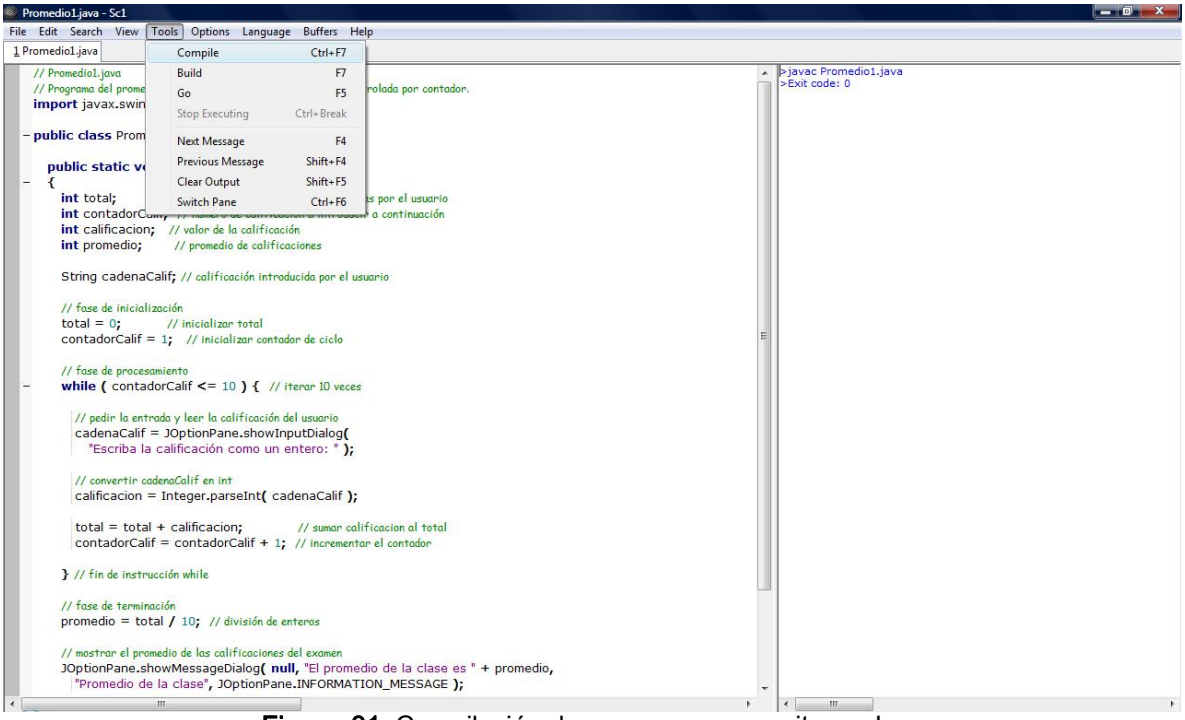


Figura 31. Compilación de un programa escrito en Java

Descripción:	El usuario ya tiene un programa y desea compilarlo. Para hacer esto, el usuario selecciona la opción de compilar en el entorno o bien teclear el comando **javac NombreArchivo.java**.

Precondiciones:	✓ Desea mostrar la forma en que se compilan los programas escritos en Java. ✓ El usuario desea mostrar los principales errores que se pueden presentar durante este proceso.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	El usuario selecciona la opción **compilar**, para iniciar este proceso o bien teclea el comando en el área de comandos del entorno.	2	El entorno compila el programa indicado.	E2, E3
			3	El entorno le indica al usuario que su programa fue compilado exitosamente.	E1, E2, E3

Flujos:	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
	4	El usuario desea visitar alguna otra sección del sistema y para ello minimiza el entorno.	5	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E4
	5	El usuario desea visitar alguna otra sección del sistema y para ello cierra el entorno.	6	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E4

Excepciones:	ID	Nombre	Acción
	E1	Errores de compilación.	El entorno le indica al usuario si su programa tiene errores, de que tipo y donde ubicarlos.
	E2	Variables de entorno de Java no configuradas.	El entorno le indica al usuario que no puede encontrar los comandos para compilación y ejecución.
	E3	JDK no instalado	El entorno le indica al usuario que el JDK de Java no está instalado.
	E4	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El entorno le indica al usuario que la compilación fue exitosa. ✓ El entorno le muestra al usuario los tipos de errores y donde ubicarlos en su programa ✓ El entorno indica que no encuentra el JDK y/o las variables de entorno no están bien configuradas. ✓ El sistema muestra la pantalla en la que el usuario se encontraba justo antes de cargar el entorno, en caso de haber minimizado o cerrado el entorno.

Caso de uso:	4.4. Ejecutar programas escritos en Java
Actor:	Profesor

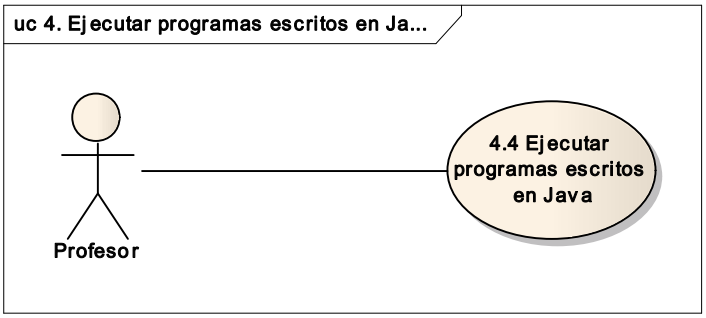


Figura 32. Caso de uso *Ejecutar programas escritos en Java*

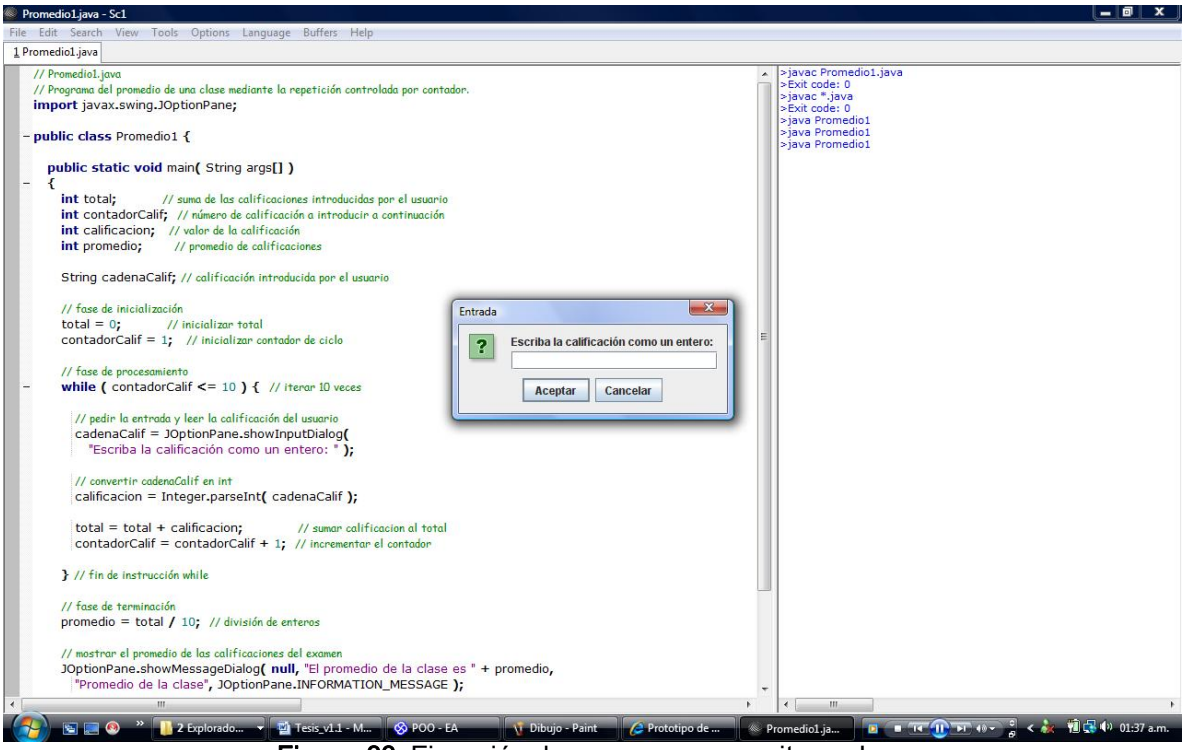


Figura 33. Ejecución de un programa escrito en Java

Descripción:	El usuario ya tiene un programa compilado y desea ejecutarlo. Para hacer esto, el usuario selecciona la opción de ejecutar en el entorno o bien teclear el comando **java NombreArchivo**.

Precondiciones:	✓ Desea mostrar la forma en que se ejecutan los programas escritos en Java. ✓ El usuario desea mostrar los principales errores de ejecución que se pueden presentar durante este proceso.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	El usuario selecciona la opción **ejecutar**, para iniciar este proceso o bien teclea el comando en el área de comandos del ntorno.	2	El entorno ejecuta el programa indicado.	E3
			3	El entorno muestra los resultados de la ejecución del programa.	E1, E2, E3
	4	El usuario desea visitar alguna otra sección del sistema y para ello minimiza el entorno.	5	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E4
	5	El usuario desea visitar alguna otra sección del sistema y para ello cierra el entorno.	6	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E4

	ID	Nombre	Acción
Excepciones:	E1	Errores de ejecución	El programa se ejecuta con ciertas inconsistencias.
	E2	Variables de entorno de Java no configuradas.	El entorno le indica al usuario que no puede encontrar los comandos para compilación y ejecución.
	E3	JDK no instalado	El entorno le indica al usuario que el JDK de Java no está instalado.
	E4	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El entorno muestra los resultados de la ejecución del programa. ✓ El usuario puede ver las inconsistencias de su programa durante la ejecución del mismo. ✓ El entorno indica que no encuentra el JDK y/o las variables de entorno no están bien configuradas. ✓ El sistema muestra la pantalla en la que el usuario se encontraba justo antes de cargar el entorno, en caso de haber minimizado o cerrado el entorno.

Caso de uso: **5. Descargar biblioteca de códigos Java**
Actor: **Profesor**

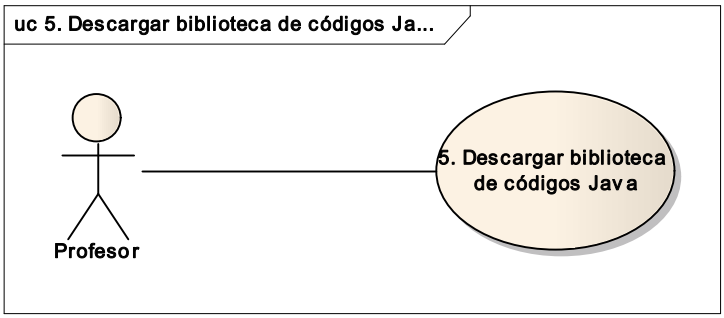


Figura 34. Caso de uso *Descargar biblioteca de códigos Java*

Descripción:	El sistema le proporciona al usuario una especie de biblioteca con los códigos que se trabajarán a lo largo de la aplicación de manera que pueda ver no solo las animaciones, sino la compilación y ejecución de los mismos. Para hacer esto, el usuario selecciona la opción de **Descargar biblioteca de códigos**.

Precondiciones:	✓ El usuario desea ahorrar tiempo y prefiere cargar programas escritos previamente. ✓ El usuario desea mostrar los principales puntos de la sintaxis de Java. ✓ Mostrar la compilación y ejecución del programa en tiempo real.

	Actor		Sistema		
Flujos:	Paso	Acción	Paso	Acción	Excepción
	1	El usuario selecciona la opción de **Descargar biblioteca de códigos**.	2	El sistema comienza la descarga de la Biblioteca.	E1

Flujos:	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
	3	El usuario elige entre guardar la Biblioteca y/o abrirla			E1, E2
	4	El usuario desea visitar alguna otra sección del sistema y para ello minimiza el entorno.	5	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E3
	5	El usuario desea visitar alguna otra sección del sistema y para ello cierra el entorno.	6	El sistema se posiciona en la pantalla correspondiente a la sección en la que se encontraba antes de abrir el entorno.	E3

Excepciones:	ID	Nombre	Acción
	E1	Archivo no encontrado	El entorno le indica al usuario que el archivo que quiere descargar no se encuentra disponible.
	E2	Entorno no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.
	E3	Cancelación de la operación cuando se dé clic en algún otro botón.	El sistema se posiciona en la pantalla correspondiente al botón seleccionado.

Postcondiciones:	✓ El usuario descargó la biblioteca con los ejemplos del sistema. ✓ El explorador muestra una pantalla donde le indica al usuario que el archivo no está disponible. ✓ El sistema muestra la pantalla en la que el usuario se encontraba justo antes de cargar el entorno, en caso de haber minimizado o cerrado el entorno.

Caso de uso:	6. Salir del sistema
Actor:	Profesor

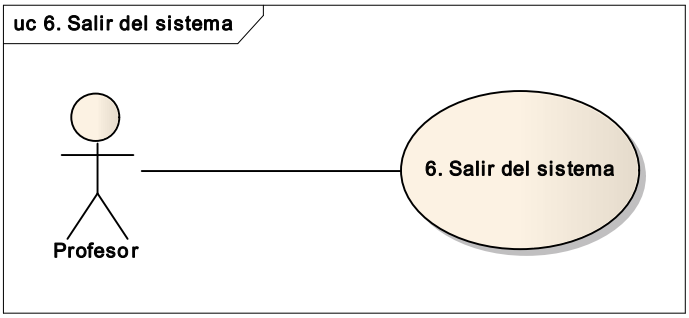


Figura 35. Caso de uso *Salir del sistema*

Descripción:	El usuario ha realizado todas las operaciones que necesitaba dentro del sistema y necesita consultar otro sistema o bien, simplemente salir del sistema y continuar o hacer alguna operación con la computadora o el navegador.

Precondiciones:	✓ El usuario se encuentra dentro del sistema y ya terminó de impartir su clase o terminó de desplegar la información que era de su interés. ✓ El usuario va a realizar otras actividades o consultar otro sistema.

	Actor		Sistema		
	Paso	Acción	Paso	Acción	Excepción
Flujos:	1	Da clic en la barra de direcciones del navegador y teclea otra dirección	2	El navegador despliega la página solicitada	E1
	1	Da clic en el botón de cerrar de la página del navegador.	2	El navegador se cierra completamente.	

	ID	Nombre	Acción
Excepciones:	E1	Página no disponible o fuera del sistema.	El navegador despliega una pantalla con el mensaje de página no encontrada.

Postcondiciones:	✓ El usuario ha salido del sistema. ✓ No se podrá ingresar nuevamente al sistema hasta que se vuelva a iniciar el Explorador de Internet y se teclee la dirección de la página del sistema. ✓ Para desplegar información sobre POO será necesario volver a ingresar al sistema. ✓ Se puede volver a utilizar el sistema desde cero.

4.2.5 Herramientas

La gran influencia de las tecnologías de la información y de la comunicación en la educación, ha cambiado completamente el panorama durante los últimos años. Las ventajas que tenía la educación clásica, en la que se contaba con profesores experimentados y los estudiantes podían tener un contacto cara a cara con ellos, poco a poco han sido rebasadas; de manera que, este modelo educativo ya no resulta tan apropiado en un mundo moderno que ha incorporado como sus pilares las tecnologías de la información y la comunicación. Este "mundo" proporciona nuevas posibilidades, las cuales permiten experimentar con nuevas formas de elaborar material para apoyar las clases e incluso, la forma en que este material es presentado a los estudiantes. Hoy en día, es común encontrar un número creciente de herramientas y entornos visuales como apoyo a la impartición de cursos, conferencias, etc., en muchos campos de la educación. La razón de estas nuevas formas de enseñanza es simple, con el apoyo de estas herramientas se reduce el costo del proceso de enseñanza – aprendizaje y por otro lado brinda la posibilidad de mostrar a los estudiantes procesos y mecanismos que usualmente estaban "ocultos", debido a que en situaciones normales podía sólo ser explicado de forma teórica. Entonces, surge una nueva cuestión, ¿qué plataforma o ambiente de desarrollo es más apropiado para cumplir con los nuevos paradigmas educativos? [42].

Cuando Tomas Alva Edison inventó la película en 1922, predijo que las películas podrían un día reemplazar los libros de texto. Sin embargo, los libros son aún considerados como material esencial en los cursos tradicionales. Durante la Segunda Guerra Mundial, las películas fueron utilizadas para poder entrenar a las personas de forma más rápida y menos costosa. Tiempo después, en las universidades abiertas en todo el mundo, comenzaron a utilizar los videos y la televisión como medio de comunicación que facilitaba el aprendizaje. A finales de los 70's y principios de los 80's, el entrenamiento basado en computadoras permitió añadir nuevas características, como: interactividad y entrenamiento automático. En los 90's, las tecnologías multimedia permitieron mejorar los

contenidos utilizados para entrenamiento. A mediados de los 90's, la popularidad de Internet y las tecnologías de la WWW marcaron el inicio de una nueva etapa en lo que a educación se refiere [25, 11].

Hoy en día la utilización de animaciones, como herramienta educativa, se ha vuelto muy popular; la gran mayoría del software y materiales de aprendizaje que acompañan a los libros de texto hoy en día, incluyen animaciones como material adjunto para apoyar la enseñanza. Actualmente es posible encontrar un sinnúmero de comunidades dedicadas a recabar los comentarios sobre la utilización de animaciones como una herramienta importante que acompaña y apoya la enseñanza basada en computadoras. Muchos profesores se encuentran ansiosos de incorporar estas tecnologías como apoyo para sus clases, sin embargo, numerosos estudios confirman que uno de los principales impedimentos para la total adopción de estas tecnologías en el sector educativo, es que se requiere mucho tiempo para aprenderlas, instalarlas y para desarrollar elementos visuales, lo cual impide que puedan ser completamente integradas en un curso formal a corto plazo. Los cinco principales impedimentos que los profesores han tenido que enfrentar al momento de que querer adoptar estas herramientas para apoyar su labor docente son [11]:

- El tiempo que se requiere para buscar e implementar buenos ejemplos.
- El tiempo que toma aprender una nueva herramienta.
- El tiempo que toma desarrollar las animaciones o elementos visuales.
- La falta de herramientas efectivas, que faciliten el desarrollo de las animaciones.
- El tiempo que toma el adaptar las animaciones a los contenidos de un curso.

Sin embargo, a pesar de las evidentes limitaciones que trae consigo la utilización de animaciones como herramienta para apoyar la enseñanza; al mismo tiempo, se han hecho algunas investigaciones con estudiantes para medir el impacto que tiene la utilización de estos recursos en el aprendizaje efectivo. Se ha trabajado

con estudiantes en dos niveles principales: impartición de cursos a estudiantes sin ninguna animación durante períodos cortos de tiempo en el salón de clases y cursos en los cuales, los estudiantes interactúan directamente con las animaciones y elementos visuales tanto dentro como fuera del salón de clases; los resultados de estos estudios revelan que la utilización de animaciones tiene un impacto benéfico en el proceso de aprendizaje de los estudiantes, se sienten más motivados y menos frustrados, ya que no tienen que enfrentarse sólo a explicaciones teóricas, sino que pueden ver en "acción" un concepto estudiado. Es importante resaltar, que aunque la utilización de animaciones puede ayudar al proceso de aprendizaje, es necesario que estas animaciones se acompañen de algoritmos o explicaciones consistentes pues de lo contrario, las animaciones por sí mismas podrían confundir a los estudiantes más que ayudarlos [11].

4.2.4.1 Flash y ActionScript

Flash es una potente herramienta creada por Macromedia que ha superado las expectativas de sus creadores y pertenece a la compañía Adobe (figura 36). Es una aplicación multimedia, que se utiliza para la generación de páginas Web animadas e interactivas, convirtiéndose en el estándar para la producción y visualización de animaciones en línea. Gracias a su potencia y versatilidad, es posible crear animaciones complejas e incluso es posible diseñar una página Web completa, sin embargo, hay que tener en cuenta que Flash no es una aplicación pensada para realizar páginas Web, sino para crear animaciones y conferir mayor interactividad a las páginas que se realizarán en editores HTML. Flash no edita ni genera código HTML directamente. Lo que genera es una película *Shockwave* (SWF) que se incorpora, posteriormente, en un documento HTML.

Por su parte ActionScript es el lenguaje de programación que utiliza Flash y que brinda el control absoluto de todo lo que rodea a una película. Es un lenguaje orientado a objetos, esto quiere decir que la información se organiza en clases. Cuando se desea emplear una clase en una película, se usa una instancia de ella, denominada OBJETO. Los objetos y las clases, tienen *propiedades* y *métodos*. Fue lanzado con la versión 4 de Flash, y desde entonces ha ido ampliándose poco a

poco, hasta llegar a niveles de dinamismo y versatilidad muy altos en la versión 9 (Adobe Flash CS3) de Flash. ActionScript es un lenguaje de script, esto es, no requiere la creación de un programa completo para que la aplicación alcance los objetivos.

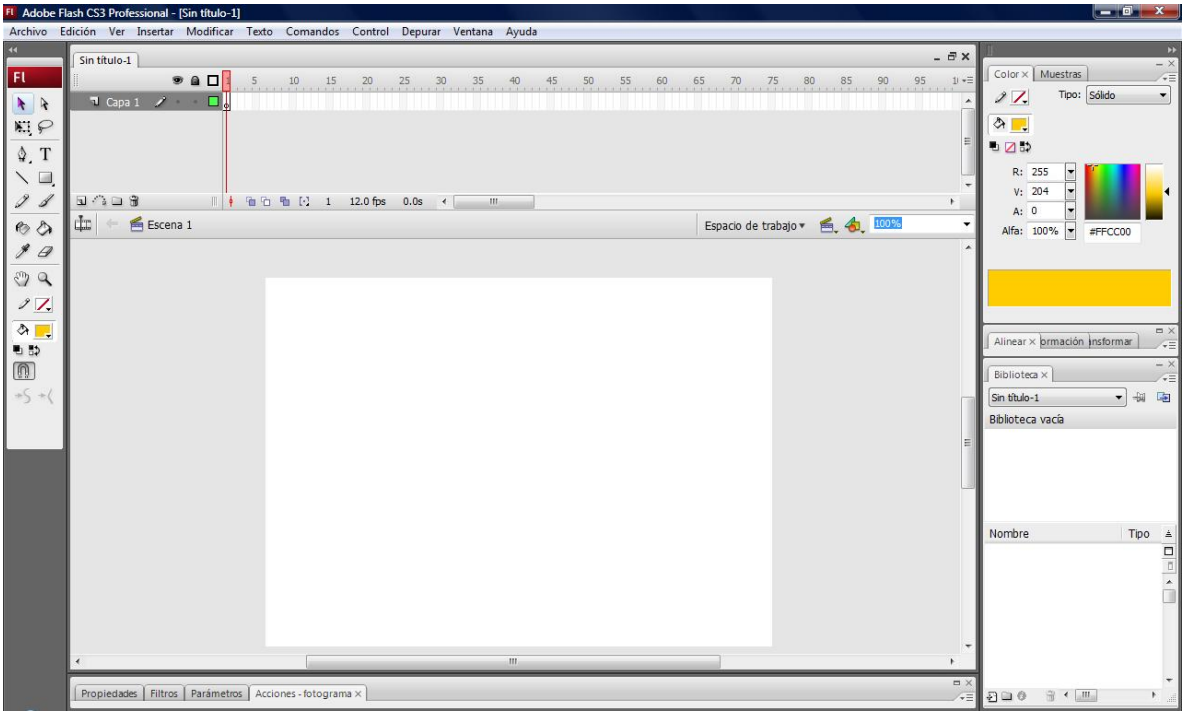


Figura 36. El entorno de Adobe Flash ©

Existen muchas razones que avalan la elección de Flash como una herramienta adecuada para el desarrollo de animaciones, quizá la principal razón es que se trata de un entorno divertido y relativamente simple de utilizar, brindando la capacidad de desarrollar interesantes aplicaciones en conjunto con ActionScript. Se trata de una herramienta intuitiva, lo cual permite a los usuarios aprender cómo funciona sin tener la necesidad de invertir mucho tiempo en el entrenamiento [24].

Las principales ventajas que proporciona Flash son: tamaño relativamente pequeño de los archivos que contienen los diseños de animaciones, un alto grado de interactividad, la facilidad de manejar tanto mapas de bits como imágenes vectoriales, disponibilidad, accesibilidad y facilidad de uso; por otro lado, proporciona interactividad a una animación, permite desarrollar navegaciones a través de botones, temporizadores y menús interactivos que pueden ser añadidos a

cualquier animación. Fragmentos de códigos pueden ser adjuntados a los objetos, botones, clips de película o fotogramas. De acuerdo con reportes publicados por Adobe, Flash es la más popular plataforma para el desarrollo de animaciones en todo el mundo. Adicionalmente, existe una gran cantidad de recursos en la Web que pueden ayudar a desarrollar una animación en Flash. Permite la fácil y rápida incorporación de audio y efectos de video. [11, 42].

4.2.4.2 Las alternativas a Flash

- ✓ **DHTML:** Se creó como solución a la falta de dinamismo del HTML y significa HTML Dinámico. Permite crear efectos gráficos sencillos, útiles para botones y barras de navegación. En realidad es JavaScript mezclado con HTML. Si el usuario tiene desactivado el Javascript en su navegador por razones de seguridad, no podrá ver nada de lo hecho usando DHTML.

- ✓ **JAVASCRIPT**: Es un lenguaje de programación similar al C pero que se ejecuta en el servidor. De este modo no accede al disco duro salvo que el programa lo requiera. Puede crear efectos interesantes tanto con texto como con gráficos pero no es tan potente.

- ✓ **CSS:** (de *Cascade Style Sheet*), también llamadas Hojas de Estilo en Cascada. Su función es complementar la carencia en la riqueza de las tipografías en el lenguaje HTML. No son una forma de animar la Web, sino son un complemento de ella.

Todos estos lenguajes de programación complementan de algún modo al HTML, pero solo Flash puede crear contenido Web dinámico por sí mismo. Se puede considerar que Flash engloba a las CSS y es perfectamente compatible con Javascript.

CAPÍTULO V.

ANÁLISIS Y DISEÑO DEL SISTEMA

EL ANÁLISIS EN EL PROCESO UNIFICADO TIENE COMO OBJETIVO ANALIZAR LOS REQUERIMIENTOS PARA LOGRAR UN MEJOR ENTENDIMIENTO DE LO QUE SE DESEA REALIZAR; SE CONSTRUYE UN MODELO DEL ANÁLISIS QUE SIRVA COMO BASE PARA ESTRUCTURAR TODO EL SISTEMA. LOS PRODUCTOS QUE SE OBTENDRÁN EN ESTA ETAPA SERÁN: LA DEFINICIÓN DE LA ARQUITECTURA CON LOS PAQUETES DEL ANÁLISIS, LOS DIAGRAMAS DE CLASES, DIAGRAMAS DE SECUENCIA Y EL DIAGRAMA DE ESTADOS PARA LA NAVEGACIÓN DE LA INTERFAZ. LA VENTAJA DEL ANÁLISIS ES QUE ES POSIBLE CONTRUIR UN MODELO OBUSTO Y ENTENDIBLE DEL SISTEMA, YA QUE NO INCLUYE ASPECTOS TECNOLÓGICOS.

POR SU PARTE EL DISEÑO TIENE EL OBJETIVO DE CREAR UN PUNTO DE PARTIDA PARA LAS ACTIVIDADES DE IMPLEMENTACIÓN. SE DESCOMPONE EL SISTEMA EN SUBSISTEMAS MÁS MANEJABLES Y SE DEFINEN CON MAYOR DETALLE LOS MODELOS DEL ANÁLISIS, INCLUYENDO CONCEPTOS DEL AMBIENTE DE IMPLEMENTACIÓN. LOS PRODUCTOS QUE SE OBTENDRÁN EN ESTA ETAPA SERÁN: DESCRIPCIÓN DE LA ARQUITECTURA, EL DISEÑO DE LAS CLASES, REALIZACIÓN DE LOS CASOS DE USO, DISEÑO DE COMPONENTES Y SUBSISTEMAS, ASÍ COMO EL DIAGRAMA DE DISTRIBUCIÓN.

.

V. Análisis y diseño del sistema

Análisis

El **Análisis en el Proceso Unificado**, tiene como objetivo analizar los requerimientos para tener un mejor entendimiento de lo que se pretende y de esta manera construir el modelo de análisis que sirva de base para estructurar todo el sistema de software que apoyará la enseñanza de la programación orientada a objetos con Java.

Los productos que se tendrán en esta etapa son los siguientes: *Diagramas de Clases, Diagramas de Secuencia* y *Diagrama de Estados para la navegación de la interfaz*. Se analizaran los casos de uso para encontrar las clases necesarias para cada uno de los diagramas, por otro lado, se definirán las clases y sus responsabilidades, así como los paquetes de análisis.

Lo que se busca es que se realicen bien las actividades para que de esta manera se pueda construir un modelo más robusto y entendible del sistema, pues por el momento no incluye aspectos tecnológicos.

En este proyecto se define una **arquitectura multicapas**:

- ✓ **Capa de presentación o de interfaz humana.** Considera todo lo relacionado con la vista externa del sistema.
- ✓ **Capa del negocio.** Que contiene todas las clases del dominio del problema.
- ✓ **Capa de almacenamiento.** Incluye la información persistente en el sistema.

5.1 Diagrama de clases del análisis

Para la definición de los diagramas de clases, es necesario que se analice cada caso de uso, para identificar las clases del análisis en las tres capas y así poder definir sus responsabilidades. Existen tres tipos de clases:

- ✓ **Interfaz.** Representa el límite entre el sistema y el ambiente
- ✓ **Control.** Representa la lógica de la aplicación.
- ✓ **Entidad.** Representa a la información almacenada en el sistema

5.1.1 Diagrama de Clases de Interfaz

En las clases de interfaz, se coloca una clase para cada: **actor, sistema externo** o **dispositivo que interacciona con el sistema**. Una clase de interfaz puede modelar un conjunto de <<pantallas>> o <<formas de entrada>> y tienen el estereotipo <<interface>>. Se define una clase por cada pantalla que se identificó en el prototipo; además, se puede poner una pantalla para cada funcionalidad importante del sistema.

Para dale mayor claridad al diagrama, éste se ha dividido en los tres paquetes que lo conforman:

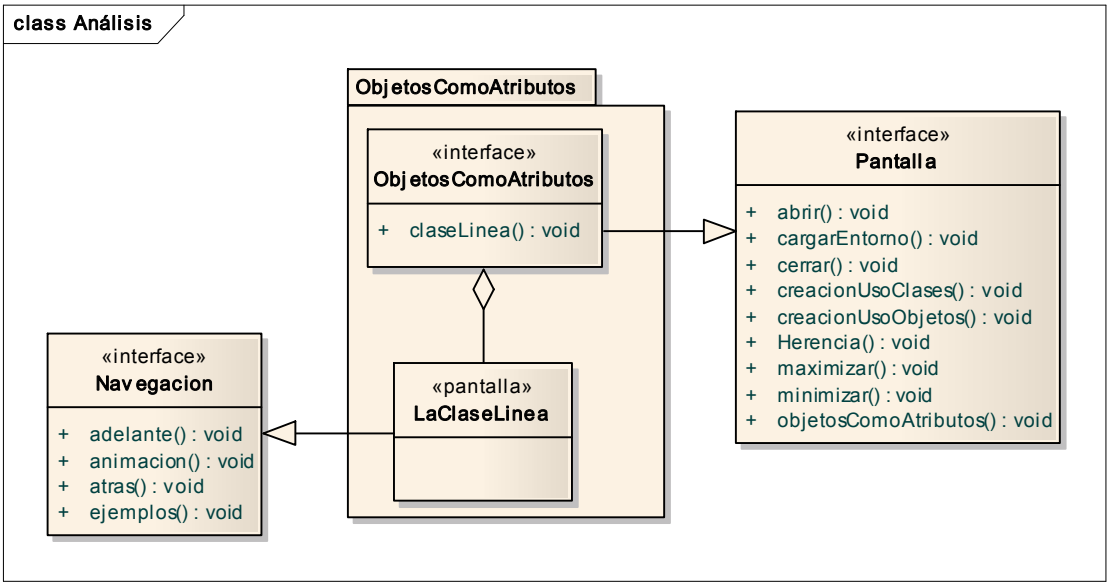


Figura 37. Clase de interfaz *Objetos como atributos*

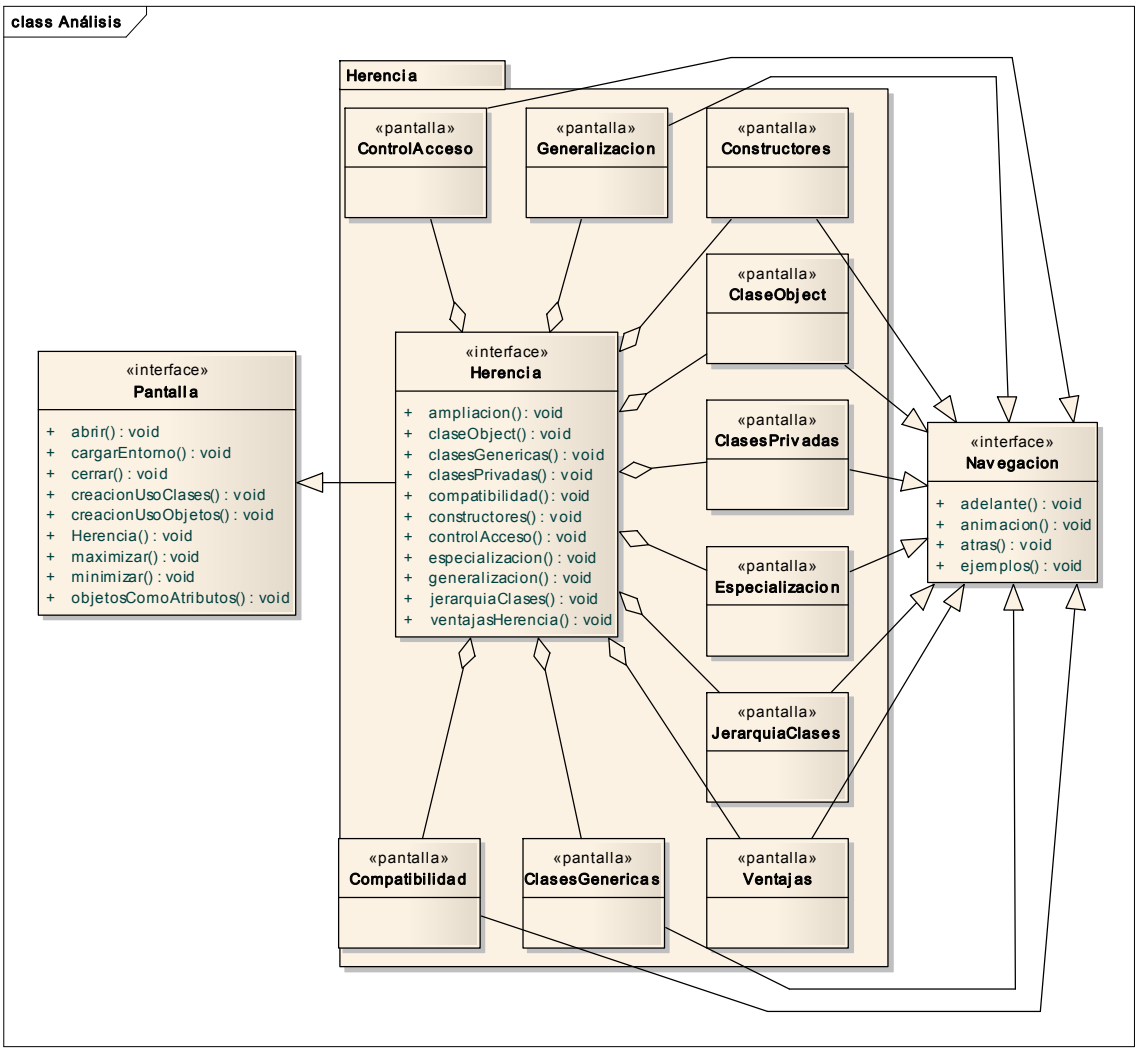


Figura 38. Clase de interfaz *Herencia*

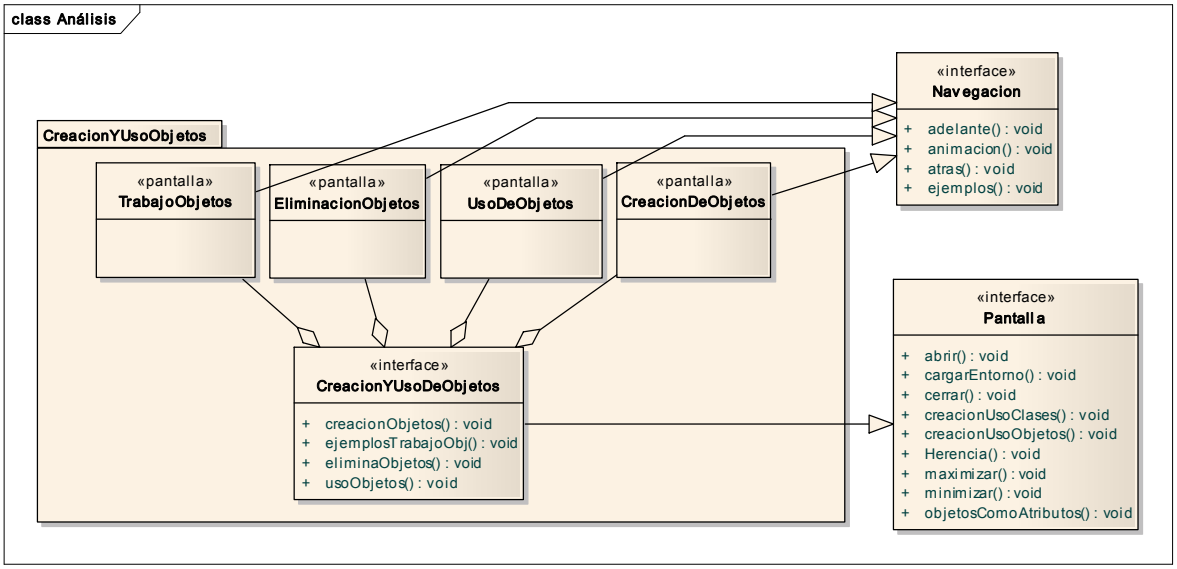


Figura 39. Clase de interfaz *Creación y uso de objetos*

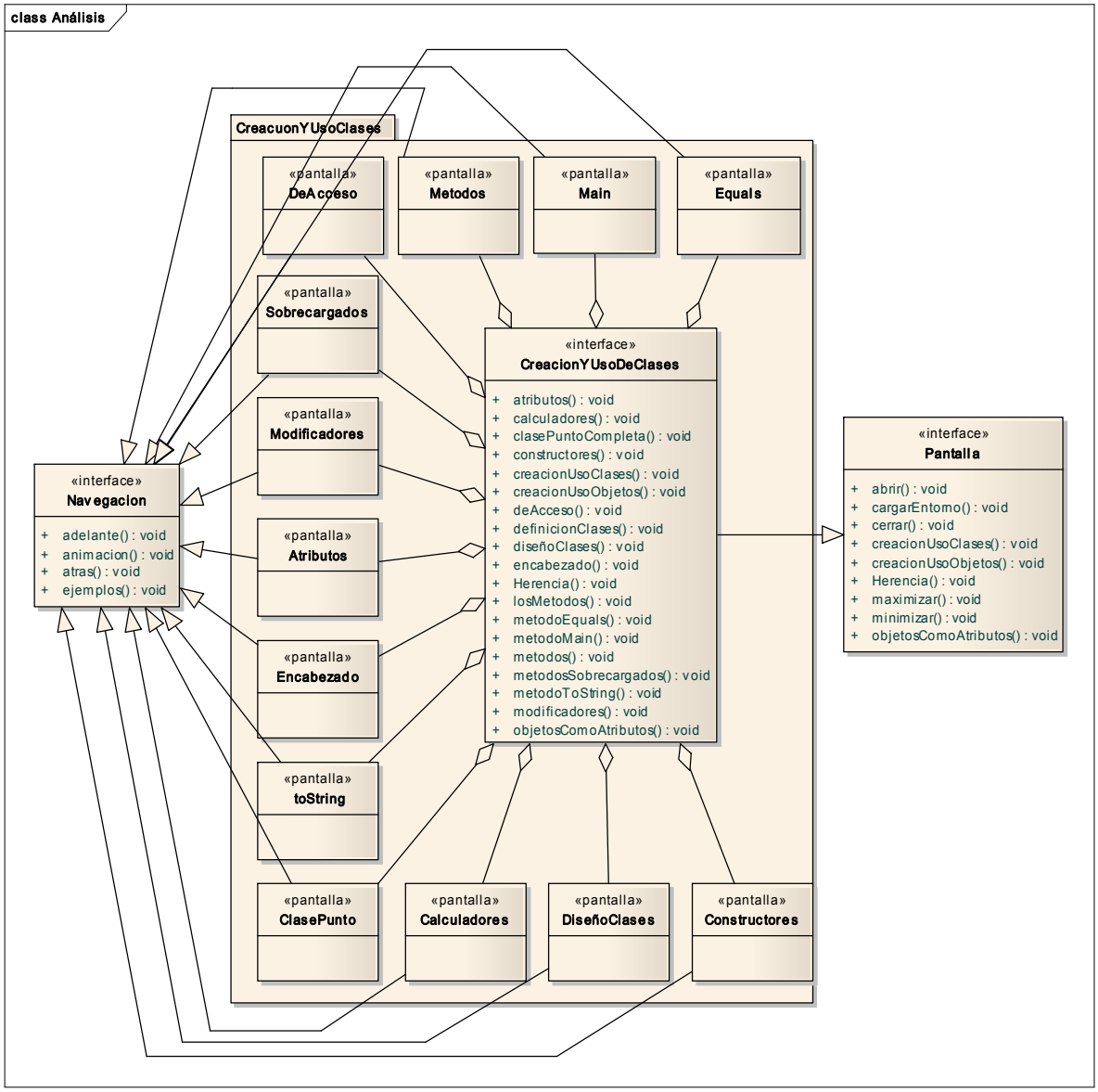


Figura 40. Clase de interfaz *Creación y uso de clases*

5.1.2 Diagrama de Clases de Control

Las clases de control encapsulan el comportamiento y manejan el control del flujo del sistema. Existen diferentes heurísticas para determinarlas. Coad [49] propone estereotipos en aplicaciones transaccionales:

- Participantes
- Transacciones
- Lugares
- Cosas
- La siguiente transacción

Para el caso del sistema de software que apoyará la enseñanza de la POO, se tiene:

Participantes	Profesores	
Transacciones	Entrar al sistema	
	Salir del sistema	
	1. Creación y uso de objetos	1.1 Creación de objetos 1.2 Uso de objetos 1.3 Eliminación de objetos 1.4 Ejemplos de trabajo con objetos
	2. Creación y uso de clases	2.1 Diseño de clases 2.2 Definición de clases 2.3 Los métodos 2.4 El método main 2.5 El método equals 2.6 El método toString 2.7 La clase Punto completa 2.8 Métodos sobrecargados
	3. Objetos como atributos	3.1 La clase Línea completa
	4. Herencia	4.1 Ampliación mediante herencia 4.2 Control de acceso 4.3 Constructores 4.4 Uso de clases derivadas 4.5 Especialización 4.6 Jerarquía de clases 4.7 Ventajas de la herencia 4.8 Compatibilidad 4.9 Generalización 4.10 La clase Object 4.11 Clases genéricas.
Lugar	Internet o bien de manera local en una computadora.	
Cosas	Notas, animaciones, programas escritos en Java	

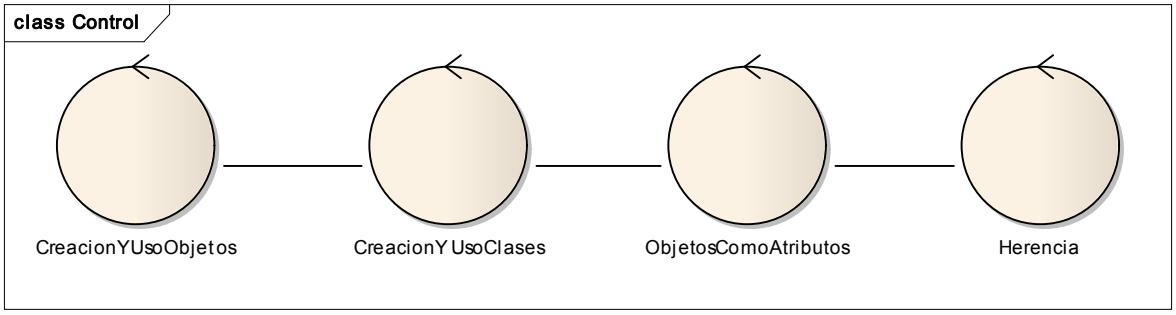


Figura 41. Diagrama de clases de control

5.1.3 Diagrama de Clases de Entidad

Las clases que componen los diagramas de clases de entidad, representan una clase de control que se desee sea persistente. En sentido estricto, el sistema no incorpora ninguna base de datos para el almacenamiento de esta información; sin embargo se contará con un banco de diapositivas, las cuales contienen la información que se desplegará a lo largo de toda la aplicación, las cuales residirán físicamente en el servidor en donde se encuentre montada la aplicación.

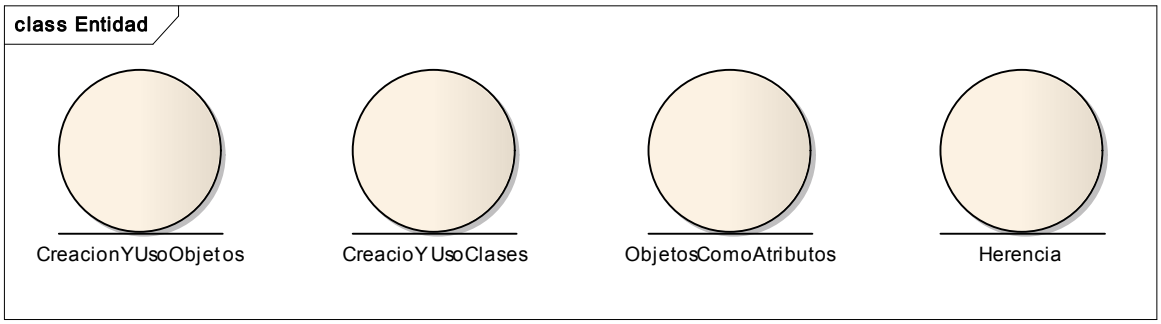


Figura 42. Diagrama de clases de entidad

5.2 Diagramas de Secuencia

Una vez que se han identificado las clases, se construye un diagrama de secuencia para cada **Caso de Uso**, mostrando las interacciones de las clases. Las iteraciones de clases, modelan sus responsabilidades; además de destacar la ordenación temporal de los mensajes, se muestran los flujos normales y excepcionales.

Caso de uso: **1. Entrar al sistema**

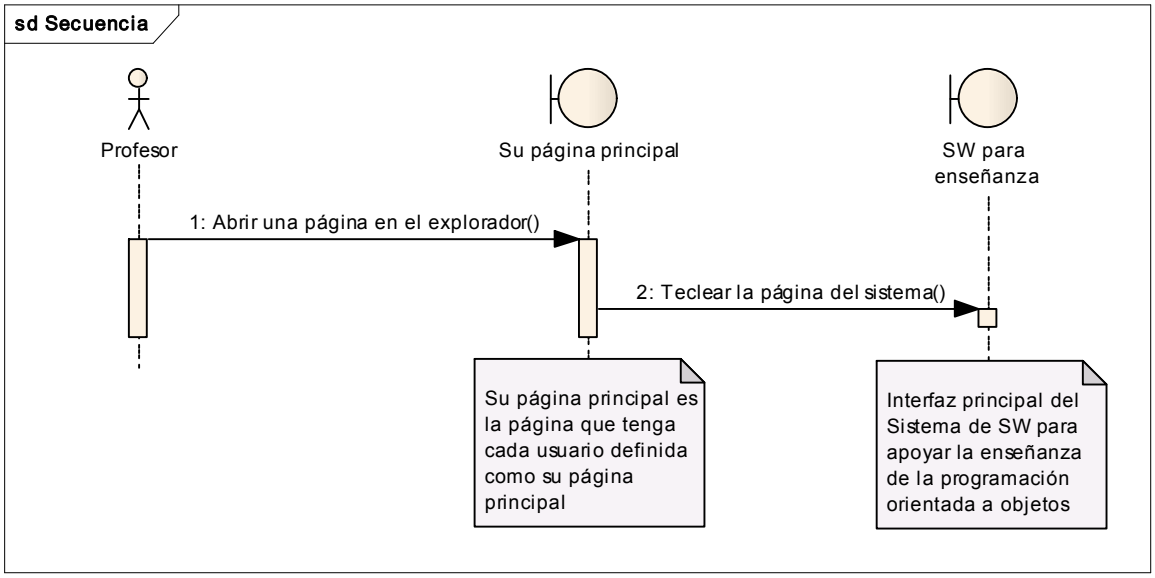


Figura 43. Diagrama de secuencia *Entrar al sistema*

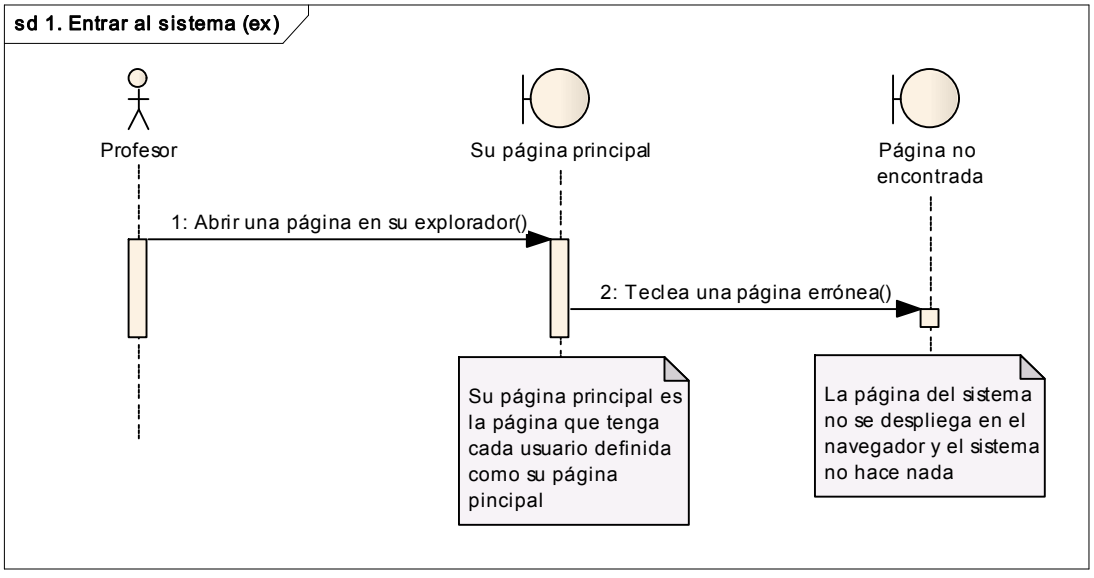


Figura 44. Diagrama de secuencia excepcional *Entrar al sistema*

Caso de uso: 2. Desplegar información de POO

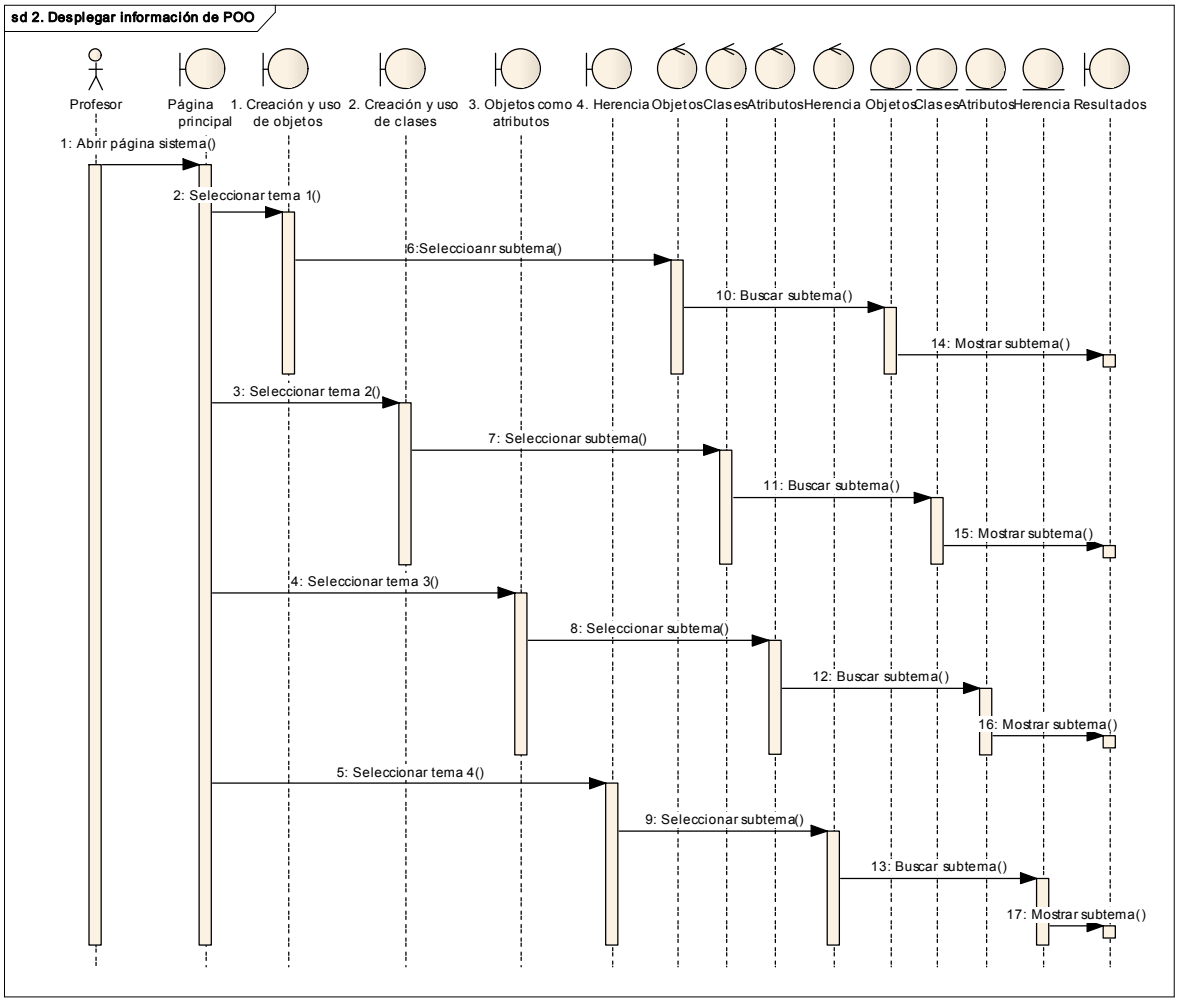


Figura 45. Diagrama de secuencia *Desplegar información de POO*

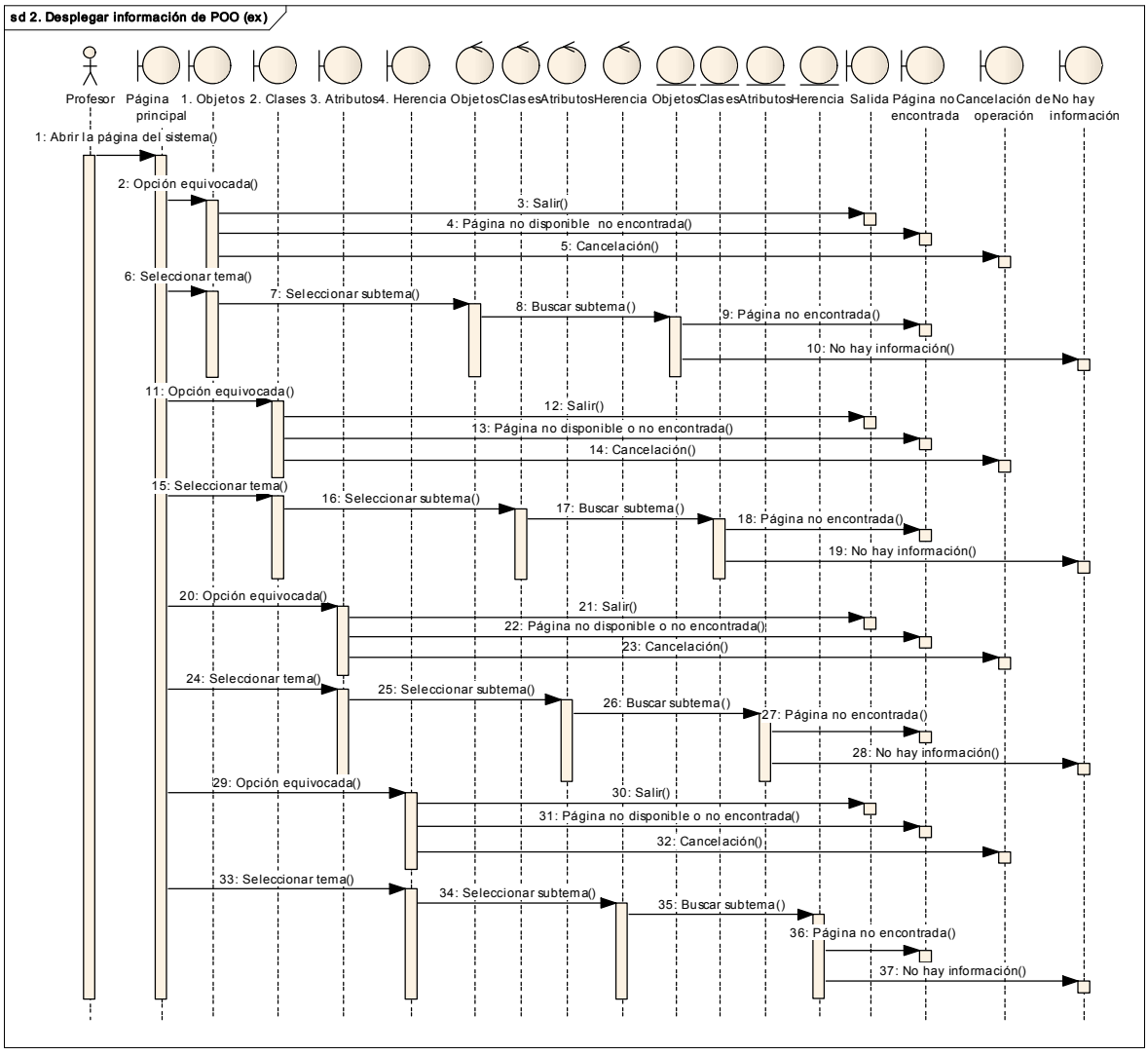


Figura 46. Diagrama de secuencia excepcional *Desplegar información de POO*

Caso de uso: 3. Ejecutar animaciones

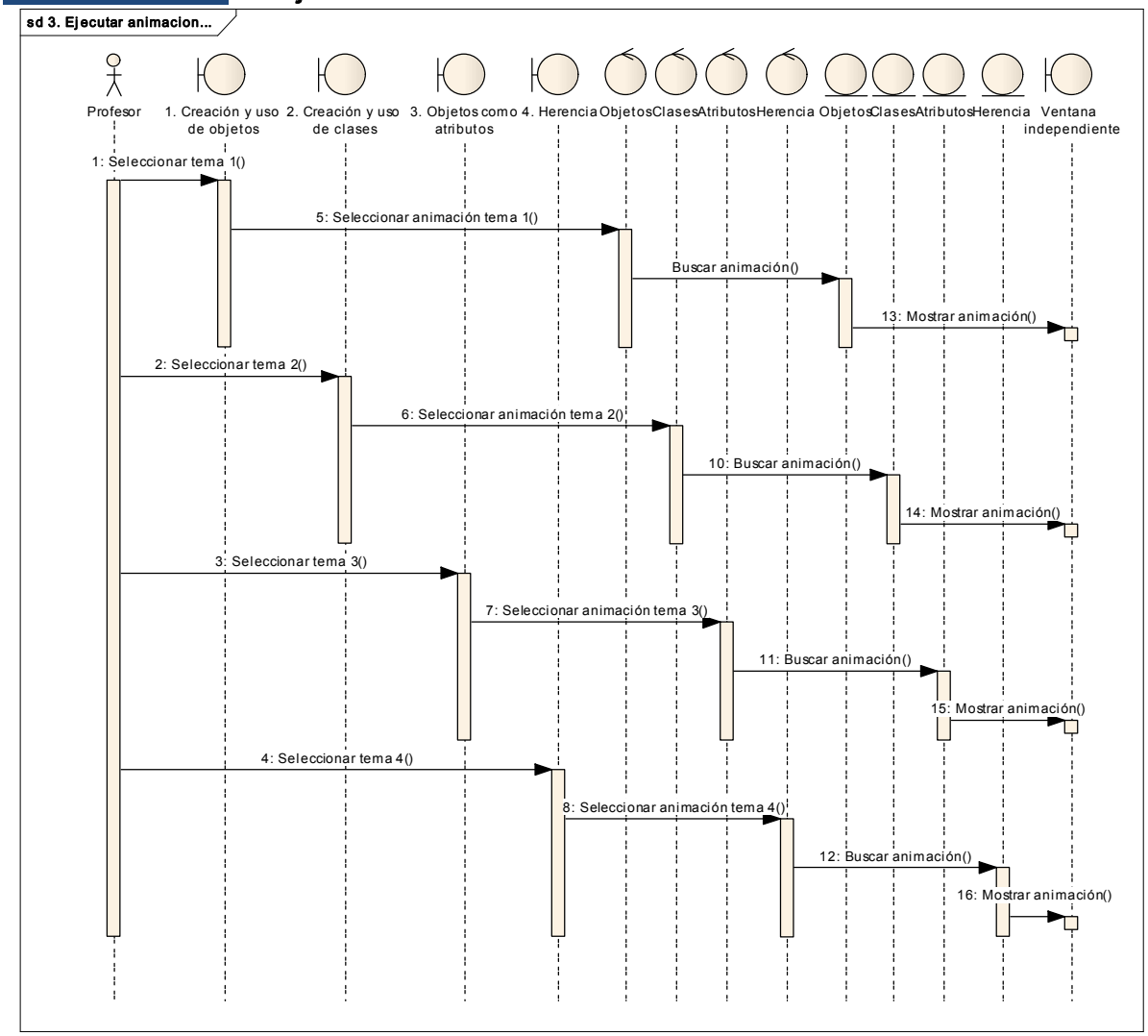


Figura 47. Diagrama de secuencia *Ejecutar animaciones*

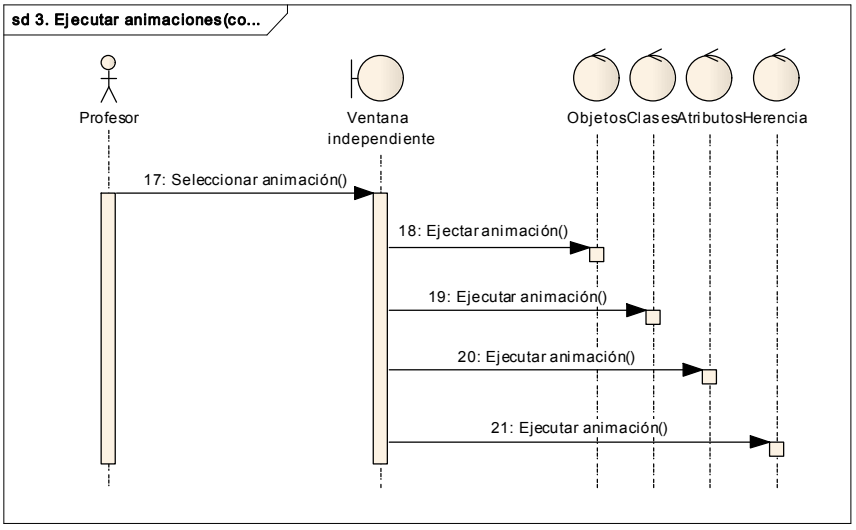


Figura 48. Diagrama de secuencia *Ejecutar animaciones* (continuación)

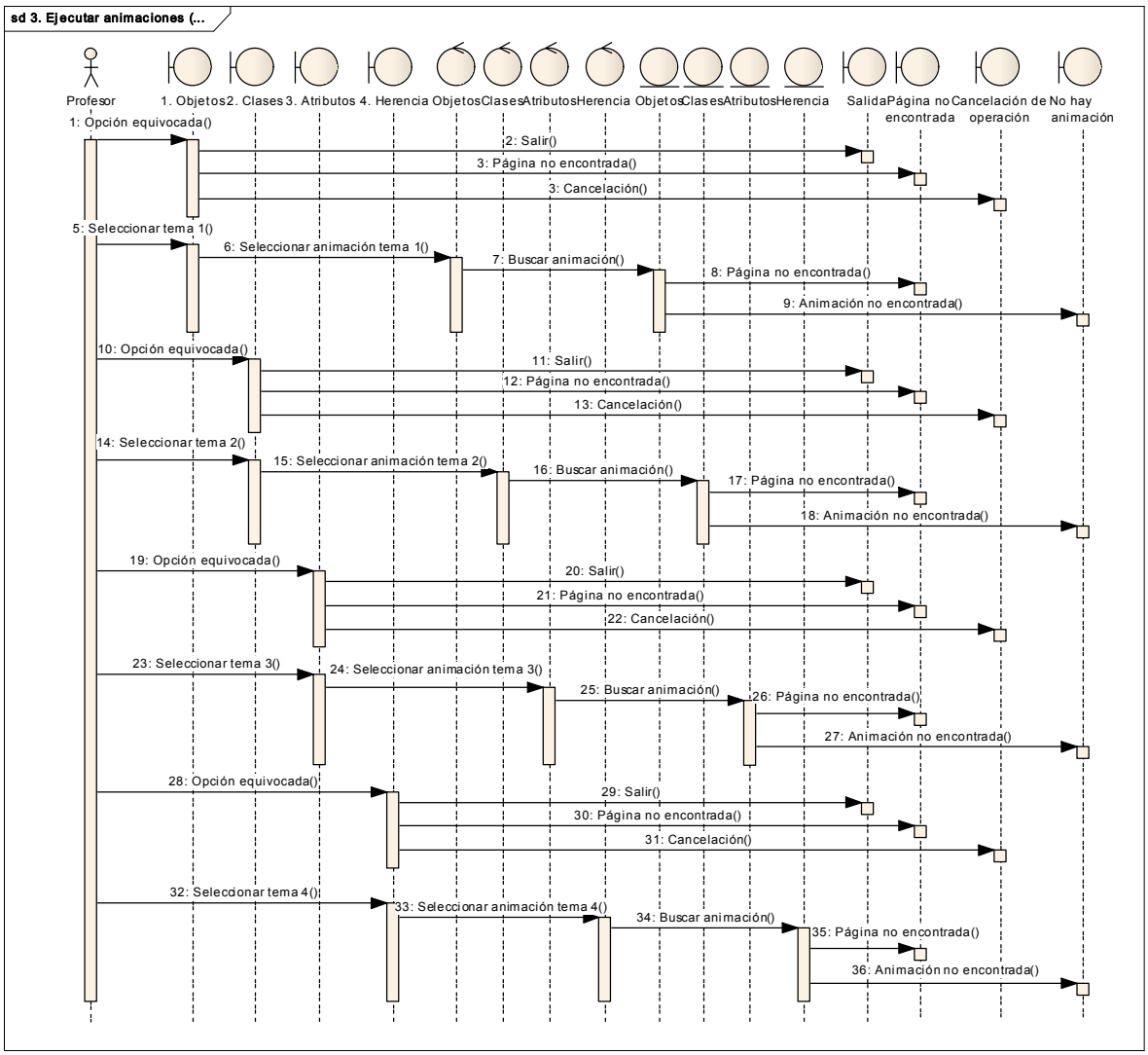


Figura 49. Diagrama de secuencia excepcional *Ejecutar animaciones*

Caso de uso: 4. Cargar entorno para programación

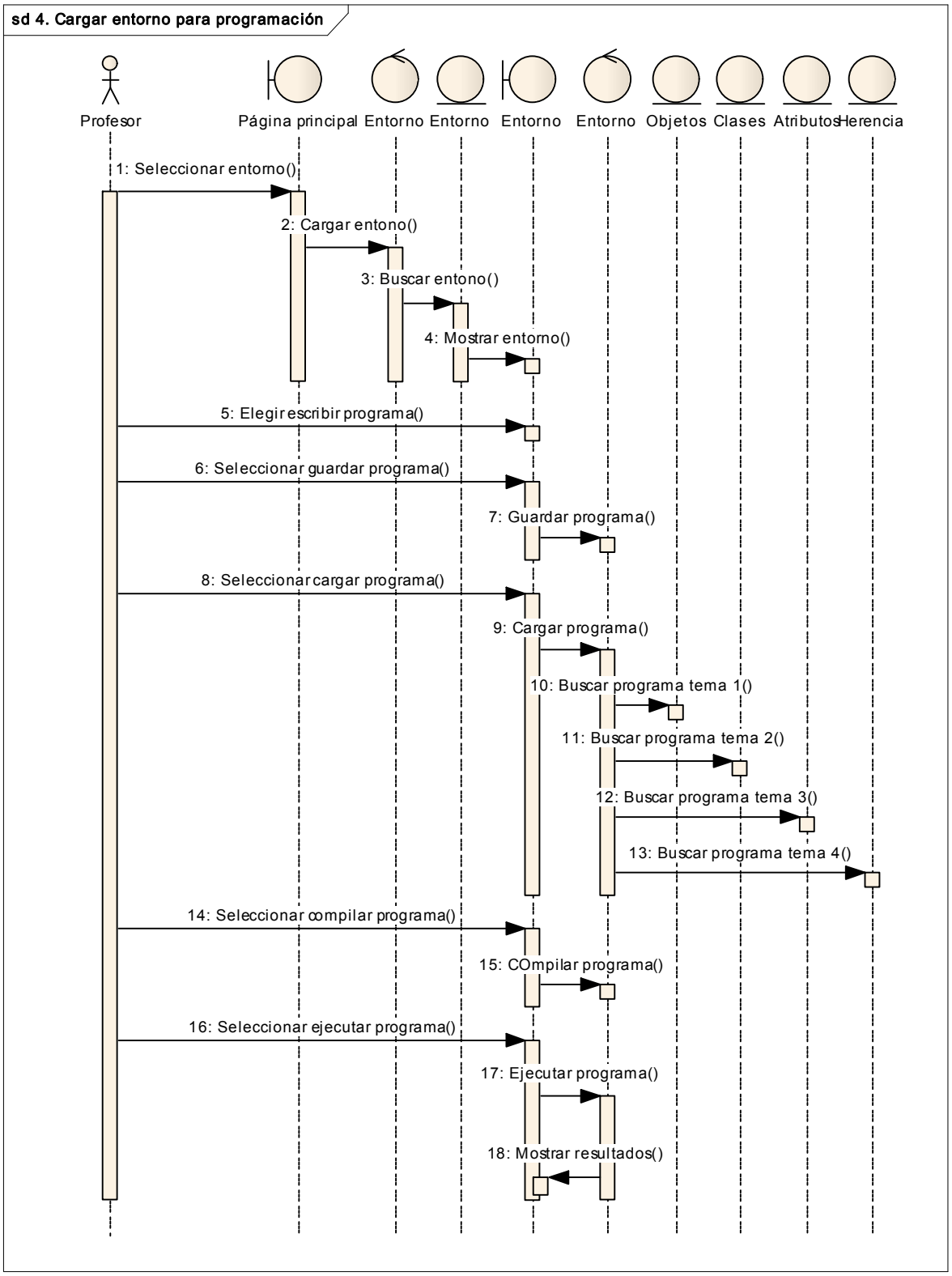


Figura 50. Diagrama de secuencia *Cargar entorno para programación*

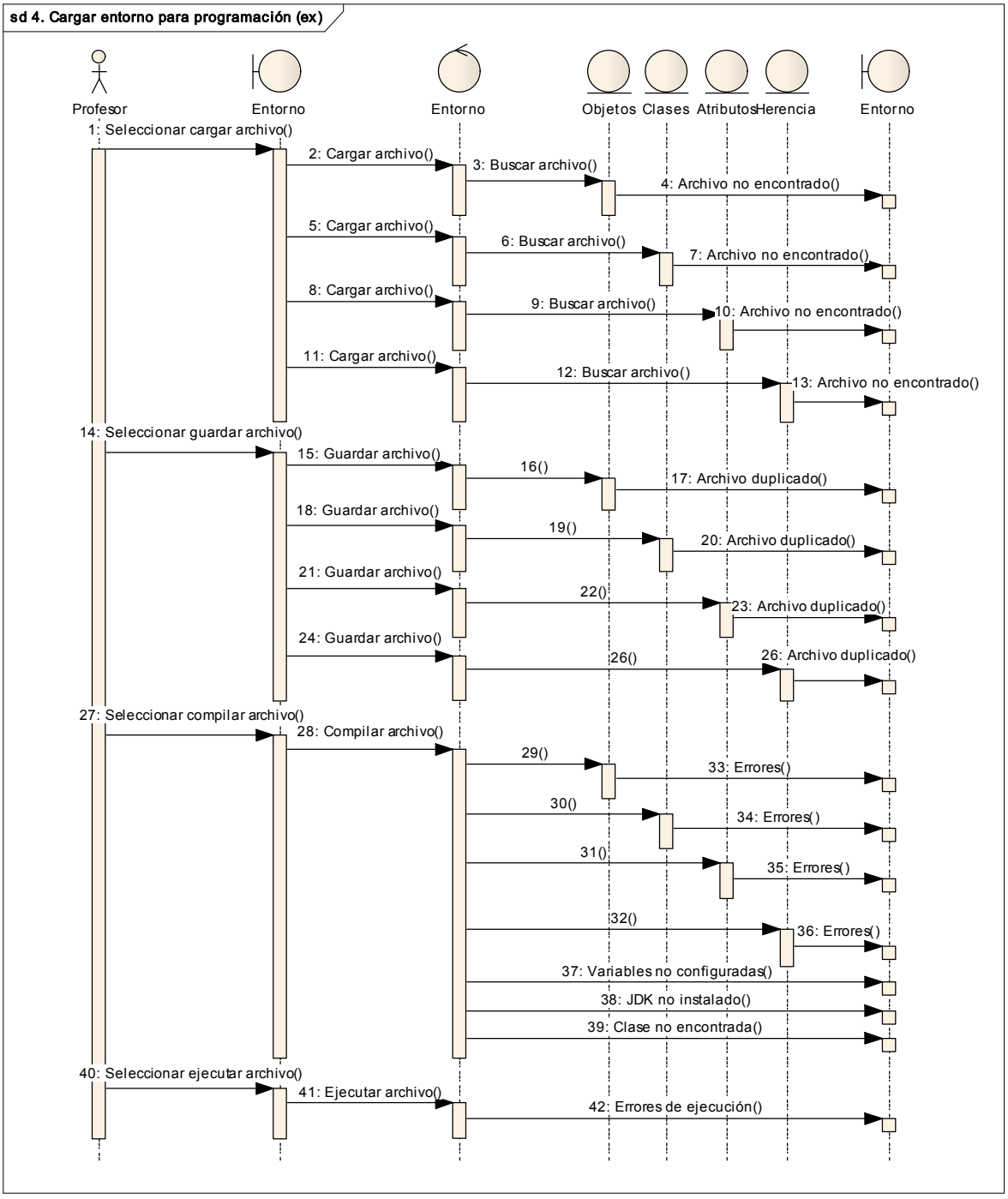


Figura 51. Diagrama de secuencia excepcional *Cargar entorno para programación*

Caso de uso: 5. Descargar biblioteca de códigos Java

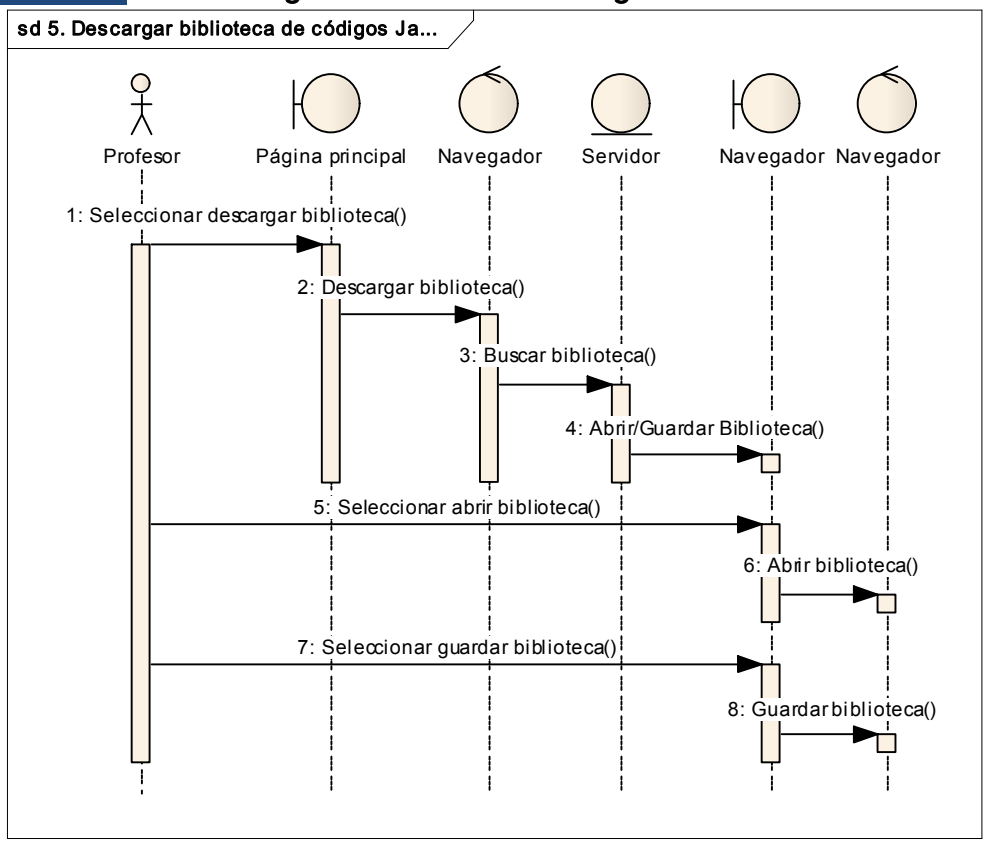


Figura 52. Diagrama de secuencia *Descargar biblioteca de códigos Java*

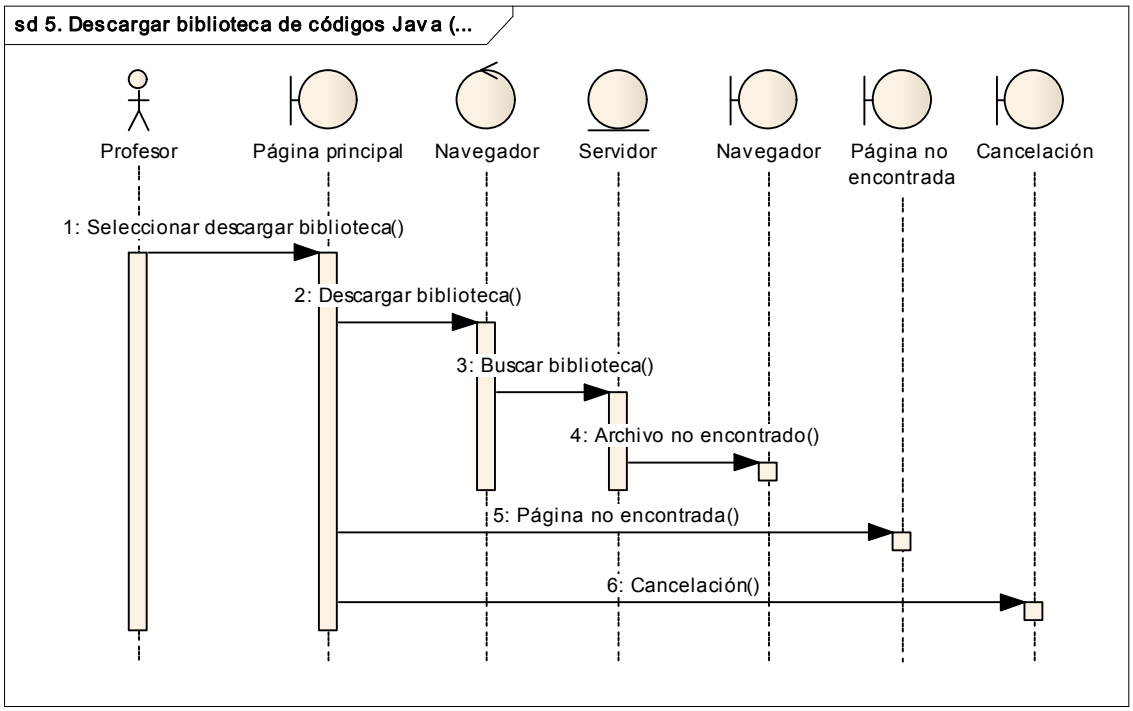


Figura 53. Diagrama de secuencia excepcional *Descargar biblioteca de códigos Java*

Caso de uso: 6. Salir del sistema

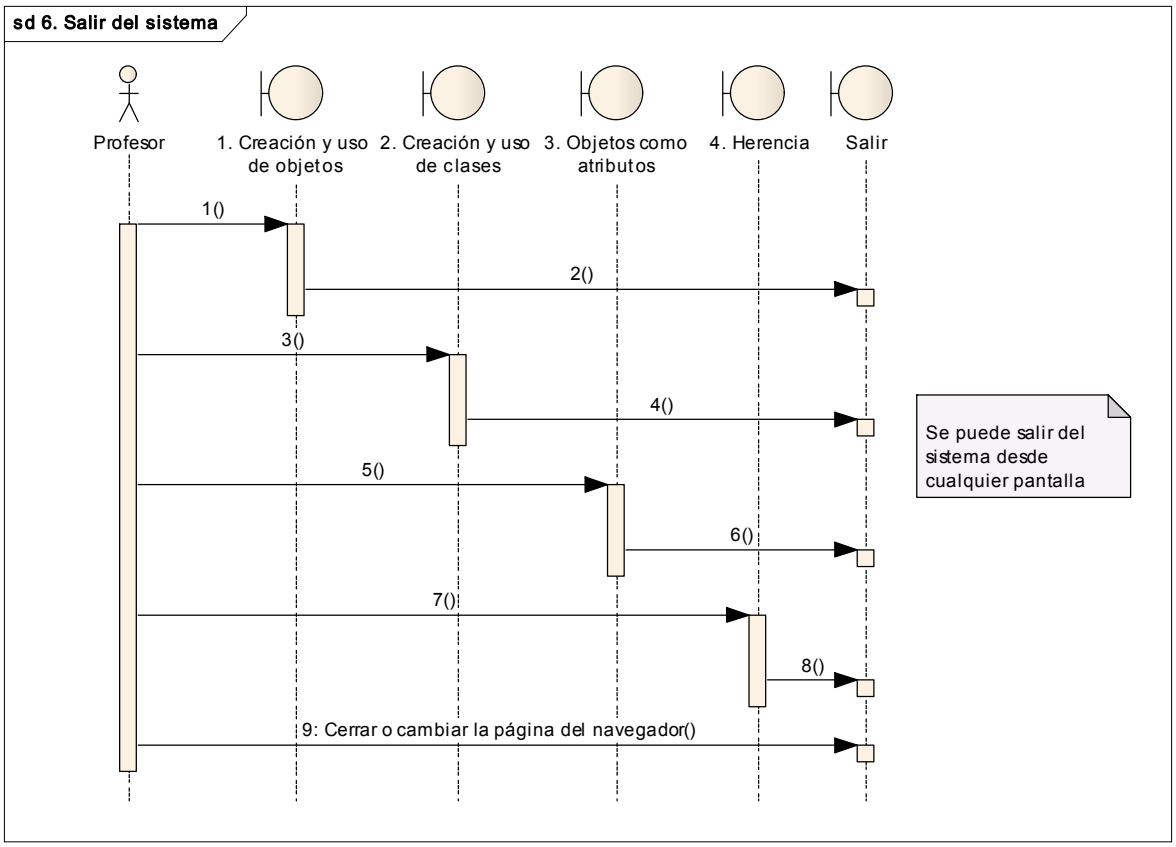


Figura 54. Diagrama de secuencia *Salir del sistema*

5.3 Diagramas de Estado

Los diagramas de estado muestran la manera en cómo se lleva a cabo la navegación a través de las pantallas del sistema. Se parte de un estado (pantalla) y según el evento o acción del usuario, se pasa al siguiente estado. Se debe de generar un diagrama de estado por cada uno de los actores del sistema.

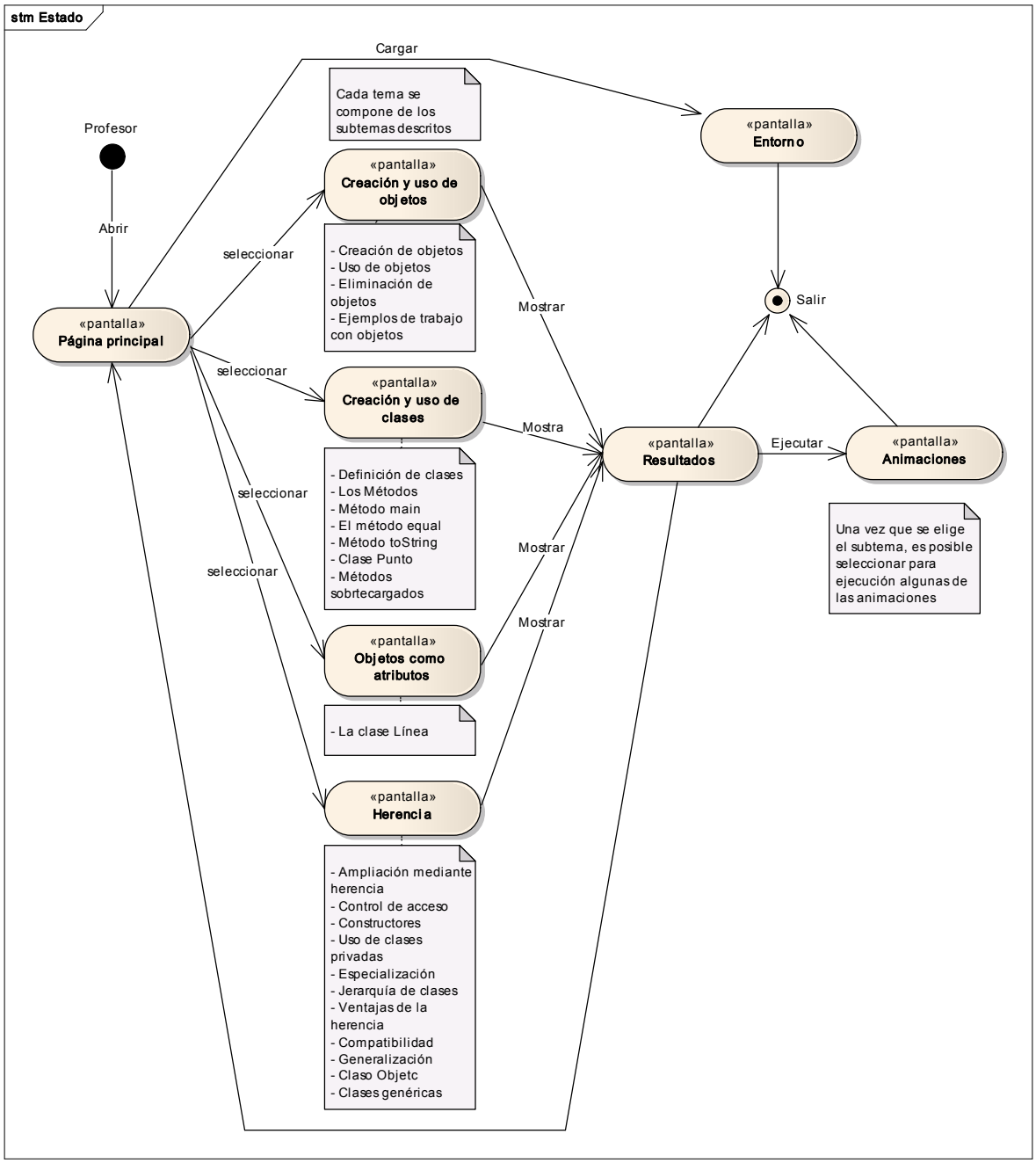


Figura 55. Diagrama de secuencia *Salir del sistema*

Diseño

El objetivo del diseño en el Proceso Unificado es crear un punto de partida para las actividades de implementación, para que de esta manera, se pueda definir la descomposición del sistema en piezas manejables o subsistemas. Lo anterior nos permitirá definir con mayor detalle los modelos del análisis incluyendo conceptos del ambiente de implementación.

Los productos que se generarán en esta parte son: *descripción de la arquitectura, diagrama de paquetes y diagramas de distribución.*

Las actividades que permiten llevar a cabo todo lo anterior son, definir la arquitectura, nodos y subsistemas; lo cual permite construir el diagrama de distribución. Además es necesario que se diseñen los casos de uso, modelando el aspecto dinámico de cada uno de ellos. El diseño de cada clase debe cubrir los detalles de la implementación y finalmente diseñar los subsistemas y componentes.

5.4 Descripción de la arquitectura del diseño

La arquitectura del diseño es el conjunto de decisiones importantes sobre la organización del software. Consiste en dividir el software en partes y las relaciones entre ellas. La arquitectura de software implica la toma de un conjunto de decisiones importantes sobre la organización del software, como pueden ser:

- ✓ Seleccionar los elementos estructurales y sus relaciones.
- ✓ Especificar las colaboraciones de comportamientos de esos elementos.

Se considera que una arquitectura es buena si es confiable, eficiente, entendible, sí separa los requerimientos funcionales de los no funcionales y sí separa lo específico de la plataforma.

Cuando se va a seleccionar un estilo arquitectónico, es necesario considerar algunos aspectos importantes:

- ✓ La facilidad de mantenimiento
- ✓ El desempeño y la escalabilidad
- ✓ La confiabilidad esperada
- ✓ Seguridad esperada
- ✓ Almacenamiento de los datos
- ✓ La distribución del software en los nodos

Al momento de definir la arquitectura es necesario hacerlo de acuerdo a las características del sistema que se está desarrollando y principalmente de debe de tomar en cuenta lo siguiente:

- ✓ Seleccionar algún patrón de arquitectura
- ✓ Seleccionar algún patrón para aplicación web
- ✓ Definir los nodos y su configuración
- ✓ Definir los subsistemas y sus interfaces
- ✓ Definir las clases importantes para la arquitectura
- ✓ Reutilización

La descripción de la arquitectura incluye:

- ✓ Objetivos y restricciones de la arquitectura. Lista de las decisiones importantes.
- ✓ Vistas del modelo de diseño. Describe los modelos usados para las vistas. Se compone de las siguientes vistas:
  - ➢ Vista de la distribución. Se identifican los nodos y los mecanismos de comunicación entre ellos, sus lenguajes de implementación y tecnologías. Para los nodos se definen las siguientes características:
    - Capacidades
    - Tipos de conexión
    - Protocolos
    - Procesos de migración
    - Respaldo de los datos
    - Redundancia

- Vista de los procesos. Muestra la distribución de los procesos en los nodos (elementos activos y sus hilos ejecutados en los nodos).
- Diseño de los casos de uso. Clases del diseño en capas, subsistemas y paquetes.

Para el desarrollo del Sistema de software para la utilización de pizarrones electrónicos para apoyar la enseñanza de la programación orientada a objetos con Java, se utilizará el Patrón de Capas, se trata de un enfoque exitoso para desarrollar software pues permite dividir las responsabilidades en capas. Una capa es un nivel de abstracción que agrupa un conjunto de tareas.

Patrón de Capas

El patón de Capas se encuentra estructurado de la siguiente forma:

a) Capa de Presentación

Contiene los elementos que residen en el cliente y en el servidor. El cliente presenta la interfaz de usuario y sus interacciones. En el servidor se procesan las peticiones del cliente y se les da respuesta.

b) Capa del Negocio

Es la responsable del dominio de problema y proporciona los servicios del negocio. Sus componentes están en el servidor y pueden ser EJB o aplicaciones de Java.

c) Capa de Integración

Proporciona acceso a los recursos de Servicios de Información de los Negocios (EIS) como bases de datos, servicios de correo, de directorios, colas de mensaje, etc.

La Arquitectura de Capas usualmente se inicia planteando una arquitectura de capas para particionar el modelo. Una capa agrupa elementos de software que tiene el mismo nivel de abstracción. Las capas inferiores tienen elementos más abstractos y reusables y los superiores elementos más concretos para la aplicación:

- ✓ **Capa de Interfaz:** Elementos para la interacción con el usuario
- ✓ **Capa de aplicación:** Contiene los elementos que llevan el flujo de control de los actores principales del sistema. Las clases que apoyan lo casos de uso se componen en esta capa. Se organizan en paquetes (se incluyen clases para los actores principales, uno o más casos de uso, para algún área funcional). Se pueden subdividir en:
  - ➢ **Capa de aplicación específica.** Las clases de uso pares se ponen en la capa de aplicación específica.
  - ➢ **Capa de aplicación genérica.** Las clases para los casos de uso a incluir se ponen en la capa de aplicación genérica.
- ✓ **Capa de dominio:** Contiene los elementos de conceptos fundamentales del domino. Contiene la información a ser mantenida, revisada o manipulada en el sistema. Estos elementos se comparten por los casos de uso y pueden ser específicos o genéricos.
- ✓ **Capa de servicio o infraestructura:** Esta capa puede ser independiente o dependiente de la plataforma.

Las capas se representan por medio de paquetes que a su vez pueden contener otros paquetes.

5.5 Diagrama de Paquetes

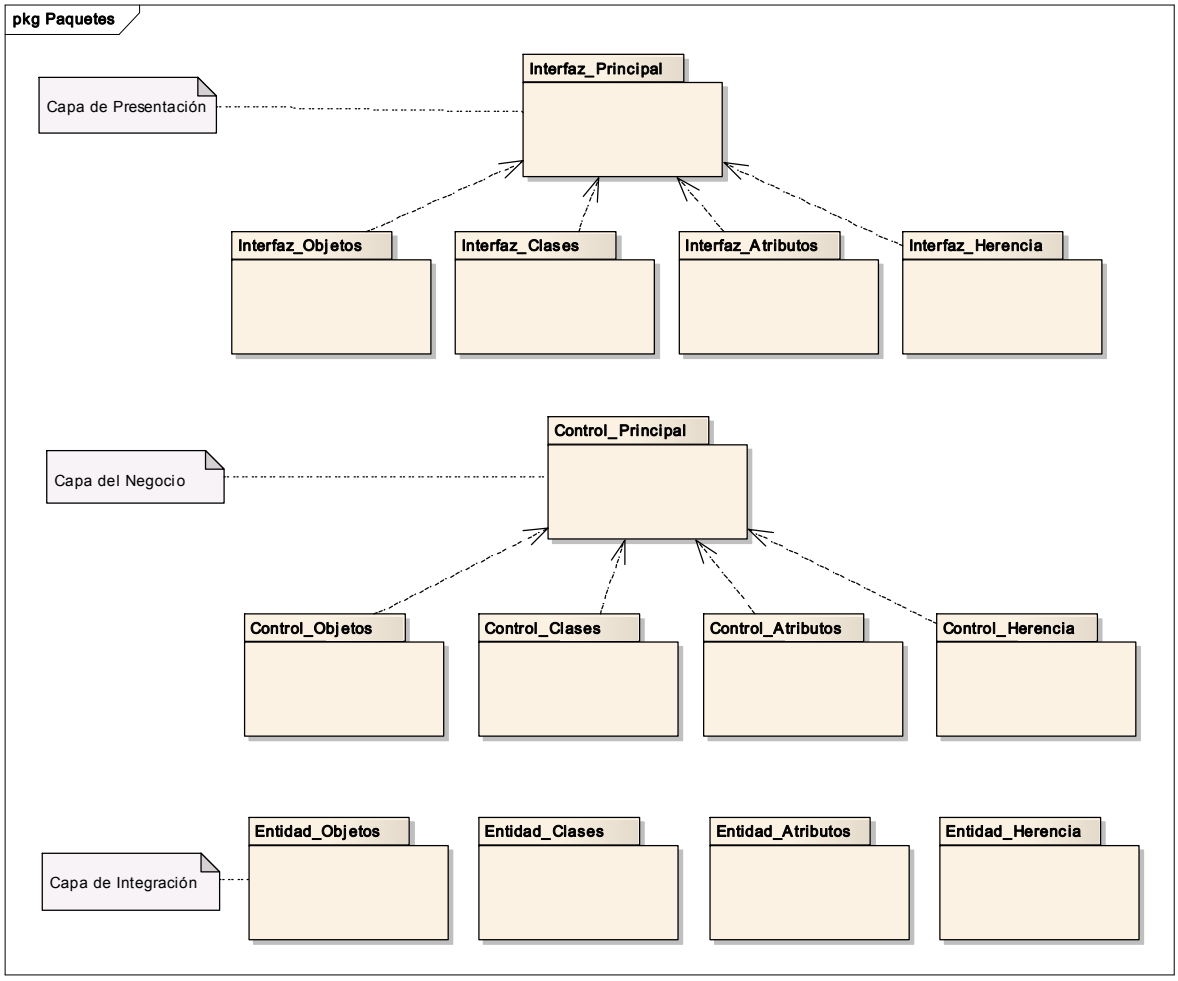


Figura 56. Diagrama de Paquetes

5.6 Diagrama de Distribución

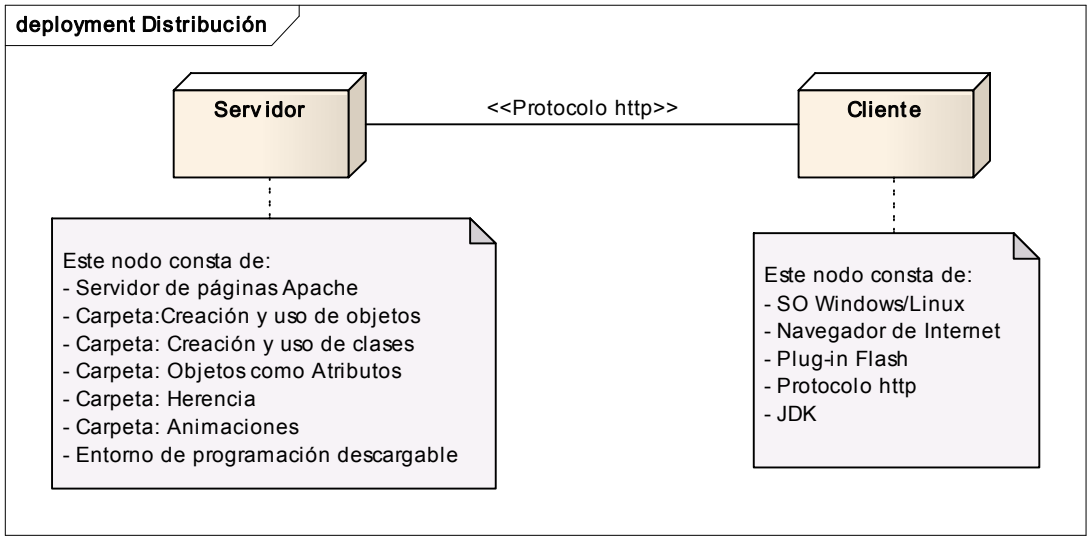


Figura 57. Diagrama de distribución

5.7 Diagramas de Clases del Diseño

5.7.1 Capa de Presentación

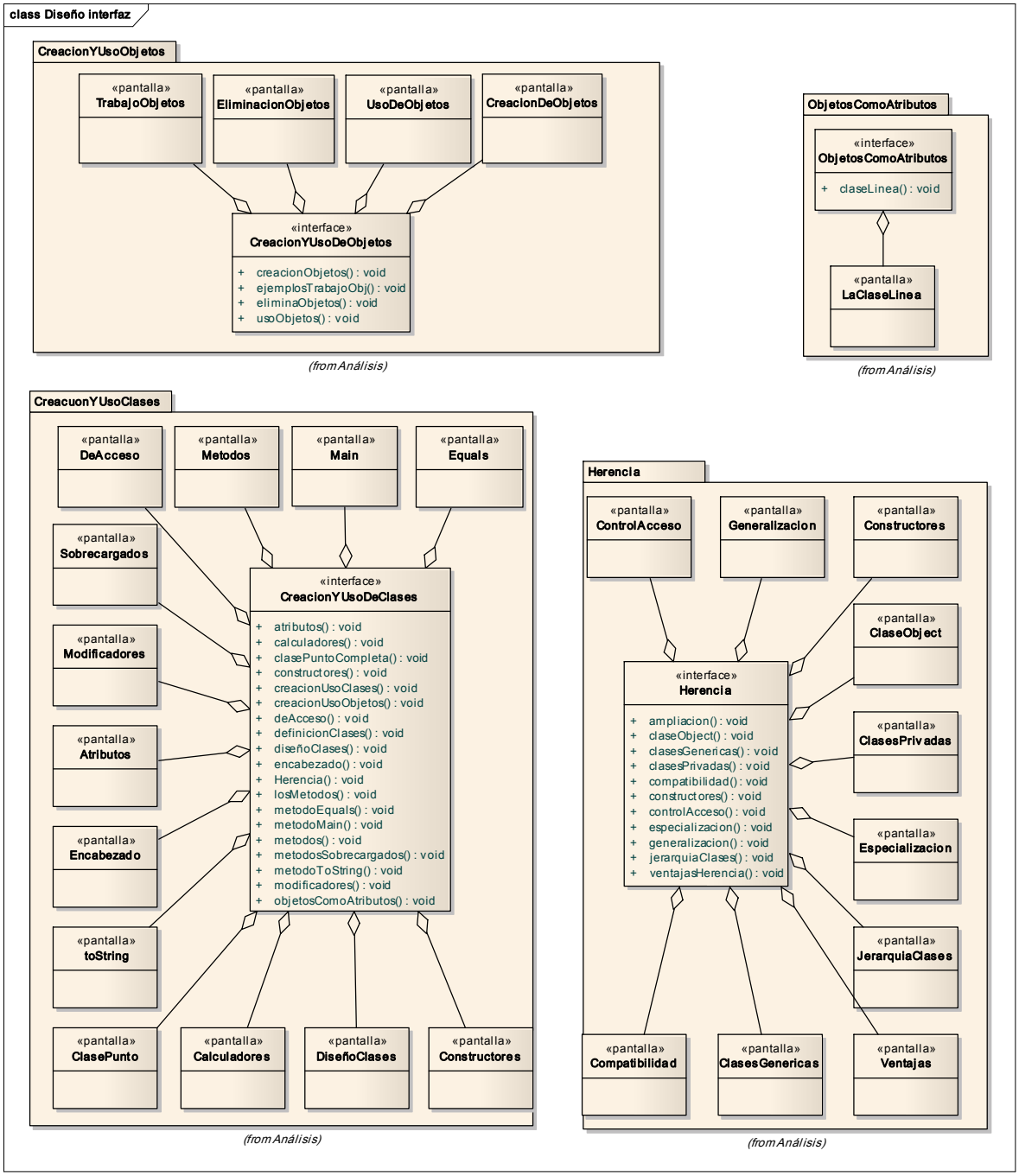


Figura 58. Diagrama de clases del diseño para la capa de Presentación

5.7.2 Capa del Negocio

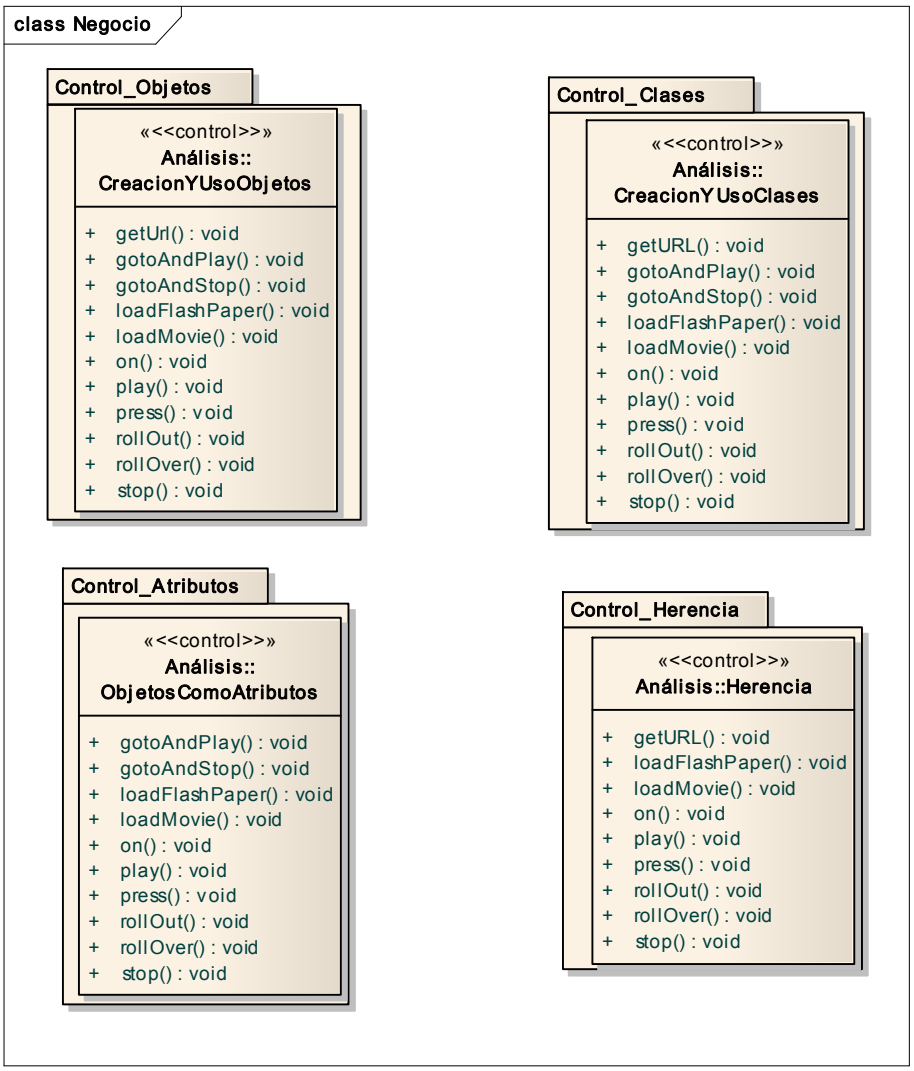


Figura 59. Diagrama de clases del diseño para la capa del Negocio

5.7.3 Capa de Integración

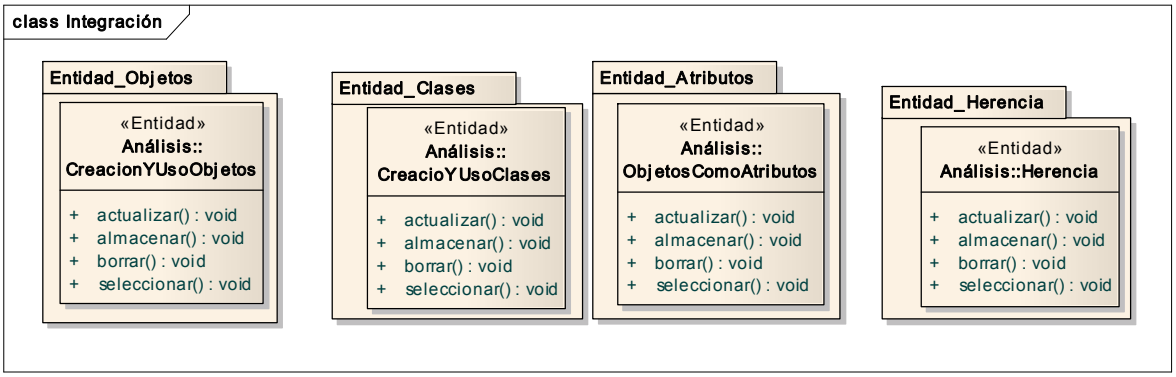


Figura 60. Diagrama de clases del diseño para la capa de Integración

CAPÍTULO VI.

IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

LA IMPLEMENTACIÓN, ES LA FASE MÁS LARGA DEL PROYECTO, YA QUE SE COMPLETA LA CONSTRUCCIÓN DEL SISTEMA TOMANDO COMO BASE LA ARQUITECTURA DESCRITA EN EL CAPÍTULO ANTERIOR. A PARTIR DE ELLA, LAS DISTINTAS FUNCIONALIDADES SON INCLUIDAS EN DISTINTAS ITERACIONES, AL FINAL DE CADA UNA DE LAS CUALES SE OBTENDRÁ UNA NUEVA VERSIÓN EJECUTABLE DEL PRODUCTO. ESTA FASE CONCLUYE CON EL HITO DE OBTENCIÓN DE UNA FUNCIONALIDAD COMPLETA, QUE CAPACITE AL PRODUCTO PARA FUNCIONAR EN UN ENTORNO DE PRODUCCIÓN.

LAS PRUEBAS SON UN FLUJO DE TRABAJO FUNDAMENTAL, CUYO PROPÓSITO ESENCIAL ES COMPROBAR EL RESULTADO DE LA IMPLEMENTACIÓN MEDIANTE LAS PRUEBAS DE CADA CONSTRUCCIÓN, INCLUYENDO TANTO CONSTRUCCIONES INTERNAS COMO INTERMEDIAS, ASÍ COMO LAS VERSIONES FINALES DEL SISTEMA QUE VAN A SER ENTREGADAS A TERCERAS PERSONAS. ASIMISMO, SE DESCRIBIRÁ LA FORMA EN QUE FUERON EFECTUADAS LAS PRUEBAS DE USABILIDAD, LA CUALES SE OCUPARON PARA VALIDAR EL CORRECTO FUNCIONAMIENTO DEL SISTEMA EN ENTORNO DE TRABAJO FINAL (UN SALÓN DE CLASES).

VI. Implementación y pruebas del sistema

6.1 Implementación del sistema

En el capítulo 5 se documentó tanto el análisis como el diseño de la aplicación, la incorporación de los diagramas de caso de uso y de secuencia que corresponden a la parte del diseño, así como los diagramas de clase del diseño, de paquetes y de distribución dan la pauta para la construcción del sistema para apoyar la enseñanza de la programación orientada a objetos.

La utilización de un software que facilite el diseño y creación de animaciones constituye un elemento fundamental para incorporar interactividad y dinamismo. Las animaciones desarrolladas se construyeron a partir de combinaciones de interpolaciones de forma y de movimiento (los dos tipos principales de animaciones en Flash), tienen el objetivo de presentar a través de una película animada los principales conceptos de la POO (figura 61).

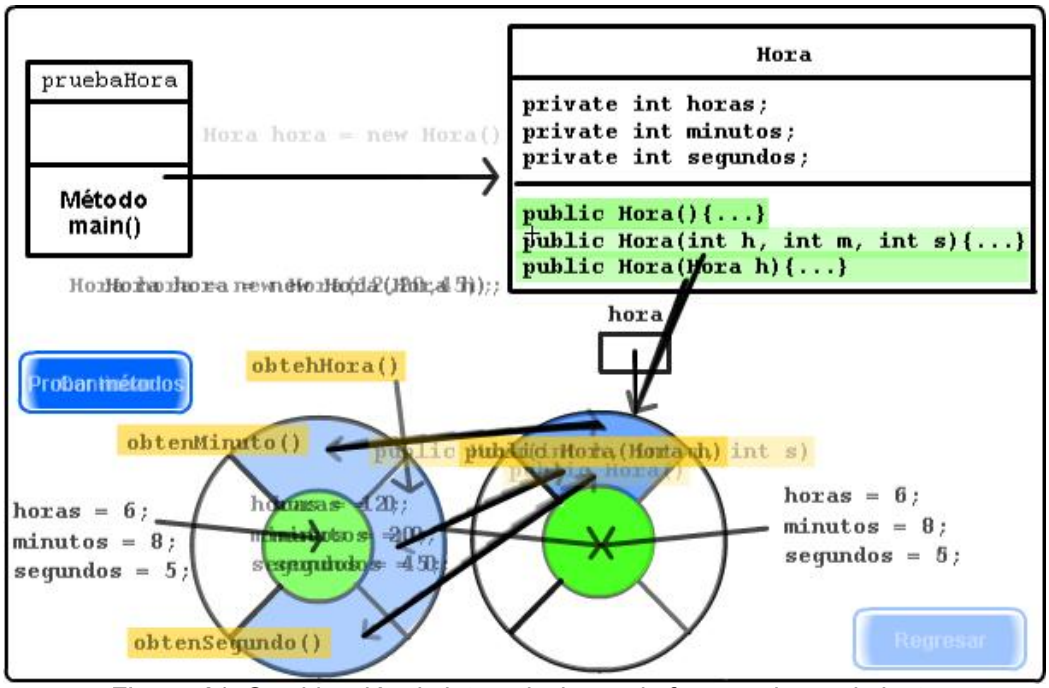


Figura 61. Combinación de interpolaciones de forma y de movimiento

Se pretende que los estudiantes, visualicen de forma gráfica las clases, la creación de objetos y la interacción entre ellos a través del paso de mensajes; los

conceptos de agregación y herencia, a través de cambios dinámicos en los objetos incorporados en cada una de las animaciones; adicionalmente y para dotarlas de cierta interactividad con el usuario de la aplicación, se añade la programación de ActionScript a través de botones, los cuales permiten: cargar y detener animaciones, continuar con la reproducción de las mismas, controlar la navegación, pausar animaciones y añadir efectos de interactividad que permitan al usuario identificar elementos particulares o bien desplegar información siguiendo la misma filosofía de los "*tooltips"*.

Para contar con un elemento que permita navegar globalmente en la información contenida en la aplicación, se ha añadido al diseño de interfaz un menú **pull – down** [48]. El diseño de menús se ha convertido en el tema central del diseño de interfaces de computadora, con el objetivo de crear aplicaciones que permitan a los usuarios con cualquier nivel de experiencia (novatos, intermedios y avanzados) poder trabajar fácilmente con ellas. Uno de los principales propósitos de un sitio Web es promover de forma efectiva información y conocimiento, razón por la cual, la forma de estructurar y vincular la información, el diseño de las estructuras de entrada y la forma en cómo se presentan; son los elementos que determinan una navegación efectiva. El diseño de menús proporciona un modelo y una lógica estructural para la organización funcional de la interfaz de usuario; además permite medir el nivel de comunicación entre el usuario y el sistema. Muchos investigadores estan de acuerdo en que los sistemas de menú pueden diseñarse para proporcionar a los usuarios un modelo organizacional eficiente y efectivo que les permita entender completamente el sitio o la aplicación y navegar sin perderse o experimentar una sobrecarga de información y conocimiento [48].

El menú de un sitio Web o de una aplicación es la parte inmediatamente visible en el diseño de la interfaz completa para el usuario. La función fundamental del menú es mostrar vínculos embebidos en la pantalla, que permitan navegar enteramente por el sitio Web o aplicación. Algunos investigadores han encontrado que los usuarios prefieren estilos de menú embebidos sobre menús explícitos. Diversas investigaciones han demostrado que los elementos de menú que emplean colores

y gráficos y que a su vez se vínculan con otros objetos semejantes y/o botones dinámicos, son más efectivos en la comunicación y son prefereridos por usuarios en sitios Web y diversas aplicaciones de escritorio. Esto se debe a que, en cualquier momento saben en qué parte del sitio o de la aplicación se encuentran, ya que mantienen en la mente una idea global de su estructura[14].

Los vínculos pueden categorizarse en dos tipos: **vínculos estructurales** y **de navegación**. Los vínculos estructurales son vínculos básicos en los que el diseño está en función de la información o estructura general del sitio Web o de la aplicación. Los vínculos de navegación son vínculos adicionales proporcionados por el diseño de menús, para mejorar la navegación. El diseño de menús debe proveer mecanismos de trayectoria flexibles, los cuales permitan al usuarios "saltar" directamente a la información buscada, sin pasar por interfaces que no le son útiles (encontrar información rápidamente). En el diseño Web, principalmente, los investigadores se han concentrado en diseñar menús que proporcionen mecanismos de navegación vertical y horizontal para una navegación más flexible [48].

Es de gran importancia el número de clics que debe dar el usuario para encontrar información específica. El número de interfaces que pueden intervenir en esa búsqueda pueden reducirse drásticamente si se adopta un diseño de menús pull – down. Este tipo de menú puede aparacer superpuesto sobre objetos que se encuentran diseñados en la interfaz, en vez de aparecer en un área de menú estática. La ventaja de este menú es que proporciona mecanismos de trayectoria bastate flexibles, comparados con otros diseños de menú. Un menú pull – down (figura 62) proporciona acceso directo a interfaces muy específicas con un solo clic; sin embargo, acelerar el acceso no mejora la estructura del sitio o aplicación. Sin una buena estructura, los usuarios difícilmente podrán saber todo lo que contiene un sitio Web o aplicación, principalmente cuando se tienen miles de interfaces donde navegar (grandes cantidades de información) [48].

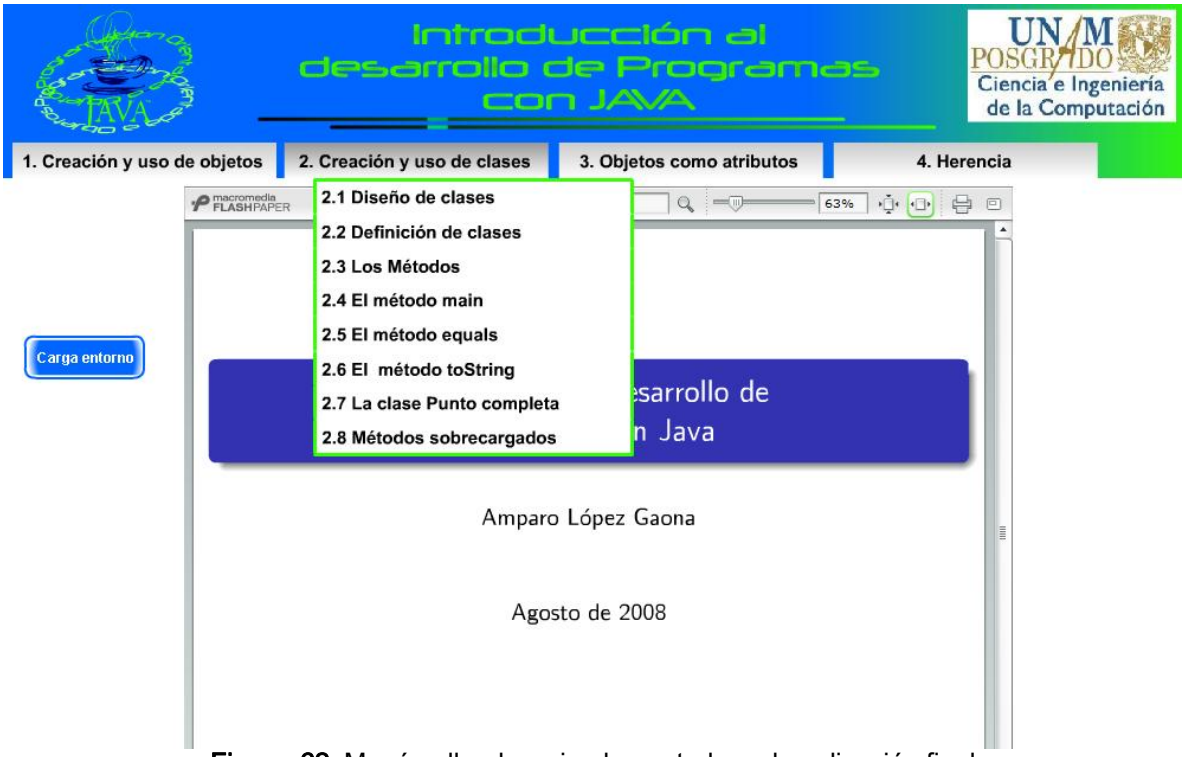


Figura 62. Menú pull – down implementado en la aplicación final

El diseño estructural de la información que contiene la aplicación se presenta en la figura 63. Se buscó en todo momento resaltar los conceptos más sobresalientes que permitan a los estudiantes comprender forma más fácil y efectiva los conceptos de la POO. El diseño de la interfaz incorpora la utilización de diapositivas, esto se hizo con la finalidad de que los usuarios encuentren sólo los puntos más importantes, que les permitan entender algún concepto o tema. Las diapositivas (es posible definir cualquier otro formato para publicación de la información) delimitan perfectamente el área donde se presenta la información. Los usuarios no tienen que navegar en una pantalla con información sobrecargada e interminable, en cada diapositiva se procuró desglosar un concepto completo y adicionalmente, se añadieron ejemplos completamente animados, para reforzar la información presentada.

Figura 63. Organización de la información en la aplicación

Publicar la información en un formato acorde con el empleado en el diseño de la aplicación es muy importante, ya que de esta forma se mantiene un diseño homgéneo y por eso se utiliza la herramienta FlashPaper; esta herramienta trabaja a través de una impresora virtual que permite crear documentos de tipo SWF o PDF (dos formatos compactos y seguros), tomando como base cualquier archivo imprimible; los documentos que se crean con esta herramienta son universalmente accesibles por más del 98% de usuarios en cualquier navegador Web y de esta forma, el documento se puede ver sin la aplicación que lo creó originalmente. Los documentos creados a partir de esta herramienta son fácilmente añadibles a cualquier proyecto Flash; además cada documento generado conserva toda la navegación interna, la cual es independiente de la que se mantiene en la aplicación, como se muestra en la siguiente figura:

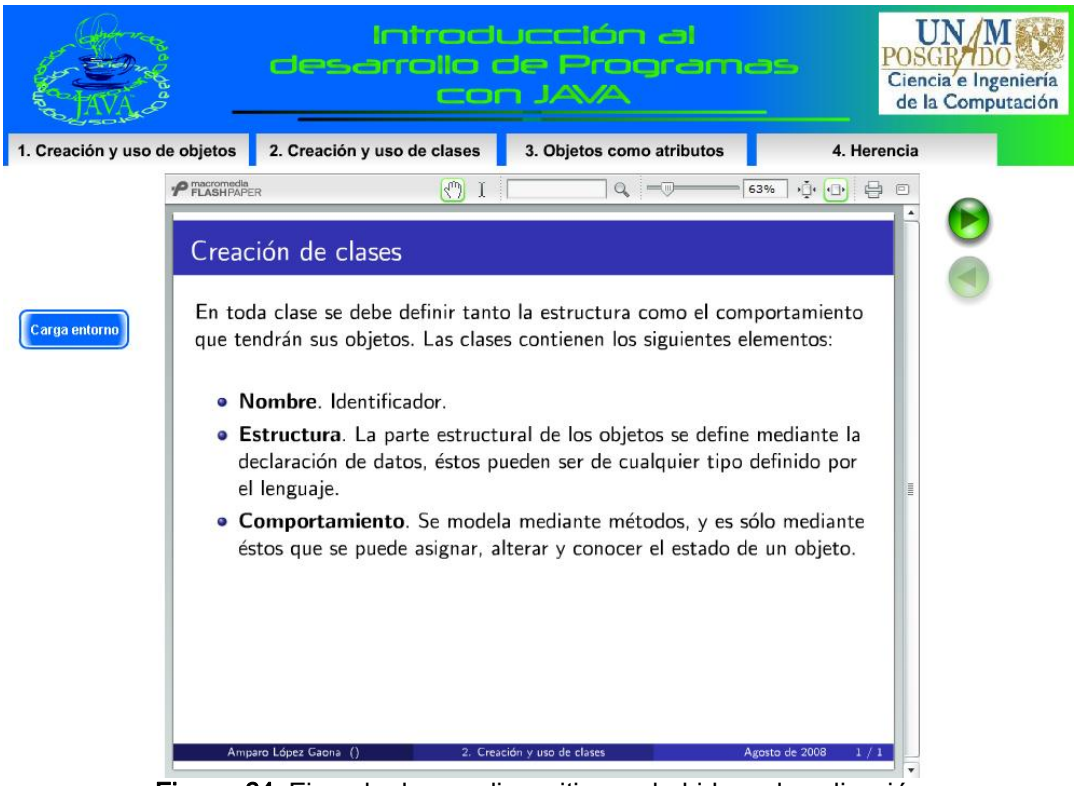


Figura 64. Ejemplo de una diapositiva embebida en la aplicación

Para brindar interactividad con el usuario, se incluye en la aplicación un conjunto de botones (figura 65), los cuales permiten: cargar los diversos ejemplos que se han animado, controlar las animaciones, decidir la forma de ejecución de una animación (dependiendo de las condiciones del problema), controlar la navegación general (avanzar y retroceder diapositiva), ejecutar las animaciones y cargar las distintas dispositivas que contienen la información que presenta la aplicación.

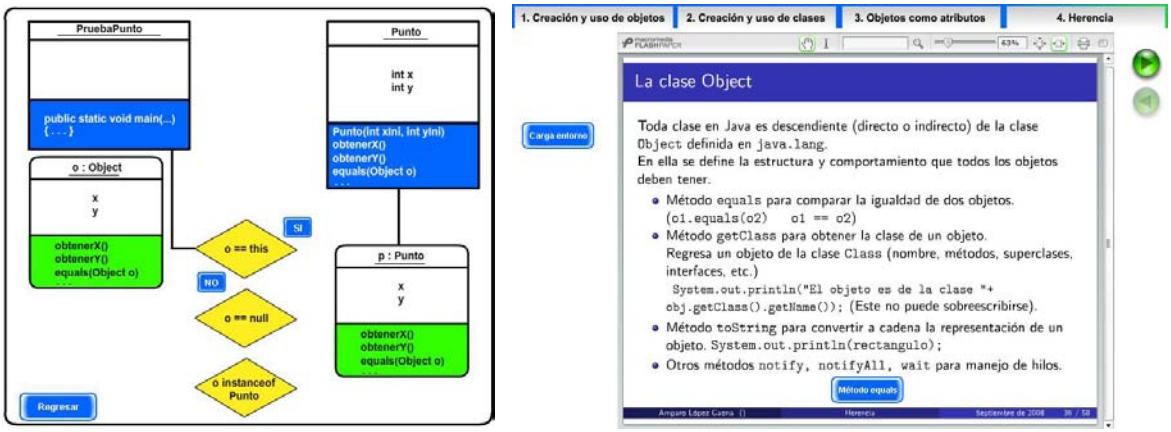


Figura 65. Ejemplos de botones creados para controlar la aplicación

Los botones que se han incorporado, utilizan las características del símbolo BOTÓN de Flash, el cual permite el diseño gráfico de los tres estados que componen a un botón: *reposo, sobre* y *presionado*. El control y asignación de las acciones que realiza un botón dependen del evento a partir del cual se desea que se ejecute la acción (presionar, liberar, etc.), a través del lenguaje de programación ActionScript.

Finalmente, la aplicación se publica en un formato universal que permita visualizarla sin la necesidad de instalar alguna aplicación muy particular, para lograr esto, se ha elegido el formato HTML. La opción de publicación Flash crea los archivos necesarios para que una película se visualice desde Internet a través de cualquier navegador. Con esta opción, el profesor puede teclear una dirección en su navegador y tener acceso a la aplicación, o bien, es posible que el profesor tenga la aplicación en su laptop o incluso traerla en una memoria USB y desde ahí ejecutarla a través del navegador de Internet.

La implementación final de la aplicación se presenta a continuación. La interfaz principal del sistema es el punto de entrada para tener acceso a la información disponible en la aplicación, se compone de un menú principal con el cual se puede seleccionar cualquiera de los temas y subtemas desarrollados. El espacio central de la interfaz se destina para mostrar la información a través de dispositivas; del lado derecho de este espacio se incorporaron dos botones, el primero de ellos permite cargar el entorno ligero de programación, el segundo botón, permite descargar la biblioteca de códigos Java. Para cada subtema que se haya elegido se incorporan además un par de botones que sirven para el control de la navegación entre las diapositivas; siempre que se permita, el usuario podrá avanzar a la siguiente o retroceder a la anterior diapositiva. A continuación se presentan las capturas de las pantallas principales que componen la aplicación.

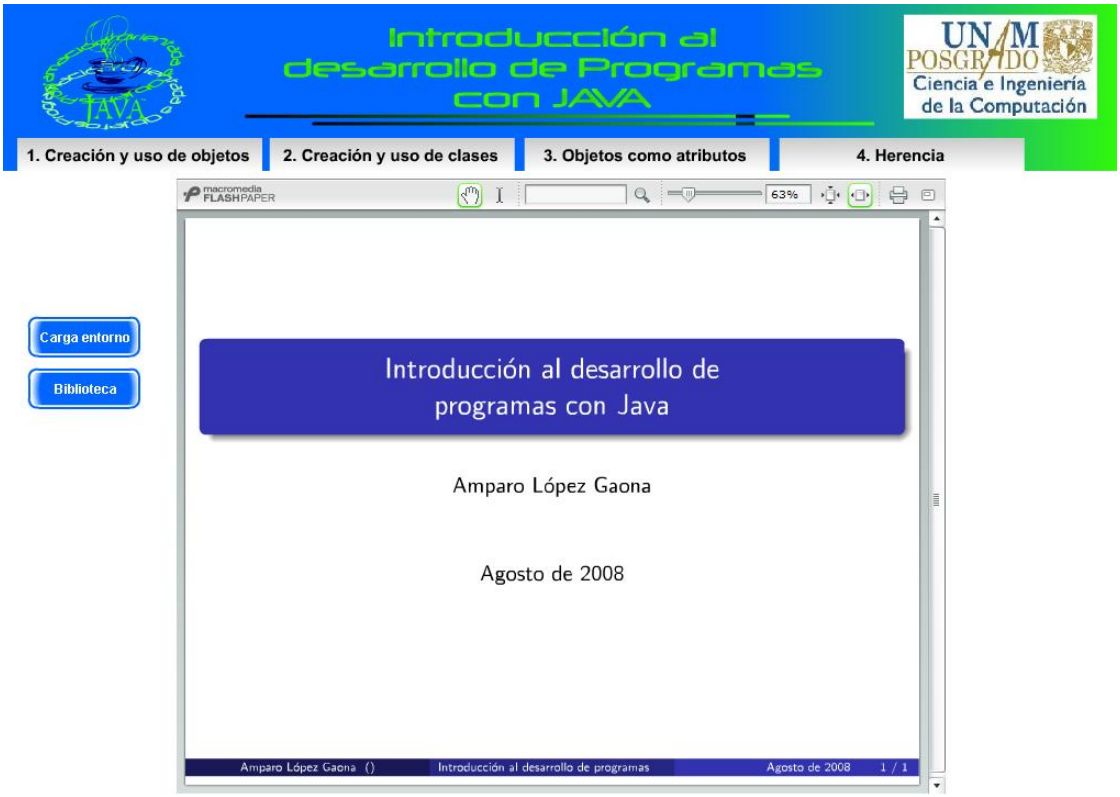


Figura 66. Interfaz principal de la aplicación.

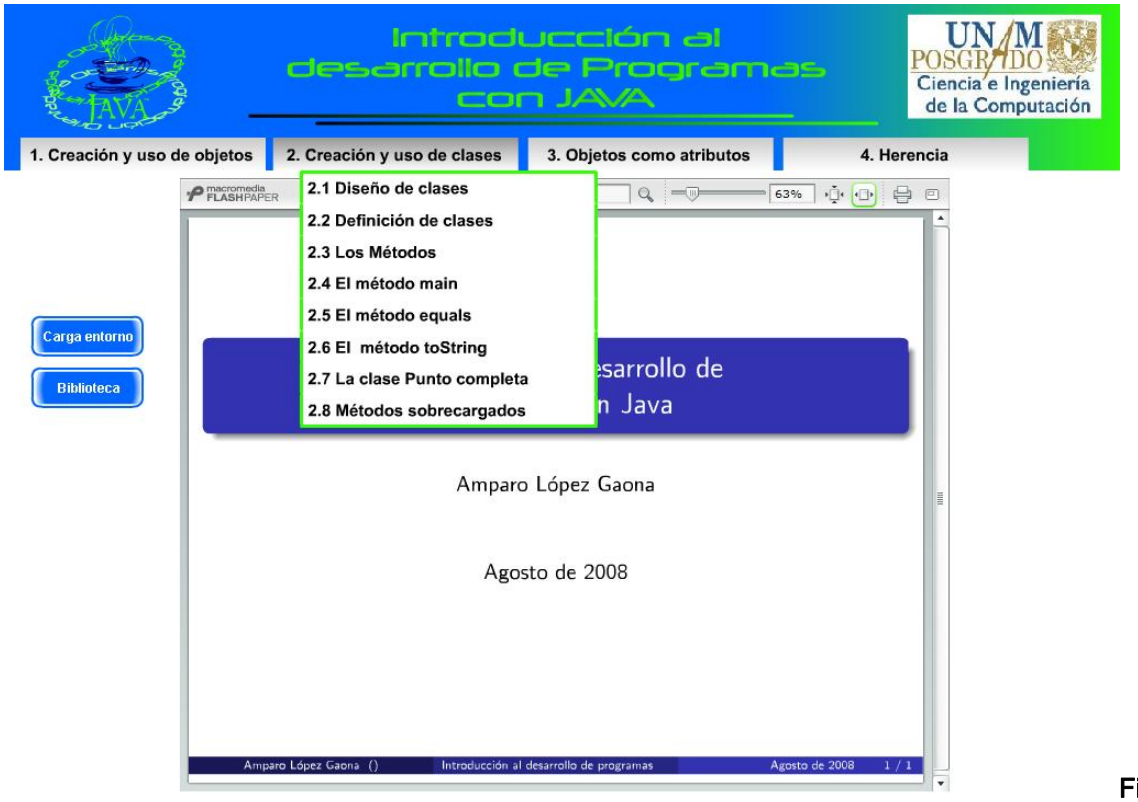


Figura 67. Menú principal de la aplicación con diseño pull – down

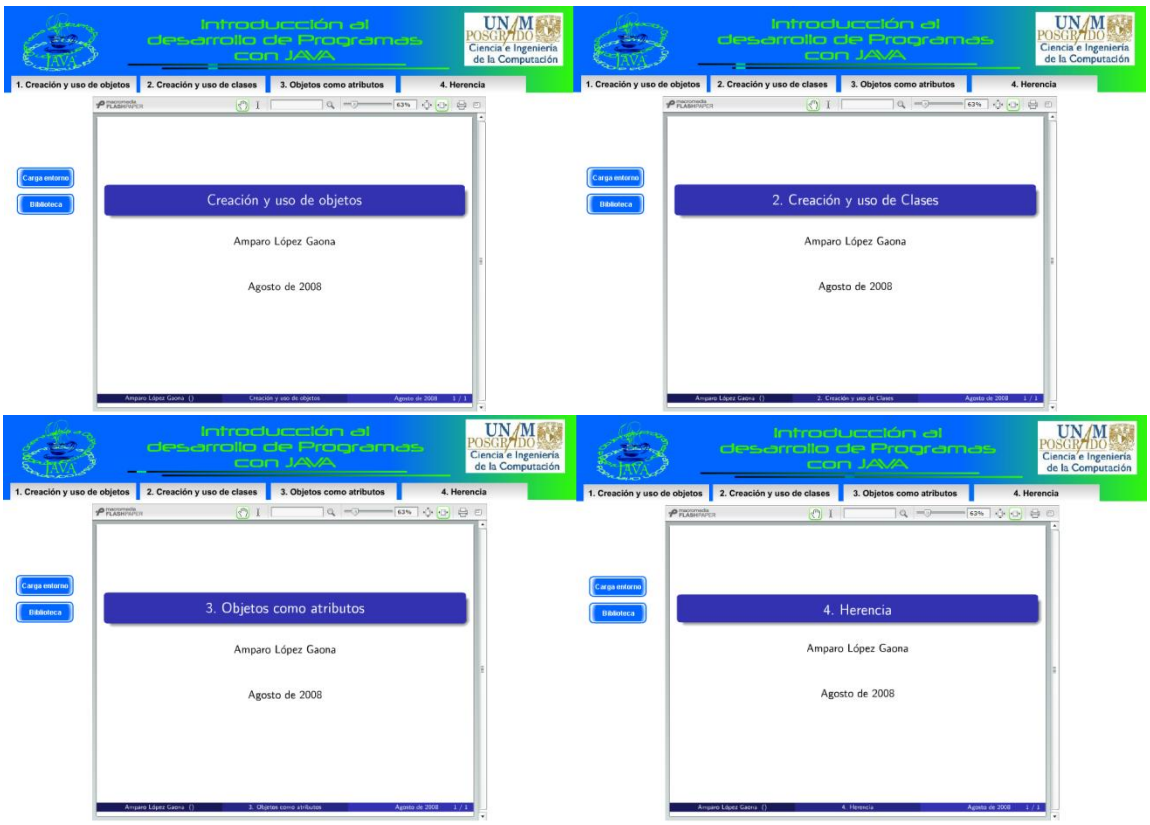


Figura 68. Interfaces diseñadas para cada uno de los temas de la aplicación

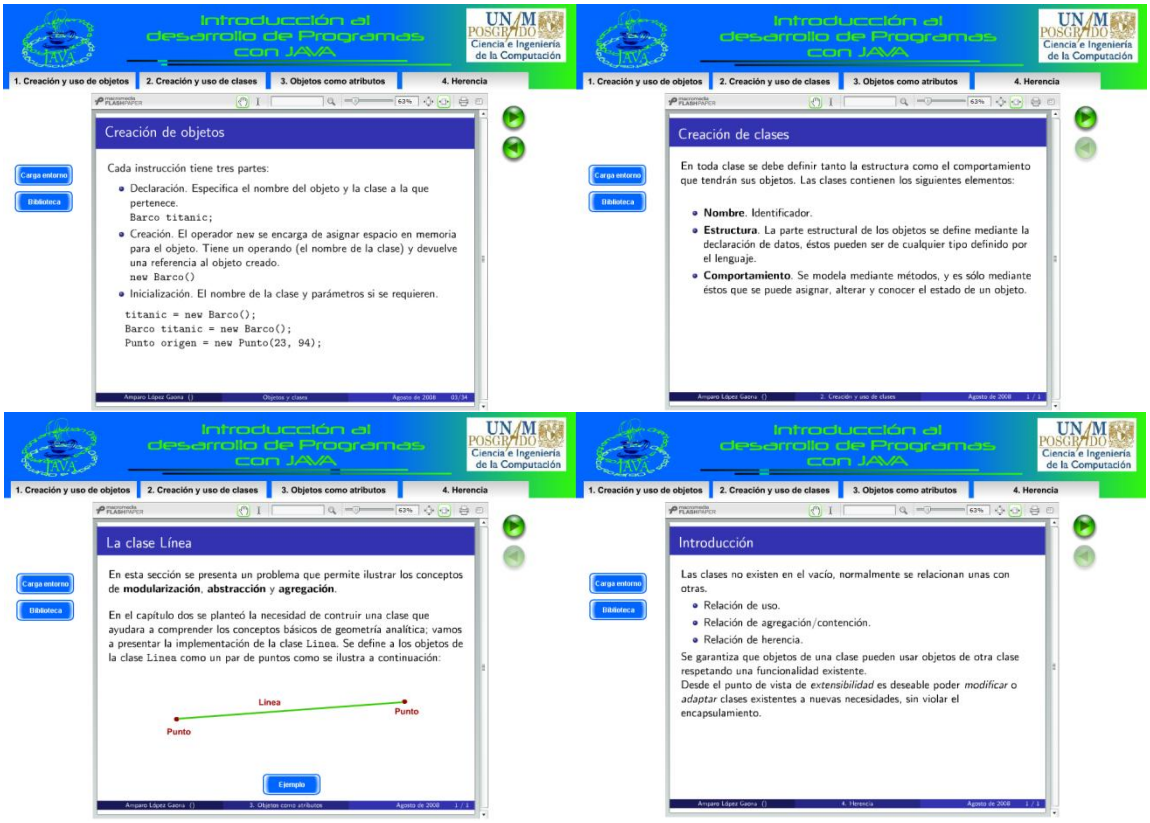


Figura 69. Ejemplos de las pantallas para algunos de los subtemas

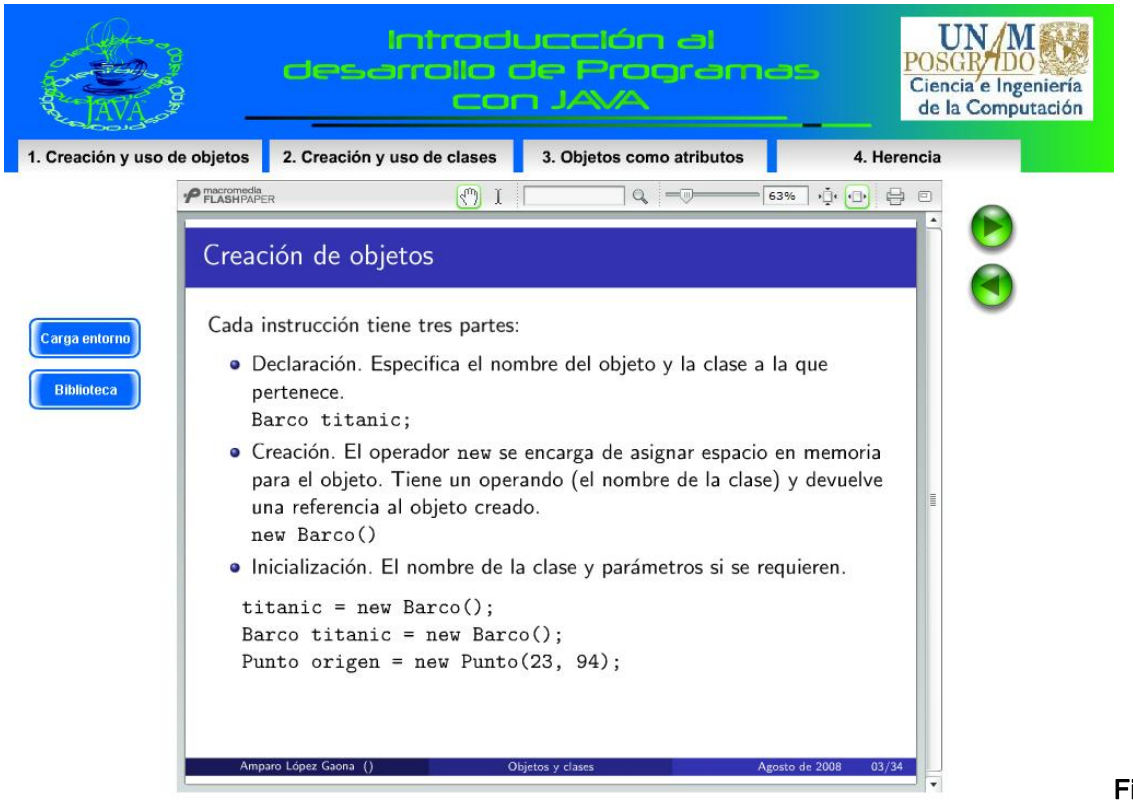


Figura 70. Ejemplos de botones creados para controlar la navegación en los subtemas

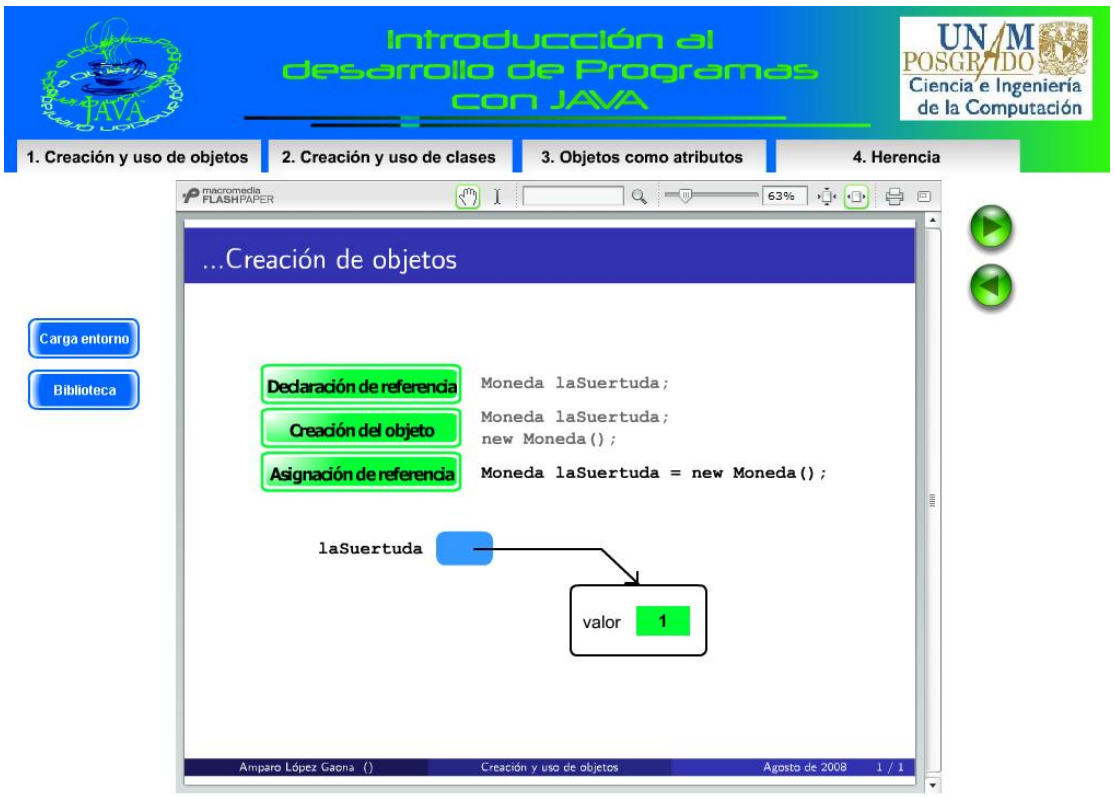


Figura 71. Ejemplo de una animación combinada con una de las dispositivas

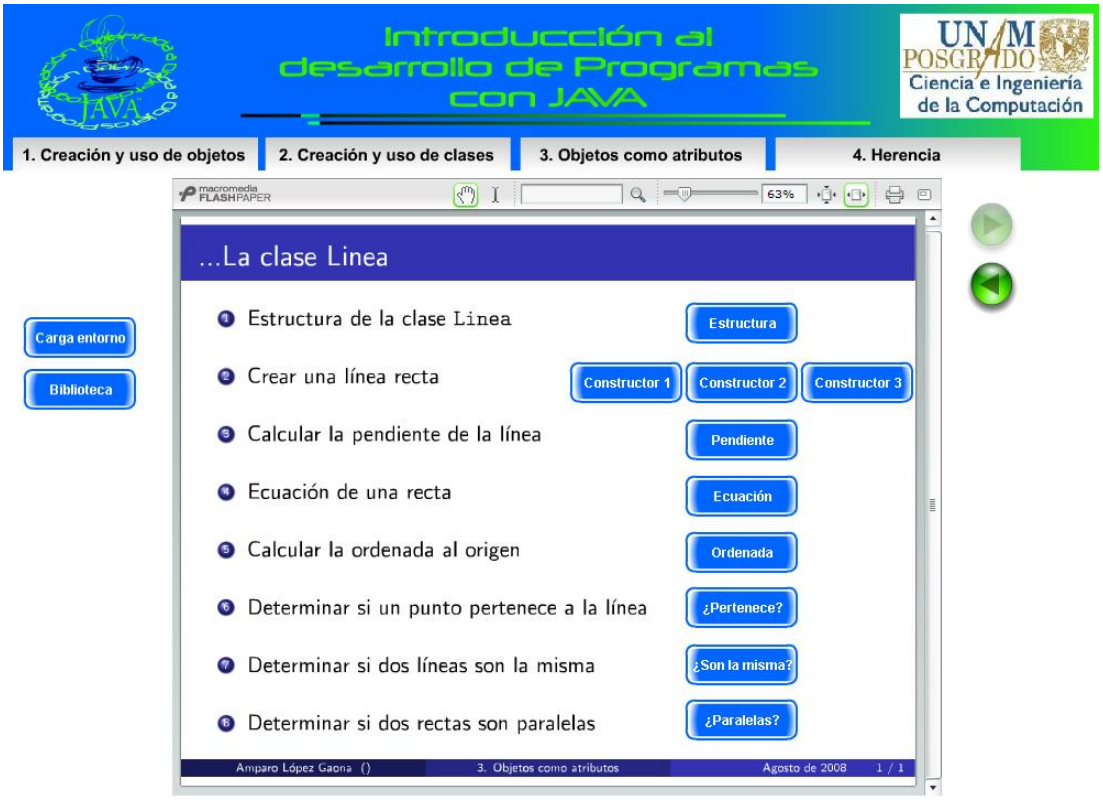


Figura 72. Ejemplos de botones para cargar los ejemplos animados

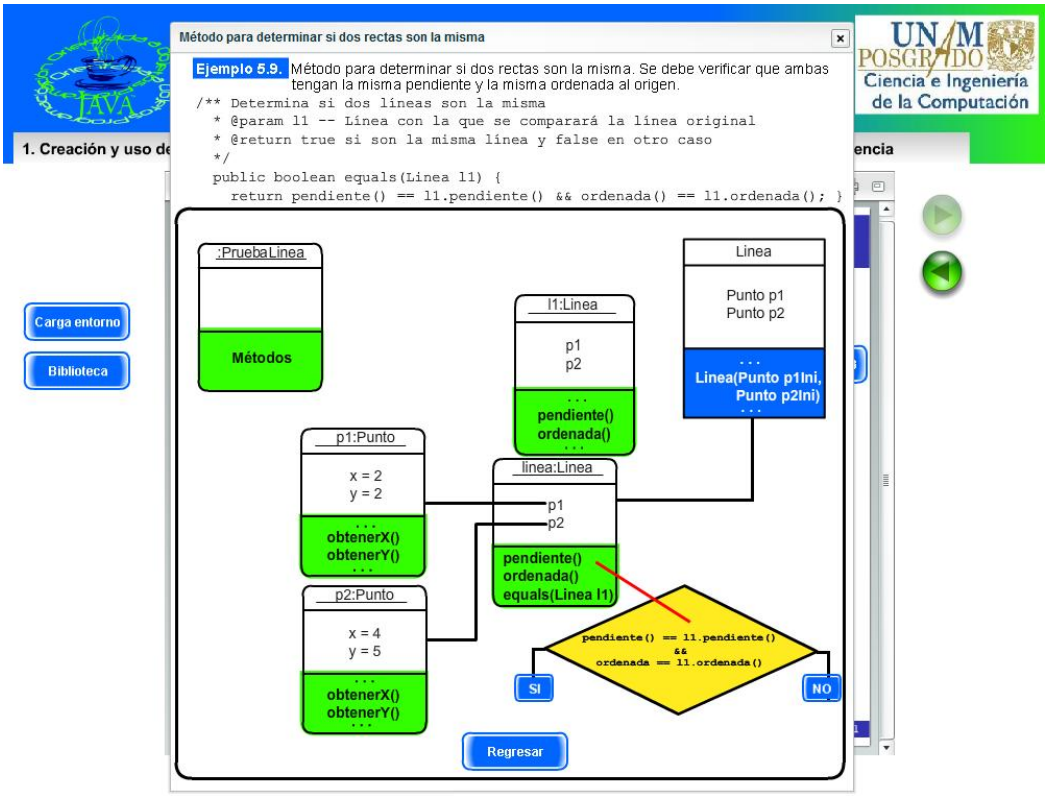


Figura 73. Ejemplo de una animación ejecutándose independiente de la aplicación

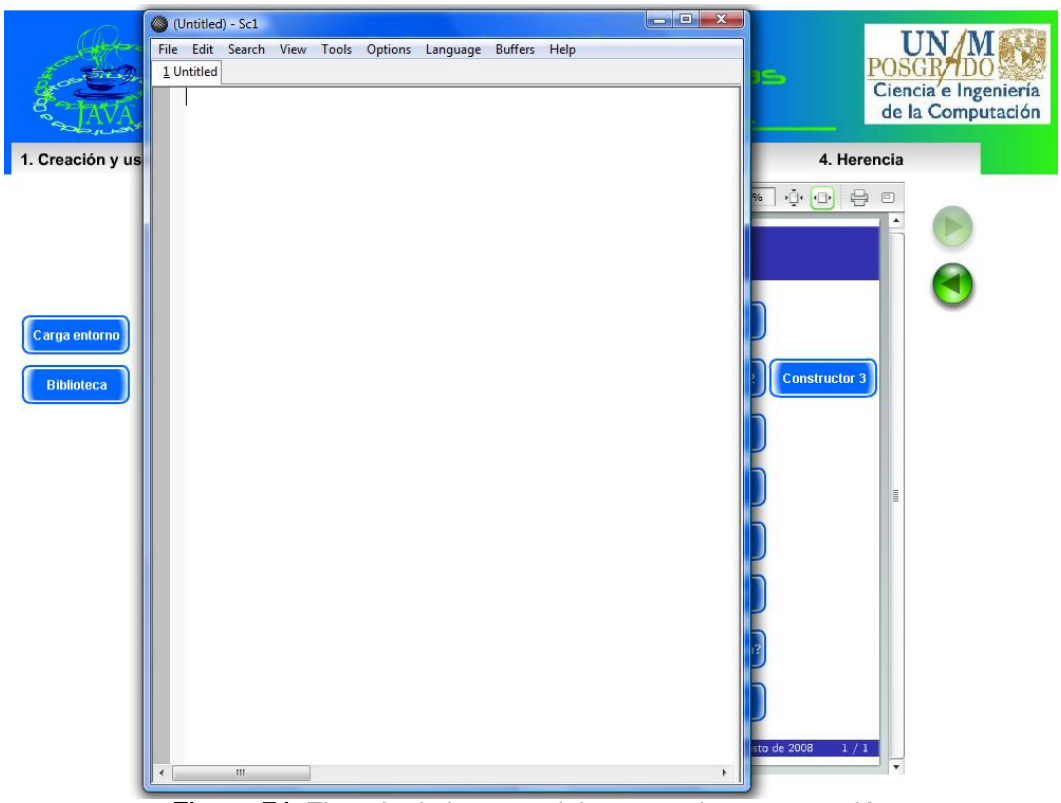


Figura 74. Ejemplo de la carga del entorno de programación

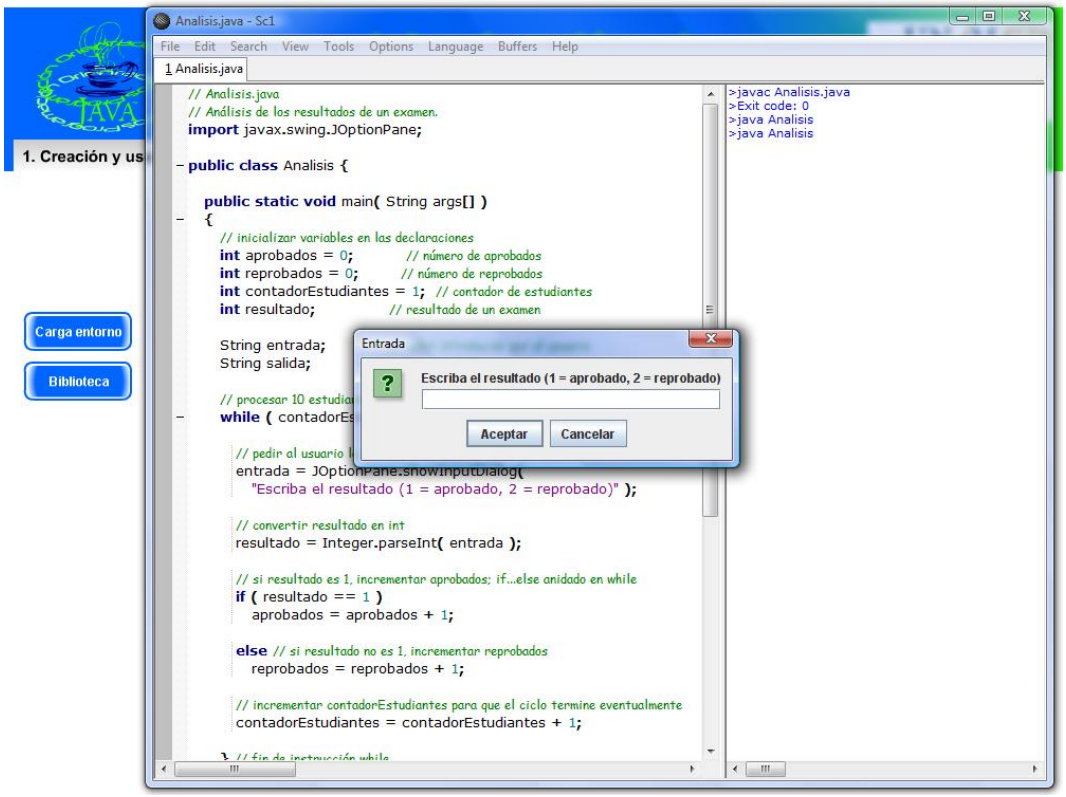


Figura 75. Carga, compilación y ejecución de un programa en Java

6.2 Pruebas del sistema

Evaluar una interfaz es el proceso por el que se determina el valor o la calidad de la misma en relación a objetivos bien definidos. Por lo tanto, no es sólo el medir dicha calidad, sino que la evaluación pretende alimentar el proceso de mejora continua de la interfaz, teniendo como meta conseguir el grado máximo de usabilidad [7].

Se define a la usabilidad de acuerdo a la norma ISO 9241-11 como l*a medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado.* La efectividad es la precisión y la plenitud con la que los usuarios alcanzan los objetivos especificados. A esta idea van asociadas la facilidad de aprendizaje (procurando sea lo más amplio y profundo posible), la tasa de errores del sistema y la facilidad del sistema para ser recordado (que no se olviden las funcionalidades principales ni sus procedimientos). Por eficiencia se entenderán los recursos empleados en relación con la precisión y plenitud con que los usuarios alcanzan los objetivos especificados. Y por satisfacción se entenderá la ausencia de incomodidad y la actitud positiva en el uso del producto; tratándose entonces de un factor subjetivo [7].

Un método de evaluación es aquel procedimiento en el que se recaba información relevante sobre la operación y usabilidad de un sistema. La evaluación puede ayudar a la interfaz a ser usable, y debe tenerse en cuenta que los aspectos detectados en la interfaz no sólo van a mejorar esta interfaz sino el producto en su conjunto. En este sentido, el mayor problema en la evaluación es que, la perfección no existe, incluso después de múltiples pruebas realizadas con numerosas y complejas metodologías y ayudándose de numerosos expertos, diseñadores y usuarios, el éxito no está asegurado [7].

Un sistema o proceso de evaluación será útil y válido en relación a su capacidad de identificar problemas, y las posibles soluciones, para una interfaz. La evaluación siempre es necesaria, aunque se tenga poco tiempo y se considere

que es un proceso que requiere demasiados recursos. El plan de evaluación debe ir de acuerdo a la etapa de diseño en la que se encuentre la interfaz (figura 76). Se debe evaluar siempre y de forma continua. El evaluar supone ahorrar recursos y da objetivos al desarrollo. La meta principal de toda evaluación es la mejora de un producto [7].

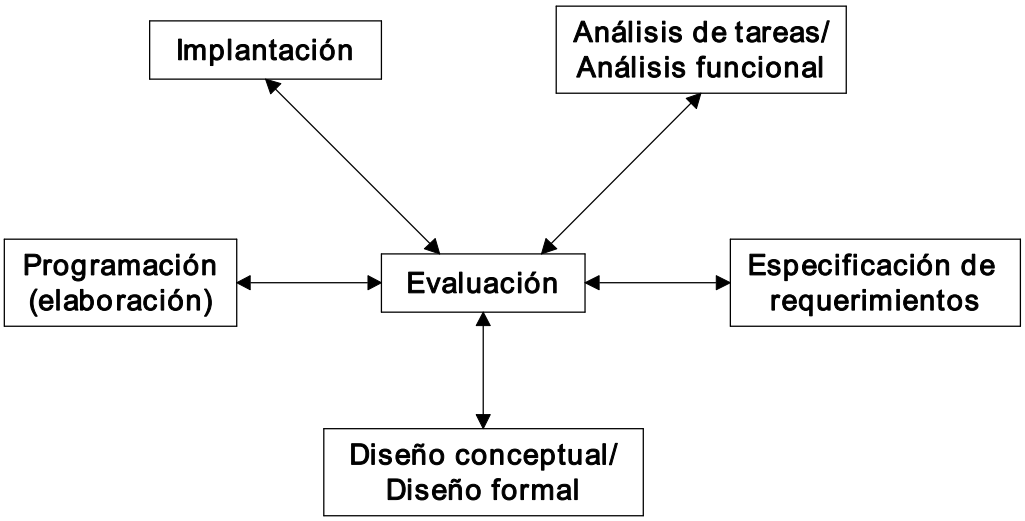


Figura 76. Proceso de evaluación

Seleccionar un plan de evaluación y/o unas vías concretas de comunicación con el usuario dependen de los aspectos siguientes

- ✓ El entorno del proceso.
- ✓ La naturaleza de la interfaz.
- ✓ La fase del proyecto en la que se encuentra.
- ✓ La información que se pretende conseguir con la evaluación.
- ✓ La experiencia en el campo del diseño y de la evaluación del equipo evaluador.
- ✓ Los recursos disponibles
- ✓ El grado de calidad en el proceso de evaluación que se pretende conseguir.

En conclusión, es necesario no precipitarse y realizar un proceso de evaluación lo más adecuado y razonable posible.

6.2.1 Tipos de evaluaciones

En el caso de las evaluaciones, se encuentran principalmente dos grandes tipos:

- ✓ **Evaluaciones con usuarios:**

En este tipo de evaluaciones se pone a prueba al sistema observando la forma en que los usuarios trabajan con él. Este análisis permite obtener información objetiva y subjetiva del usuario, y comprobar de una manera fiable si la aplicación evaluada resulta usable. La principal ventaja de efectuar estas evaluaciones es que son concluyentes; sin embargo, son evaluaciones costosas ya que es necesario conseguir usuarios, contar con instalaciones especiales y consumen mucho tiempo. Sí no se diseñan con cuidado, los resultados que arrojen no sirven para hacer un juicio de valor. Las evaluaciones con usuarios se pueden llevar a cabo:

- En ambiente controlado: Ambiente similar, de manera que los aspectos que lo afecten sean controladas. Sus ventajas son que existe un control más alto de las condiciones de trabajo y un registro de resultados más fidedigno. La desventaja es que requieren de instalaciones especializadas, eventualmente se pueden suprimir condiciones que si afectan en la vida real.
- En el terreno: Ir a dónde se esta trabajando y donde finalmente se utilizará la aplicación. La ventaja es que no se requiere de instalaciones especializadas; sin embargo, se pierde el control de muchos aspectos.

- ✓ **Evaluaciones sin usuarios:**

Son el tipo de pruebas que el desarrollador o el equipo de desarrollo efectuan a lo largo del ciclo de vida de un producto de software. Las ventajas que tienen es que son más baratas, es decir, requieren menos dinero, tiempo, esfuerzo, etc., debido a esto, se pueden llevar a cabo desde los primeros prototipos. La gran desventaja que tienen es que no son concluyentes. Este tipo de evaluaciones se pueden llevar a cabo: con base en documentos (guías de estilo, guías de uso, criterios), con base en modelos de usuario y/o de interacción y con base en un experto.

6.2.2 Aplicación de las pruebas

Las pruebas para esta aplicación se realizaron en dos direcciones:

- ✓ **Plan de pruebas del sistema**. Su objetivo es identificar las pruebas que son necesarias para asegurar el cumplimiento de los requerimientos especificados. La manera de efectuarlo es tomas como base cada uno de los casos de uso especificados y se proponen las entradas para verificar que se cumpla con lo especificado en cada caso de uso.
- ✓ **Evaluaciones con usuarios**. En este caso se efectuaron dos tipos de pruebas:
  - **Con un grupo de la Facultad de Ciencias**
  - **Con cinco usuarios**

  En este sentido, las pruebas se aplicaron sobre los usuarios finales, es decir, sobre los alumnos. Esta decisión se debe a que, aunque se trata de una herramienta para apoyar a los profesores, serán los alumnos quienes procesen la información y los ejemplos mostrados en la aplicación, por lo que es importante que dicha información se encuentre clara y conscisa; que los conceptos mostrados sean claros. De esta manera es posible tener una aplicación robusta, que lejos de confundir ayude al profesor y que los alumnos la perciban como una herramienta que responda sus necesidades.

6.2.2.1 Plan de pruebas del sistema

A continuación se definen las entradas para cada uno de los requerimientos analizados, además, se detallan los diferentes casos de prueba para verificar todas las acciones importantes en los casos de uso; de esta manera el usuario, estará en posibilidad de comprobar que se cumplieron con todas sus necesidades.

Prueba 1 **Entrar al sistema**

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos

Entrar al sistema	Introducción de dirección de la página del sistema incorrecta.	El navegador despliega el mensaje de página no encontrada.	Se obtuvo el resultado esperado.
	Introducción de la dirección del sistema correcta.	El navegador despliega la página del sistema.	Se obtuvo el resultado esperado.

Prueba 2 **Desplegar información de POO**

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
Desplegar información de POO	No se selecciona ninguna opción del menú desplegable.	El sistema no hace nada pues para que se despliegue la información contenida en el sistema, es necesario que se seleccione alguno de los subtemas del menú principal.	Se obtuvo el resultado esperado.
	Se selecciona cualquiera de los temas o subtemas mostrados en el menú.	El sistema despliega la interfaz adecuada, dependiendo de la opción seleccionada.	Se obtuvo el resultado esperado.
	Se elimina alguno de los subtemas del sistema y se intenta seleccionar después desde el menú principal.	El sistema indica que el tema seleccionado no se encuentra disponible	Se obtuvo el resultado esperado
	Se selecciona alguna de las animaciones contenidas en la aplicación.	El sistema carga la animación seleccionada en una ventana independiente del sistema.	Se obtuvo el resultado esperado
	Se elimina alguna de las animaciones del sistema y se intenta seleccionar después desde el subtema correspondiente.	El sistema indica que la animación seleccionada no está disponible.	Se obtuvo el resultado esperado
	En el caso de las animaciones, se prueban los controles que permiten la interactividad con el usuario	La animación se ejecuta de manera satisfactoria, la animación se ejecuta de acuerdo a la opción	Se obtuvo el resultado esperado.

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
		seleccionada.	
	Se introduce otra dirección en la barra de direcciones del navegador.	El usuario no selecciona ninguna opción y sólo se cambia de página.	Se obtuvo el resultado esperado.

Prueba 3 **Ejecutar animaciones**

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
Ejecutar animaciones	No se selecciona el botón que permite cargar la animación en una ventana.	El sistema no hace nada pues para que se despliegue la animación de un subtema particular, es necesario que se seleccione alguno de los botones habilitados para cargarlas.	Se obtuvo el resultado esperado.
	Se selecciona cualquier botón de los habilitados para cargar una animación.	El sistema despliega la interfaz adecuada, dependiendo de la animación seleccionada.	Se obtuvo el resultado esperado.
	Se elimina alguna de las animaciones del sistema y se intenta seleccionar después desde el botón que la carga.	El sistema indica que la animación seleccionada no se encuentra disponible	Se obtuvo el resultado esperado
	Se prueban los controles que permiten la interactividad con el usuario	La animación se ejecuta de manera satisfactoria, la animación se ejecuta de acuerdo a la opción seleccionada.	Se obtuvo el resultado esperado.
	Una vez que finalizó la ejecución de una animación, el usuario da clic en el botón que permite volver a	El sistema vuelve a reproducir la animación solicitada	Se obtuvo el resultado esperado.

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
	reproducir la animación.		
	Se introduce otra dirección en la barra de direcciones del navegador.	El usuario no selecciona ninguna opción y sólo se cambia de página.	Se obtuvo el resultado esperado.
	El usuario da clic en alguna otra parte de la interfaz del sistema.	El sistema despliega la opción que el usuario haya seleccionado o bien, abre otra página o bien cierra la ventana de navegador.	Se obtuvo el resultado esperado.

Prueba 4 **Cargar entorno para programación**

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
Cargar entorno de programación	No se selecciona el botón que permite cargar el entorno de programación.	El sistema no hace nada pues para que se cargue el entorno, es necesario que se seleccione el botón que se ha habilitado para realizar esta operación.	Se obtuvo el resultado esperado.
	Se selecciona el botón que permite cargar el entorno ligero de programación.	El sistema despliega el entorno de programación en una ventana independiente.	Se obtuvo el resultado esperado.
	Se elimina el entorno del sistema y se intenta cargar posteriormente.	El sistema indica que entorno de programación no se encuentra disponible	Se obtuvo el resultado esperado
	Se escribe un programa en el área dentro del entorno destinado para ello.	El entorno muestra el programa que se acaba de teclear.	Se obtuvo el resultado esperado.
	Se elige del menú Archivo la opción de guardar el archivo que se acaba de teclear.	El entorno muestra la interfaz adecuada para guardar el archivo con el nombre elegido	Se obtuvo el resultado esperado.
	Se da clic en el botón	El sistema cierra el	Se obtuvo el

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
	cerrar del entorno.	entorno si no había programa por guardar, en caso contrario pregunta si se desea almacenar.	resultado esperado.
	Se selecciona la opción Abrir archivo desde el menú Archivo.	El entorno muestra la interfaz que permite explorar el disco duro en busca del archivo deseado	Se obtuvo el resultado esperado.
	Una vez que se tiene un programa en el área de trabajo, se selecciona la opción de Compilar del menú Herramientas.	El entorno compila el programa si es que no tiene errores, en caso contrario los muestra en el área de resultados.	Se obtuvo el resultado esperado.
	Una vez compilado el programa se selecciona de la opción Build del menú Herramientas.	El entorno ejecuta el programa si y sólo si, ya no tiene ningún error de compilación y muestra los resultados en la sección de resultados.	Se obtuvo el resultado esperado.
	Se intenta compilar un programa que está relacionado con otras clases inexistentes en el dispositivo de almacenamiento.	El entorno muestra errores en la ventana de resultados que indican cuales y cuantas clases hacen falta.	Se obtuvo el resultado esperado.
	Se introduce otra dirección en la barra de direcciones del navegador.	El usuario no selecciona ninguna opción y sólo se cambia de página.	Se obtuvo el resultado esperado.
	El usuario da clic en alguna otra parte de la interfaz del sistema.	El sistema despliega la opción que el usuario haya seleccionado o bien, abre otra página o bien cierra la ventana de navegador.	Se obtuvo el resultado esperado.

Prueba 5 **Descargar biblioteca de códigos Java**

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
Descargar biblioteca de códigos Java	No se selecciona el botón que permite cargar la biblioteca de códigos Java.	El navegador de internet no hace nada pues para que se descargue la biblioteca, es necesario que se seleccione el botón que tiene asignada tal operación.	Se obtuvo el resultado esperado.
	Se selecciona el botón que permite descargar la biblioteca de códigos.	El navegador despliega una venta para indicarle al usuario que desea hacer: guardar o abrir la biblioteca.	Se obtuvo el resultado esperado.
	Se elimina la biblioteca del sistema y se intenta seleccionar después desde el menú principal.	El navegador indica que la biblioteca seleccionada no se encuentra disponible	Se obtuvo el resultado esperado
	El usuario da clic en alguna otra parte de la interfaz del sistema.	El sistema despliega la opción que el usuario haya seleccionado o bien, abre otra página o bien cierra la ventana de navegador.	Se obtuvo el resultado esperado.
	Se introduce otra dirección en la barra de direcciones del navegador.	El usuario no selecciona ninguna opción y sólo se cambia de página.	Se obtuvo el resultado esperado.

Prueba 6 **Salir del sistema**

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
Salir del sistema	El usuario da clic en el botón Cerrar página del navegador.	El navegador se cierra completamente y con ello el sistema.	Se obtuvo el resultado esperado.
	El usuario no da clic en el botón Cerrar página del navegador.	El navegador no se cierra.	Se obtuvo el resultado esperado.
	Se introduce otra dirección en la barra de direcciones del navegador.	El usuario no selecciona ninguna opción y sólo se cambia de página.	Se obtuvo el resultado esperado.

Caso de uso	Entradas	Resultados	
		Esperados	Obtenidos
	El usuario da clic en alguna otra parte de la interfaz del sistema.	El sistema despliega la opción que el usuario haya seleccionado o bien, abre otra página o bien cierra la ventana de navegador.	Se obtuvo el resultado esperado.

6.2.2.2 Evaluaciones con usuarios. Con un grupo de la Facultad de Ciencias

A mediados del semestre 2009 – 1 se comenzó a utilizar el **sistema de software para apoyar la enseñanza de la POO** en el grupo de Programación 1. La prueba se realizó de la siguiente forma, se mostró a un grupo de 30 alumnos aproximadamente la herramienta en el laboratorio y se les pidió que estuvieran atentos a la explicación que se les brindaba, a la forma en que era utilizada; por otro lado se les pidió a los alumnos que pusieran atención a los conceptos que vienen incluidos en el sistema, a los ejemplos animados que se presentan para reforzar los conceptos presentados. Se les pidió que al final de la clase entregaran una hoja con comentarios sobre alguno o algunos de los siguientes rubros:

- ✓ La apariencia visual de la herramienta
- ✓ Legibilidad
- ✓ Sobrecarga de información
- ✓ Claridad de la información
- ✓ Apariencia visual de las animaciones
- ✓ Forma de presentación y ejecución de las animaciones
- ✓ Animaciones claras y entendibles

Adicionalmente, se les solicitó, para cada rubro que anotaran algún error que hubieran detectado y alguna propuesta de mejora que ellos consideraran relevante.

La utilización de la herramienta con este grupo era importante, pues se trata de un grupo de personas para quienes este curso representa su primer contacto con la programación orientada a objetos y era importante recabar sus impresiones y así poder medir el impacto que tiene el empleo de la herramienta con usuarios que de entrada no tienen contato con las Ciencias de la Computación y de esta manera identificar las fallas y las áreas de oportunidad. Por otro lado, los temas que incorpora la herramienta en su primera versión cubren gran parte del temario de esta materia.

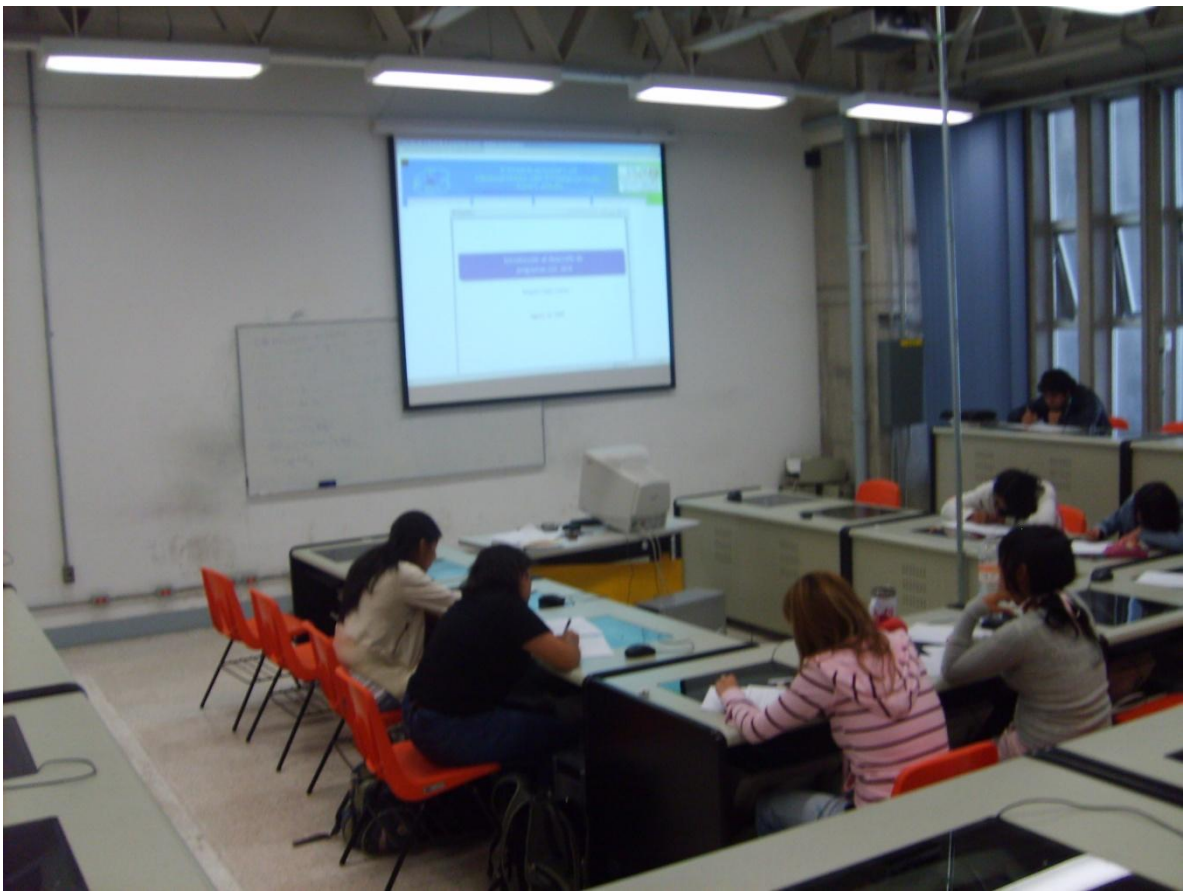


Figura 77. El sistema utilizado con el grupo de Programación 1

Una vez que se terminaron las clases en salón con la herramienta, ésta se subió a un servidor de la Facultad de Ingeniería, para poner a la disposición de los alumnos la herramienta con la cual se les habían impartido las clases de los temas que cubre la aplicación. De esta manera se les volvió a pedir una hoja con sus comentarios acerca de la aplicación y de esta manera también conocer el impacto que provocó en ellos la utilización de la herramienta por ellos mismos. Se les

dejaba que utilizarán la aplicación durante los fines de semana para que en su siguiente clase de laboratorio entregaran sus comentarios.

6.2.2.3 Evaluaciones con usuarios. Pruebas de usabilidad

En este segundo grupo de pruebas se hicieron pruebas más específicas y encamindas a medir la usabilidad de la interfaz que se ha diseñado para la presentación de la herramienta. Este tipo de pruebas tienen como objetivo revelar porqué los usuarios hacen una acción determinada de forma "correcta o incorrecta" y ofrecer opciones encaminadas al proceso de mejora de la usabilidad y utilidad. Se busca observar el comportamiento real de los usuarios, analizándolos.

Figura 78. Pruebas de usabilidad

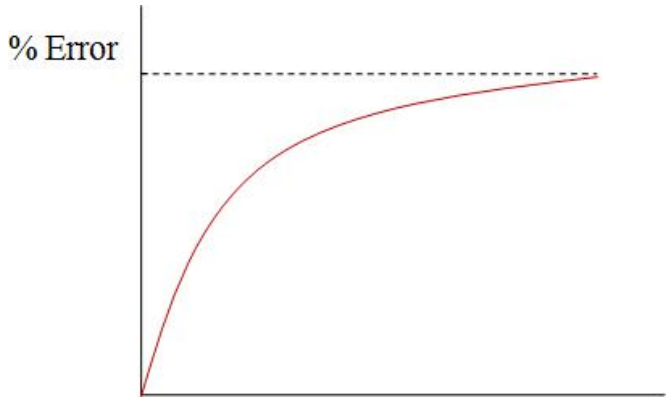
Una vez que el plan de evaluación se aprueba, los participantes en el mismo deben ser una muestra representativa de la población usuaria final de la interfaz. El personal evaluador debe controlar las reacciones físicas de la muestra de usuarios teniendo en cuenta que el objeto a evaluar no son estos usuarios sino la interfaz. Para ello, debe informarse sobre las tareas a realizar y sobre cuánto tiempo va a durar la prueba. Además, la participación debe ser total y expresamente voluntaria. La prueba debe realizarse en un entorno lo más realista posible y debe tenerse en cuenta que el usuario puede no sólo "evaluar" la interfaz, sino informar al desarrollador del entorno de funcionamiento final que, presumiblemente, va a tener el producto. Un hecho que puede distorsionar el comportamiento del usuario y que se deberá tomar en cuenta a la hora de extraer

conclusiones es la presión que supone el saberse observado. Este efecto puede atenuarse si se avisa con anticipación a los usuarios de que van a ser observados y monitoreados. La función de un evaluador debe limitarse a servir de apoyo a los usuarios, sin dar instrucciones pero atendiendo a sus demandas y explicaciones. Tras la realización de las actividades propuestas, los usuarios deben ser invitados a realizar sugerencias y comentarios o a responder algunas preguntas.

En el momento de analizar y presentar los resultados de este tipo de metodologías de evaluación, el enfoque puede dirigirse hacia dos direcciones principales:

- ✓ **Analizando el comportamiento de los usuarios evaluados**. Es decir, remarcando cómo han logrado los usuarios realizar lo convenido y dónde se han encontrado los mayores problemas. El informe de resultados de la prueba debe describir y priorizar los problemas encontrados, e intentar ser cuantitativo. El equipo evaluador debe medir las opiniones expresadas por los usuarios con la ayuda de por ejemplo, encuestas, formularios o entrevistas que se realizarán una vez finalizada la prueba en cuestión. Así, la prueba no es la meta en sí misma, sino el camino que le permitirá al usuario el poder concretar su opinión acerca de la interfaz para poder reflejarla fielmente y de una forma medible.
- ✓ **Analizando las tareas realizadas**. Tiene como objetivo realizar una estadística de logros y errores del sistema a través de las tareas llevadas a cabo en la prueba. En este caso, la medición es más sencilla, ya que se basa en el recuento del éxito o el fracaso en la *realización* de las tareas propuestas en el entorno y condiciones acordadas. Es aquí cuando variables como la "tasa de éxito", "tiempo empleado" o preguntas tipo "¿le ha gustado la interfaz?" tienen especial prioridad.

Para la realización de estas pruebas se eligieron 5 usuarios, pues en diversas investigaciones se ha comprobado que con este número de usuarios es posible encontrar el 80% de los errores en un diseño de interfaz.



PREPARACIÓN

Durante esta etapa se buscaron a los usuarios que se les iba aplicar las pruebas, se instaló el ambiente necesario en las computadoras, la dinámica a seguir y se prepararon documentos pertinentes para la ejecución de las pruebas. Se citaron a 5 usuarios (3 de la Facultad de Ingeniería y 2 de la Facultad de Ciencias). Se prepararon las computadoras a utilizar con el software necesario para realizar las pruebas.

Como dinámica a seguir se estableció lo siguiente:

1. Se leerá el protocolo de Bienvenida.
2. Se le solicitará verbalmente al usuario su autorización para filmarlo.
3. Se le proporcionarán los cuestionarios y las listas de tareas.
4. Se comenzará la filmación (en este caso sólo se cuenta con una cámara, de manera que sólo se hará una o dos filmaciones, esto por cuestiones de tiempo).
5. Al terminar el usuario su evaluación, se terminará la filmación.

Entre los documentos que se prepararon se encuentran los siguientes:

- Hipótesis de uso
- Protocolo de Bienvenida
- Cuestionario de entrada

- Lista de tareas para usuario
- Cuestionario de Salida
- Instrumento de Calificación

Hipótesis de uso

El sistema de software (herramienta para apoyar la enseñanza de la POO) ayudará a los estudiantes a comprender de una forma fácil y efectiva los principales conceptos que involucra la orientación a objetos (objetos, clases, agregación y herencia) a través de la animaciones que les permitan entender visualmente como se crean las clases y los objetos, así como la forma en que éstos se comunican con otros objetos por medio del paso de mensajes.

El sistema apoyará a los profesores a presentar su material de clases de forma dinámica. La incorporación de un entorno de programación les permitirá escribir y/o cargar programas escritos en Java, compilarlos y ejecutarlos, para darles a sus alumnos un panorama más completo de los elementos que involucra la POO.

Protocolo de Bienvenida

Se trata de una herramienta de comunicación que debe ser leída a los participantes. Describe que es lo que sucederá durante la sesión de prueba, establece el tono de la sesión para hacerla amena y agradable; esto se logra informando a los participantes que es lo que harán, haciendo hincapié en el hecho de que será el producto y no los participantes quienes serán evaluados.

Cuestionario de entrada

Proporciona información histórica acerca de los participantes y ayuda a entender su comportamiento y desempeño durante la evaluación. Está compuesto por preguntas que revelan su experiencia, aptitudes y preferencias en todas las áreas que pueden afectar su desempeño. Explora que paquetes ha usado el participante, cuánto tiempo lleva trabajando con ellos y cuántos paquetes ha

aprendido. El diseño del cuestionario debe ser breve y se aplica antes de la evaluación del producto.

Cuestionario de salida

El propósito principal del cuestionario de salida, es obtener información de los participantes para clarificar y profundizar el entendimiento de las fuerzas y debilidades del producto. La información que se obtiene incluye las opiniones y sentimientos acerca de la facilidad de uso del producto y la facilidad para aprender el sistema. Los cuestionarios escritos son usados para colectar información acerca del la población entera de participantes. Las mismas preguntas se hacen para cada individuo exactamente de la misma manera y la precisión es importante.

Ejemplos de estos formatos se presentan en el Apéndice I.

6.3 Resultados de las evaluaciones con usuarios

6.3.1 Con un grupo de la Facultad de Ciencias

En términos generales, los alumnos opinan que la aplicación les fue de gran utilidad. Los comentarios en este sentido son bastante interesantes, pues el grupo de alumnos con el que se trabajó no había tenido anteriormente ningún contacto con la programación orientada a objetos y mucho menos con el lenguaje Java; además, en el caso de los estudiantes de Actuaría, el curso de Programación 1 es su primer contacto en la Facultad con temas de programación.

Los alumnos opinan que la información que se les presenta les fue de gran utilidad, pues es clara y sencilla; la forma en cómo se estructura la información, los conceptos y los ejemplos presentados les ayudaron a comprender mejor las explicaciones que se daban en clase y en muchas ocasiones les sirvió para resolver algunas dudas. Consideran que las animaciones están bien hechas ya que ayudan a entender mejor lo que se está explicando, los ejemplos que se presentan en cada tema son claros y concisos, lo cual les agradó mucho.

Los alumnos comentan que un aspecto que les sirvió bastante, fue el hecho de que ahora no sólo podían ver la ejecución de un programa a través de la ventana de comandos, sino que además, lo pueden complementar con las animaciones de los mismos ejemplos, de esta manera, fueron bastantes los alumnos que comentan que por fin pudieron entender a que se refieren los conceptos: *clase, objeto* y *mensaje*. La manera en que se les presentó la interacción entre objetos les ayudó a comprender cómo es que se comunican los objetos y a entender el concepto de encapsulación. Otro aspecto clave es que la aplicación les permite dar un repaso completo de los temas vistos en el curso, con la ventaja de que a la par de que se explica un ejemplo, ellos pueden ver su correspondiente animación y posteriormente la compilación y ejecución del mismo.

La forma en que se presenta la información también es de gran ayuda pues les permite partir de los conceptos básicos y poco a poco, avanzando a su ritmo estudiar los temas más complejos, todo complementado con información clara y concisa. El menú principal que incorpora la aplicación es de gran utilidad pues les permite identificar muy fácilmente cualquier tema en la aplicación; esto es de suma importancia cuando se presentan dudas, pues de esta manera les es más fácil llagar al punto donde se explicó algún concepto que no haya quedado claro y ver su correspondiente ejemplo.

Otro aspecto que resaltan frecuentemente, es que les agrada que cada una de las ventanas donde se les muestra algún concepto siempre este acompañada de un ejemplo, esto es de vital importancia para ellos porque les permite resolver de forma eficaz las dudas en el momento en que surgen. Por otro lado, a pesar de que la información que se les muestra es breve, encuentran que los pasos más importantes se encuentran adecuadamente detallados y eso es importante porque les permite, por ejemplo, poder ver paso a paso el desarrollo de un programa desde cero.

De manera general, a los alumnos les agradó la idea de poder tener una herramienta que apoye la enseñanza de la POO, a través de ella, muchos de ellos pudieron entender conceptos que no les habían quedado claros, a otros, el

material que se presenta les facilitó la programación, pues la estructura que se sigue para el planteamiento de los ejemplos que se describen en la aplicación les ayudó para poder realizar sus propios planteamientos y lo más importante, por primera vez sienten que se les toma en cuenta, pues perciben a la aplicación como una herramienta destinada para los usuarios que tratan de aprender la POO y a programar con un lenguaje como Java.

A la par de los buenos comentarios realizados a la aplicación, también los alumnos expresaron sus comentarios acerca de los errores que encontraron y las mejoras que se podrían incluir, sin embargo, esta información se presentará más adelante en la sección del **Reporte de hallazgos**.

6.3.2 Pruebas de usabilidad

INSTRUMENTO DE CALIFICACIÓN

Entrar al sistema
Despliegue de la interfaz principal para el tema *Creación y uso de objetos.*
- ✓ Seleccionar alguno de los subtemas mostrados

Tiempo aproximado	Tiempo usuario 1	Tiempo usuario 2	Tiempo usuario 3	Tiempo usuario 4	Tiempo usuario 5
3.5 min	3 min.	5min.	2.5 min.	3 min.	4 min.

U1	U2	U3	U4	U5	
X		X	X	X	Exitosa
	X				Con problemas menores
					Con problemas mayores
					No la terminó

Seleccionar alguno de los subtemas de *Creación y uso de clases.*
- ✓ Navegar en la información para el subtema seleccionado hasta encontrar un botón para cargar una animación
- ✓ Ejecutar la animación
- ✓ Volver a ejecutar la animación. Cerrar la ventana al finalizar

Tiempo aproximado	Tiempo usuario 1	Tiempo usuario 2	Tiempo usuario 3	Tiempo usuario 4	Tiempo usuario 5
12.5 min	9 min.	15 min.	14 min	12 min.	12 min.

U1	U2	U3	U4	U5	
X		X	X	X	Exitosa
	X				Con problemas menores
					Con problemas mayores
					No la terminó

Seleccionar el subtema disponible del tema *Objetos como atributos*

- ✓ Navegar dentro de la información disponible hasta llegar a la estructura de la clase *Linea* y ejecutar todas las animaciones disponibles.

Tiempo aproximado	Tiempo usuario 1	Tiempo usuario 2	Tiempo usuario 3	Tiempo usuario 4	Tiempo usuario 5
12 min	15 min.	10 min.	12 min.	11 min.	11 min.

U1	U2	U3	U4	U5	
X	X	X	X	X	Exitosa
					Con problemas menores
					Con problemas mayores
					No la terminó

Navegar en los subtemas contenidos en el tema *Herencia*

Tiempo aproximado	Tiempo usuario 1	Tiempo usuario 2	Tiempo usuario 3	Tiempo usuario 4	Tiempo usuario 5
5 min.	5 min.	5 min.	6 min.	5 min.	5 min.

U1	U2	U3	U4	U5	
X	X	X	X	X	Exitosa
					Con problemas menores
					Con problemas mayores
					No la terminó

Cargar el entorno de programación

Tiempo aproximado	Tiempo usuario 1	Tiempo usuario 2	Tiempo usuario 3	Tiempo usuario 4	Tiempo usuario 5
1 min.	1 min.	2 min.	1.5 min.	1 min.	1 min.

U1	U2	U3	U4	U5	
X	X	X	X	X	Exitosa
					Con problemas menores
					Con problemas mayores
					No la terminó

RESULTADOS DEL CUESTIONARIO DE SALIDA

1. En general, me resultó sencillo usar el sistema de bolsa de trabajo.

Completamente de acuerdo	***

De acuerdo	**
Ni de acuerdo ni en desacuerdo	
En desacuerdo	
Completamente en desacuerdo	

2. **El sistema de software me pareció un método efectivo para apoyar la enseñanza de la programación orientada a objetos.**

Completamente de acuerdo	***
De acuerdo	**
Ni de acuerdo ni en desacuerdo	
En desacuerdo	
Completamente en desacuerdo	

3. **Los siguientes aspectos del sistema de software para apoyar la enseñanza de la POO, me parecieron fáciles de usar (Por favor lista de 1 - 3 aspectos).**

	Comentarios
U1	La ubicación de temas y subtemas.
	El avance dentro de la información.
	La visualización de las animaciones en los ejemplos.
U2	La información estaba organizada.
	Había ejemplos.
	La información era breve.
U3	Ventanas claras.
	Imágenes explicativas.
	Menús adecuados.
U4	Animaciones con explicación
	Disposición de apuntes sin necesidad de apuntar.
	Retroalimentación inmediata por medio de ejemplos.
U5	Las animaciones.
	La interfaz.
	Los gráficos.

4. **Los siguientes aspectos del sistema de software para apoyar la enseñanza de la POO, me parecieron difíciles de usar (Por favor lista de 1 - 3 aspectos).**

	Comentarios
U1	La ubicación de un botón que como tal dijera “animación”.
	El área de visualización requiere ser un poco más amplia.
	-
U2	El cursor parpadeando.

	Comentarios
	La posición de las ventanas donde se despliegan los ejemplos.
	La obstrucción del menú del tema de Herencia por un conjunto de botones.
U3	Confusión en los botones de player.
	Los enlaces de las imágenes un poco lentos
	El botón de prueba me gustaría de otro color ya que no lo ubiqué a primera vista.
U4	El tamaño de la pantalla (es necesario adaptarla sin el monitor es pequeño)
	Falta un mensaje que avise que es necesario permitir pop-up's de lo contrario no se pueden observar bien las pantallas
U5	La aplicación ejecutada posterior al clic en botón cargar entorno.
	Los menús desplegables.

5. **La interfaz del sistema de software me pareció un método sencillo para desplegar información relevante sobre algún concepto particular.**

Completamente de acuerdo	**
De acuerdo	***
Ni de acuerdo ni en desacuerdo	
En desacuerdo	
Completamente en desacuerdo	

6. **Usando la siguiente columna de intervalos, encierra el número más cercano a lo que sientas respecto a la característica del sistema de software.**

	3	2	1	0	1	2	3	
Simple	**	*	*			*		Complejo
Tecnología alta		***	*		*			Tecnología baja
Confiable	**	*	**					No confiable
Fácil de usar	**	*	**					Difícil de usar
Amigable	***	*		*				No amigable
Profesional	**	*	*	*				No profesional
Seguro	**	**	*					No seguro
Durable	**	**		*				Frágil
Atractivo	**	***						No atractivo
Alta calidad	*	***	*					Baja calidad
Me gusta	**	**	*					No me gusta

7. **Por favor selecciona la opción que más refleje tu opinión sobre el sistema.**
   a. Las tareas fueron fáciles de realizar.

Completamente de acuerdo	**

De acuerdo	***
Ni de acuerdo ni en desacuerdo	
En desacuerdo	
Completamente en desacuerdo	

b. La información en las pantallas está organizada de manera óptima

Completamente de acuerdo	**
De acuerdo	**
Ni de acuerdo ni en desacuerdo	*
En desacuerdo	
Completamente en desacuerdo	

c. La cantidad de pasos para realizar una tarea es la correcta.

Completamente de acuerdo	***
De acuerdo	*
Ni de acuerdo ni en desacuerdo	*
En desacuerdo	
Completamente en desacuerdo	

d. La terminología fue clara y precisa.

Completamente de acuerdo	**
De acuerdo	***
Ni de acuerdo ni en desacuerdo	
En desacuerdo	
Completamente en desacuerdo	

e. Siempre pude realizar la operación que necesité.

Completamente de acuerdo	**
De acuerdo	
Ni de acuerdo ni en desacuerdo	**
En desacuerdo	*
Completamente en desacuerdo	

f. La cantidad de información en la pantalla fue adecuada para desarrollar la tarea.

Completamente de acuerdo	***
De acuerdo	*
Ni de acuerdo ni en desacuerdo	

En desacuerdo	*
Completamente en desacuerdo	

8. **Por favor agrega cualquier comentario que creas que puede ayudarnos a mejorar el sistema de software. Agradeceremos tus opiniones específicamente en cuanto a:**
   - ✓ *Funciones que son esenciales o superfluas para la tarea.*
   - ✓ *Características que te gustaría ver en productos futuros.*

	Comentarios
U1	Aumentar el área de visualización.
	Posiblemente sería bueno agregar una pequeña ventana a un costado de la aplicación en la que aparezca una descripción breve de los que sucederá al oprimir algún botón o al pasar sobre áreas específicas de código, que requieran descripción.
U2	Que se pueda copiar la información.
	Que se puedan ejecutar el código de los ejemplos animados.
U3	Quizás una función de menú general por si buscas un subtema específico.
	Me gustaría ver un poco más de imágenes o un poco más de diseño para que además de funcional sea 100% agradable a la vista.
U4	Es necesario especificar que los botones de navegación sirven para un subtema y que no ayudan a moverse entre subtemas. Podría hacerse enviando un mensaje de fin de subtema.
U5	En los menús desplegables si selecciona la sección indicada únicamente se muestra la portada de dicho tema pero no da opción a con los botones > < empezar a recorrer los subtemas de la misma, de igual forma al pasar de un subtema a otro podría darse esa opción también.
	En el subtema 4.11 se da clic y manda al subtema 2.8
	Posiblemente sería también conveniente reforzar las animaciones de los ejemplos con algunos efectos de sonido para despertar el interés del usuario.

6.4 Reporte de hallazgos

Una vez que se han revisado los resultados, tanto de los usuarios a los que se les aplicaron las pruebas de usabilidad, así como de los alumnos a los que se les impartió clase con esta herramienta, y analizados los videos, se procede a generar

el reporte de hallazgos, en este reporte se plasman los errores que se detectaron durante las prueba, así como las áreas de oportunidad (mejoras) del sistema. Se utilizó la siguiente categorización para los hallazgos:

- ✓ **Mayor**. Se considera que es un error que no permite cumplir el objetivo de la funcionalidad implementada.
- ✓ **Menor.** Se considera como un error que no interfiere con el objetivo de la funcionalidad implementada, sin embargo, puede hacer que el usuario la perciba erróneamente.
- ✓ **Mejora.** Son sugerencias para que el sistema cumpla su función mejor.

No.	Funcionalidad en la que se encontró	Descripción	Tipo	Propuesta de Solución
1	Ventanas donde se despliegan las animaciones	La ventana se posiciona aproximadamente a la mitad de pantalla y hay que desplazarla para poder ver algunas opciones de control o incluso la animación completa.	Menor	En los botones que despliegan las animaciones o ejemplos se añadirá el código necesario para lograr que la ventana aparezca en una posición que permita la correcta visualización.
2	Botones	En ocasiones los usuarios necesitan que se les brinde una descripción de la operación que realizará el botón a manera de *tooltip*.	Menor	Aunque no interfiere con las operaciones, siempre que se sea necesario se añadirán los mensajes a algunos botones.
3	Área de despliegue de información	A los usuarios les gustaría poder copiar la información que se despliega en las dispositivas	Mejora	Se buscará algún mecanismo que permita la distribución del material presentado.
4	Menú del tema 4: *Herencia*	Cuando se despliega el menú, si se encuentra algún otro objeto como una animación o un grupo de botones, el menú queda bloqueado por estos elementos y no	Mayor	Se modificará el orden de apilamiento de las capas para obligar a que el menú quede por encima de cualquier objeto colocado en el diseño.

No.	Funcionalidad en la que se encontró	Descripción	Tipo	Propuesta de Solución
		se aprecian bien sus opciones.		
5	Animaciones	Velocidad: algunas animaciones se reproducen muy rápido	Mayor	Se modificará la velocidad de reproducción de las animaciones, en función de la animación presentada.
6	Animaciones	Las animaciones se reproducen de manera autónoma y en ocasiones se requiere que sean controladas por el usuario.	Menor	Se evaluarán las animaciones y se añadirán controles de reproducción a aquellas animaciones que lo requieran.
7	Encabezado	Color verde muy vistoso	Menor	Se buscará una alternativa de color.
8	Encabezado	Las animaciones que se tienen en el encabezado de la página pueden llegar a distraer.	Menor	Se buscará hacer que las animaciones que se diseñaron no se conviertan en un distractor muy fuerte.
9	Botones de navegación en los subtemas	Siempre que se termina un subtema es necesario seleccionar el siguiente subtema desde el menú principal. La navegación no es fluida entre temas.	Mejora	Se habilitaran los controles de navegación en las dispositivas para que permitan el avanzar al siguiente subtema o retroceder al anterior subtema.
10	Diapositivas	El área donde se despliega la información carece de un título que la identifique.	Mejora	Se buscará añadir un elemento que le indique en todo momento al usuario en que sección se encuentra posicionado.
11	Interfaz general	Hacen falta más imágenes a lo largo de todo el sistema para hacer la interfaz más agradable.	Mejora	Se buscará insertar más imágenes en la aplicación.
12	Tema: *Creación y uso de clases*	Hace falta una descripción más clara de los métodos *equals* y	Mayor	Se buscará ampliar la información que se presenta para explicar

No.	Funcionalidad en la que se encontró	Descripción	Tipo	Propuesta de Solución
		toString. Hacen falta ejemplos más claros.		estos dos métodos y se buscará además incorporar algún ejemplo representativo.
13	Tema: *Creación y uso de clases*	Inconsistencia en la información que se presenta en la clase *Hora*	Mayor	Se hará consistente la información que se presenta en las clases para evitar confundir a los alumnos.
14	Tema: *Creación y uso de objetos*	Sería conveniente que se mostraran más ejemplos sobre la creación de objetos.	Mejora	Se añadirá algún ejemplo adicional al que se presenta actualmente (Moneda).
15	Tema: *Creación y uso de objetos*	Hay algunos botones en los ejemplos animados que no despliegan la información que se supone deberían de mostrar.	Mayor	Se hará una revisión completa de las animaciones que se presentan en este capítulo para habilitar los botones que hacen falta.
16	Ejemplos	Sería conveniente que se mostraran siempre que se pudiera la codificación de los ejemplos que se muestran.	Mejora	Se buscará añadir siempre que se puedan las codificaciones, con el fin de brindar más claridad.

Errores Mayores: 5 Errores Menores: 5 Mejoras: 6

6.5 Análisis de criterios ergonómicos

En este caso, sólo se hará un análisis de aquellos criterios ergonómicos tomados en cuenta para el diseño general de la interfaz de usuario [1].

Criterio	Descripción	Subcriterio	Sistema
Guía	Medios disponibles para orientar al usuario a través de su interacción con la	Agrupación	El sistema tiene una agrupación lógica, tanto en las opciones que despliega así como en la información que muestra.
		Retroalimentación Inmediata	Aunque no propiamente el sistema, el navegador despliega un mensaje cuando no alguna página no se

Criterio	Descripción	Subcriterio	Sistema
	computadora		encuentra disponible. Por otro lado el entorno de programación que se incorpora da la retroalimentación inmediata al usuario cuando se compila un programa.
		Legibilidad	La información que el usuario ve en pantalla tiene un adecuado tamaño de letra, las dispositivas que se incorporan no sufren ningún cambio puesto que FlashPaper mantiene el formato original y el tamaño permanece constante de manera que el usuario no tenga problemas al leerla. No se utilizaron combinaciones de colores en el texto, para no interferir con la legibilidad de la misma. Se cuidó la utilización de fondos en las páginas del sistema, y sólo se utilizaron algunos colores para el diseño de encabezados, menú y botones.

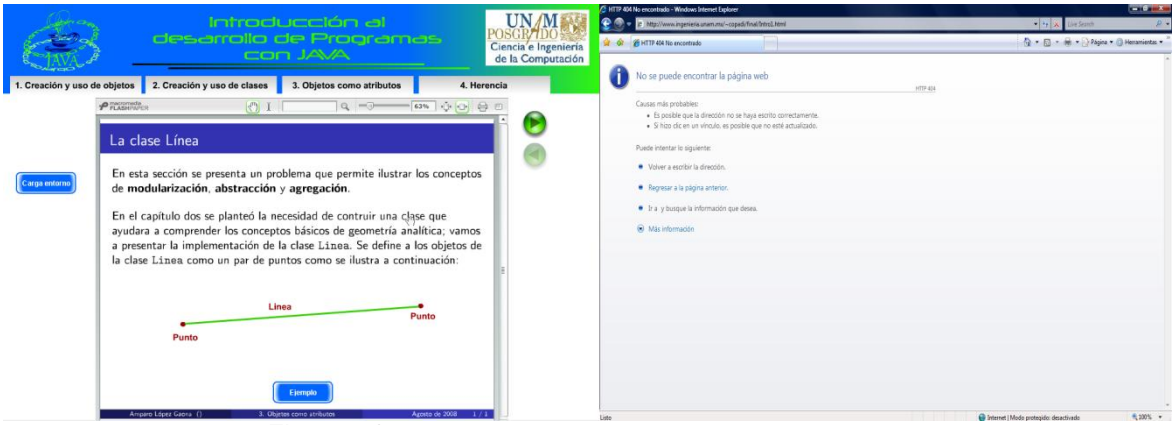


Figura 79. Ejemplos del criterio de *Guía* en el sistema

Criterio	Descripción	Subcriterio	Sistema

Carga de trabajo	Elementos de la interfaz que juegan un rol en la percepción del usuario	Brevedad	Se buscó que los botones que ejecutan las acciones particulares como cargar ejemplos o animaciones tuvieran textos breves, pero que le dieran la noción al usuario de que operación realizan. En el caso de los botones del menú, aunque los textos son más largos tiene la ventaja de que el usuario puede saber que información desplegará cada una de estas opciones, sólo por el nombre que se colocó en el menú.
		Densidad de Información	Al emplear el diseño de las dispositivas en combinación con la aplicación se procura que éstas presenten la menor cantidad de información. Esto se debe a que siempre aparecen con el mismo tamaño y si se sobrecargan de información se corre el riesgo de que la letra no sea legible.

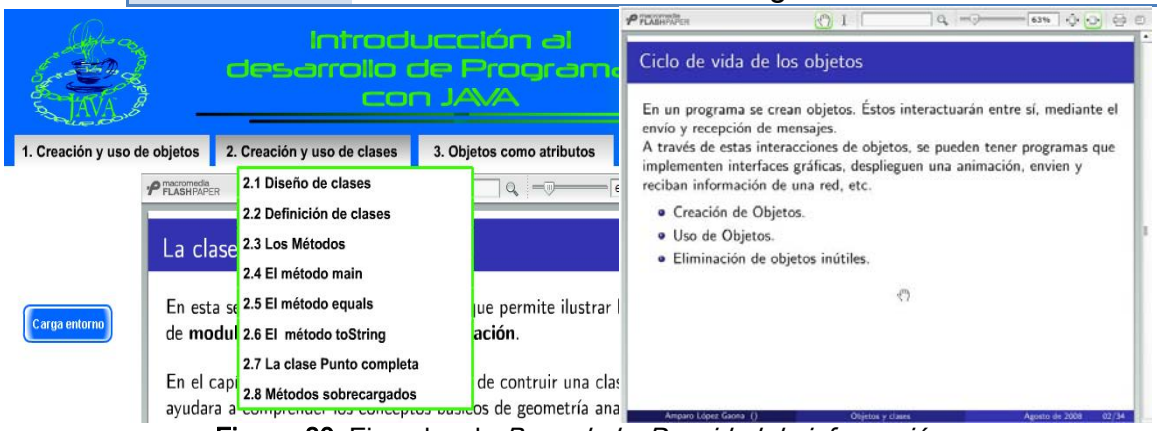


Figura 80. Ejemplos de *Brevedad* y *Densidad de información*

Criterio	Descripción	Subcriterio	Sistema
Control Explícto	Saber qué es lo que está haciendo el sistema.	Acciones explícitas del usuario.	El sistema está diseñado para que el usuario sepa en todo momento donde se encuentra, la utilización del menú principal es de gran ayuda pues le da la libertad de moverse a cualquier sección del sistema.
		Control de usuario	El usuario tiene pleno control sobre el sistema pues todas las opciones disponibles son susceptibles de cancelación. El navegar de una sección a otra es básicamente con un solo clic.

Criterio	Descripción	Subcriterio	Sistema
Consistencia	El diseño de la interfaz se mantiene para contextos similares y se diferencia para contextos diferentes		El sistema está diseñado para que presente interfaces con ligeros cambios, dependiendo del contexto en el que se encuentre, por ejemplo las pantallas de presentación de cada subtema no tienen habilitados los controles que tienen las interfaces para algún subtema particular. Por otra parte, siempre que no se pueda avanzar o retroceder, los botones de navegación se encuentran deshabilitados.

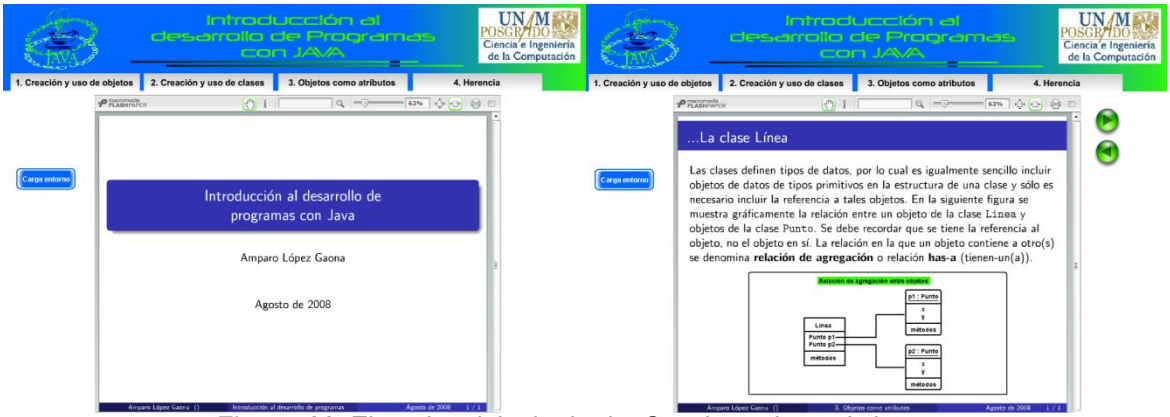


Figura 80. Ejemplos del criterio de *Consistencia* en el sistema

Hasta el momento, la aplicación se ha usado en un curso solamente utilizando una laptop o computadora y un videoproyector; sin embargo aún no se han hecho las pruebas pertinentes sobre su funcionamiento y desempeño en un pizarrón electrónico, por tal motivo, ya se programó para enero del 2009 un curso de 20 horas en la Facultad de Ingeniería en donde se planea utilizar el sistema en conjunto con las características que brindan los pizarrones electrónicos e impartir un curso de introducción a la POO a alumnos de dicha Facultad y se espera poder reportar los resultados de esta prueba en un anexo a este documento. Cabe destacar que independientemente de que no se han hecho estas pruebas, el sistema implementa las características suficientes para que el profesor pueda interactuar con esta herramienta de apoyo a la enseñanza por medio de un pizarrón electrónico.

CONCLUSIONES

Conclusiones

El avance que ha tenido la tecnología en los últimos años ha sido impresionante, hoy en día es posible tener equipos de cómputo muy potentes en los escritorios, equipos que hace unos años era difícil imaginar; en la actualidad el cómputo móvil es ya una realidad, cada vez es más común ver en los pasillos de las escuelas a los estudiantes con laptops con capacidad para conectarse a un número cada vez más creciente de redes inalámbricas, incrementando con ello las posibilidades de comunicación e intercambio de información. Según la Asociación Mexicana de Internet (AMIPCI) en su estudio anual sobre "Hábitos de los usuarios de Internet en México", reveló que durante el 2007 de los 23.7 millones de internautas que se tienen identificados, 19.9 millones de ellos tienen 13 años en adelante; además, hay 14.8 millones de computadoras personales en México, de las cuales, el 59% están conectadas a Internet [49]. Estos datos nos hablan de que a pesar que existen muchas condiciones de rezago y desigualdad en el país, el acercamiento que tienen los jóvenes mexicanos a la tecnología es importante, sin embargo, esta facilidad de tener acceso a la tecnología no siempre se ve reflejada en los salones de clase.

En los últimos años se ha venido experimentando una reducción importante en los costos de los equipos videoproyectores y aunque los proyectores de acetatos han sido reemplazados por una laptop y un cañón, la función sigue siendo la misma, sólo que el acetato se ha convertido en una diapositiva. Las instituciones de educación superior han tomado consciencia de la importancia que está cobrando la tecnología como una herramienta que puede apoyar la enseñanza; diversas Facultades de la Universidad Nacional Autónoma de México han hecho importantes esfuerzos en este rubro y ya es posible encontrar aulas acondicionadas con computadoras de escritorio, videoproyectores y pizarrones electrónicos, sin embargo, a pesar de estos importantes esfuerzos es necesario destacar que aún hay mucho trabajo por hacer, pues las aulas con estas características son contadas. Una gran desventaja de este tipo de tecnologías es que el software que permite la comunicación entre el pizarrón electrónico y la

computadora es particular para cada tipo de pizarrón y depende completamente del sistema operativo en el que se instale, lo cual impide que cualquier persona que cuente con una computadora pueda conectarse directamente a estos dispositivos. Una alternativa que desde mi punto de vista parece viable es el uso de las TabletPC, ya que se trata de computadoras que incorporan las capacidades de un pizarrón electrónico pero a un precio más bajo, pues sólo basta con conectarla a un videoproyector (para lo cual no se necesita un software particular) para tener un funcionamiento similar.

Ahora bien, como se ha podido observar a lo largo de este documento, se han hecho esfuerzos importantes para incorporar estas herramientas tecnológicas al ámbito educativo. Un primer esfuerzo lo podemos ver en el proyecto Enciclomedia, un esfuerzo importante del Gobierno Federal por incorporar la tecnología al salón de clases, pero que no tuvo el éxito que se tenía proyectado, las causas son muy variadas, un aspecto importante es que la gran mayoría de los profesores no tienen la iniciativa de aprender una nueva tecnología. Esto en gran parte se debe a que no se les brinda la capacitación suficiente, de manera que son pocos los casos en que se aprovechan estos recursos al máximo, por el contrario, en muchas ocasiones se les deja a los profesores con los conocimientos mínimos para que puedan ocuparla pero no explotarla; aunado a esto, son poco los profesores que tienen el interés de aprender los nuevos elementos que incorporan este tipo de tecnologías, ya que no solo basta con ocuparlas, significa aprender el software necesario para poder crear material que pueda aprovechar al máximo estas herramientas, esto se convierte en un obstáculo importante, ya que no se cuentan con los recursos necesarios para una óptima capacitación. De ahí que hoy el principal uso que se les da a estas herramientas sea sólo la proyección de material a los alumnos dentro del salón de clases.

Como se puede observar, es innegable la amplia gama de posibilidades y facilidades que estas herramientas brindan y más cuando las trasladamos al ámbito de los cursos de computación que se imparten en muchas universidades del país. Uno de esos cursos básicos y que es precisamente el tema central de

este trabajo es el que se refiere a los cursos de programación. Hoy en día, debido a las necesidades que se tienen en el mercado en cuanto al desarrollo de productos de software de calidad reutilizables, se requiere de un lenguaje que sea orientado a objetos, distribuido, dinámico, robusto, seguro, multitarea y portable; y son precisamente estás las características que han hecho de Java el lenguaje idóneo para incorporarse en los cursos básicos de programación en muchas instituciones de educación superior. Sin embargo, como se ha podido ver, lejos de lo que muchos opinan, Java no es un lenguaje fácil de enseñar, por el contrario, el lenguaje involucra un paradigma que muy pocas veces es explicado en los libros de texto y que en muchas ocasiones es enseñado desde el enfoque procedural. El enfoque orientado a objetos encuentra grandes obstáculos en la enseñanza, uno de los principales es que son pocos los profesores que lo entienden y lo enseñan como tal y por otro lado, como ya se ha visto, existen multitud de enfoques referentes a la forma cómo enseñar este lenguaje a los alumnos.

Una de las principales complicaciones, es que muchas veces se les enseña a "programar en orientación a objetos" pero sin asegurarse de que los alumnos entiendan los conceptos que este paradigma involucra; esto se debe a que, al tratarse de un lenguaje que maneja un alto nivel de abstracción es en este mismo nivel donde se trabajan todos los conceptos; de manera que se da por hecho que los alumnos comprenden los conceptos de clase, objeto y mensaje y se pasa directamente a al proceso de compilación, para mostrarles cómo reacciona el compilador ante distintos escenarios de error y la mejor forma de corregirlos; se dedica una buena cantidad de horas a enseñarles la sintaxis del lenguaje considerando que ésta es la forma en que el alumnos puede aprender el lenguaje y después mostrando la ejecución de los programas, dejando como único incentivo el hecho de ver que un programa se ejecuta de acuerdo a los requerimientos solicitados. Incluso se busca siempre que el alumno se encuentre frente a una computadora para que pueda estar practicando y así adquirir esa experiencia en el planteamiento e implementación de soluciones; sin embargo, considero que no es un enfoque que funcione al 100%, pues a pesar de estos esfuerzos, se sigue teniendo a bastantes alumnos reprobados y frustrados ante el

hecho de no comprender los conceptos que se les presentan. La mayoría de las veces, el problema no es que no estudien o no practiquen la programación, el principal obstáculo es que se les dificulta pensar en las clases y objetos, tal y como se piensa en la vida cotidiana, pues no es un enfoque al que estén acostumbrados.

El objetivo de este trabajo fue precisamente el desarrollar un sistema de software que les facilitara a los alumnos el proceso de acostumbrarlos a pensar en objetos, clases y mensajes; y que además, bajara el nivel de abstracción de estos conceptos, para que ellos vieran cómo es que se crean estas *fábricas de objetos*, cómo es que se crean los *objetos* y cómo interactúan entre ellos a partir del paso de *mensajes* y que a través de una herramienta visual pudieran ver una posible representación gráfica que les ayudara a formarse una idea y entender los conceptos involucrados. Además de esto, se buscó que fuera un sistema que pudiera utilizar la tecnología como una herramienta para apoyar la enseñanza de la POO, es decir, una herramienta visual que pudiera explotar las características de los pizarrones electrónicos e incluso de las TabletPC para conformar una aplicación que pudiera presentar la teoría de la POO en combinación con un conjunto de ejemplos y programas a los que se doto de animación y riqueza interactiva, claro, sin dejar de lado la parte de compilación y ejecución de programas, de manera que se tenga un enfoque más integral de enseñanza.

Como se ha visto a lo largo del documento, se tienen en el mercado una buena cantidad de herramientas visuales para apoyar la enseñanza de la POO, sin embargo, ninguna presenta una metodología estructurada para el desarrollo de programas; por el contario, la filosofía que manejan es simplemente el construir programas a través de elementos gráficos, lo cual a la larga implica que el profesor y el alumno tengan, no sólo que aprender el paradigma orientado a objetos, sino que aprendan a utilizar la herramienta gráfica, con lo cual se incrementa la curva de aprendizaje, al ser más elementos los que se tienen que manejar. Por otro lado, considero que no es un enfoque que sea exitoso en la mayoría de los casos, pues de esta manera se acostumbra a los alumnos a

quedarse con una sola herramienta (sería difícil el apoyarse en varias herramientas en un solo curso), lo cual les dificulta que puedan desarrollar sus programas por sí mismos, pues si en algún momento dejan de tener la herramienta, ya no saben qué hacer (caso similar a lo que sucede cuando los alumnos optan por una IDE de desarrollo). Con esto no quiero decir que esta herramienta es superior a las herramientas que están disponibles, pues eso depende de los gustos y necesidades de cada quien, lo que si se busca es ofrecer una alternativa más e introducir un nuevo enfoque y una nueva forma de enseñar la POO.

La herramienta que se detalla en este trabajo se desarrolló de manera que el alumno conozca todos los elementos que implica el desarrollo de programas en Java y sólo utiliza los programas ejemplos animados como un apoyo visual en la comprensión de los conceptos principales de la POO; al final el alumno sigue escribiendo, compilando y ejecutando sus programas sólo utilizando los comandos del JDK con el que se cuente.

Por otro lado, se creó una aplicación que puede verse en cualquier parte y puede ser utilizada por cualquier persona, sin embargo, se mantiene el enfoque incial, es decir, un apoyo para los profesores interesados en enseñar POO; de manera que se cuenta con una aplicación Web que sólo requiere de un *plug-in* para poder visualizar las animaciones (Flash player) y un JDK instalado y configurado en la máquina cliente. Para su visualización no se requiere más que algún navegador de Internet y de esta forma también se puede ofrecer una herramienta independiente de la plataforma en que se trabaje.

Un aspecto que ha enriquecido este trabajo, es el hecho de ya haberlo utilizado en una clase formal en la Facultad de Ciencias, los comentarios de los alumnos al respecto fueron bastante constructivos; en términos generales podemos decir que se mostraron contentos con el hecho de contar con una herramienta que se preocupe porque ellos comprendan mejor los temas de la POO. Se mostraron satisfechos con la cantidad y calidad de la información, uno de los aspectos que más les agrado, fue el hecho de contar con un conjunto de ejemplos y programas

completamente animados pues esto les permitió resolver dudas que tenían y que a veces no se atrevían a preguntar. También se mostraron críticos y dispuestos a colaborar, pues siempre hacían ver los errores, los aspectos que no les agradaban y aquellos que podían mejorar e incluso lo que les gustaría tener para futuras versiones. El trabajo con el grupo de Programación 1 permitió descubrir muchas áreas de oportunidad a corto y mediano plazo. Sin duda alguna este es un medio eficaz que permite saber cómo vamos y a donde podemos llegar, todo siguiendo un diseño centrado completamente en el usuario y eso es algo que les gusta a los alumnos, el saber que se toman en cuenta sus inquietudes. Por esta razón es que ya se programó un curso intersemestral en enero de 2009, en este curso se trabajará ya en un salón que tenga instalado un pizarrón electrónico donde seguramente se descubrirán nuevas áreas de oportunidad y nuevos comentarios; también se utilizará en un curso de para profesores que se impartirá en Facultad de Ciencias, además de seguirla ocupando como herramienta en el salón de clases y en este caso se utilizará en la materia de ICC1, también de Facultad de Ciencias. La retroalimentación que seguramente se recibirá en estos foros permitirá que se tenga un proceso de mejora continua y al mismo tiempo brinda los elementos necesarios para hacer un mantenimiento más efectivo (se sigue un proceso iterativo incremental).

Trabajos futuros

1. Incorporar, previa evaluación, los comentarios que se hagan al sistema en los diversos foros donde se presente esta herramienta de software.
2. Se planea continuar incluyendo algunos conceptos más de la POO (por ejemplo: *agrupación de objetos, clases abstractas e interfaces,*etc.), ya que en esta primera versión sólo se tomaron en cuenta los cuatro más representativos: *objetos, clases, agregación* y *herencia.*
3. Incorporar información relacionada con el manejo de las versiones y su importancia.
4. Buscar un congreso de educación para someter este trabajo en forma de artículo.
5. Extender esta idea de construir una herramienta similar para otros temas de programación como pueden ser las estructuras de datos.
6. Capacitar a algunos alumnos interesados en la creación de animaciones para que de esta forma el trabajo se vaya enriqueciendo con otros enfoques para el planteamiento de las animaciones.
7. Desarrollar módulos para otros temas que involucren el lenguaje de programación Java: servlets, hilos, conexión a bases de datos, interfaz gráfica, etc.
8. Trabajar con expertos en el área de pedagogía, esto con el fin de lograr un mejor ambiente de enseñanza que ayude a aprovechar de manera más óptima tanto del sistema y los recursos del aula.
9. Efectuar pruebas de usabilidad con los profesores.

Reflexión final

Solo me resta decir que este fue un trabajo que me encantó desarrollar, en donde puse a prueba los conocimientos obtenidos a lo largo de estos dos años en la Maestría y por supuesto, fue una excelente oportunidad para aprender nuevos elementos del diseño de interfaces, del diseño de animaciones y comprender mucho mejor los conceptos relacionados con la POO, también me sirvió para aplicar de una manera más formal, los conceptos vistos en mis clases de la

maestría. Este trabajo me llevó a la Facultad de Ciencias, donde me enseñaron una nueva forma de trabajar y de dar clases, muy diferente a la que había trabajado en la Facultad de Ingeniería; sin duda alguna, me llevo grandes lecciones aprendidas.

A lo largo de este estudio comprendí la gran responsabilidad que se tienen al pararse frente al pizarrón y dictar una clase de programación orientada a objetos, me siento muy agradecido y muy orgulloso de trabajar con una maestra en toda la extensión de la palabra, una investigadora que siempre está innovando, buscando nuevas formas de impartir clase, sin duda alguna comprometida con la educación de calidad y a quién debo mucho de lo que aquí está plasmado.

Disfrute mucho la elaboración de este trabajo de Maestría, espero que sirva para futuros desarrollos, aún hay mucho por hacer, pues la educación es dinámica y cada nuevo grupo de alumnos impone nuevos retos, nuevos tropiezos y sobre todo grandes satisfacciones. Por mi parte es todo y muchas gracias a mis Facultades, al Posgrado, a mi querida Universidad por todo lo que me han brindado. Estoy nuevamente en deuda y mi pago será a través de acciones de bien, tanto en mi devenir profesional, como en mi formación humana.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

[1] Bastien, C. y Scapin, D., "Ergonomic criteria for the evaluation of Human Computer interfaces (v. 2.1), *Technical report no. 165, INRIA,* 1993.

[2] Ben-Bassat, R., Ben-Ari, M., Uronen, P., "An Extended Experiment with Jeliot 200", *Program Visualization Workshop, University of Joensuu,* 2000.

[3] Bennedsen, J., Caspersen, M., "Teaching Object – Oriented Programming – Towards Teaching a Systematic Programming Process".

[4] Bergin, J., Bruce, K., Kölling, M., "Objects-early tools: a demonstration", *ACM SIGCSE Technical Symposium on Computer Science Education,* 390 – 391, 2005.

[5] Berón, M. et al., "Herramientas para la comprensión de programas", **http://ficcte.unimoron.edu.ar/wicc/Trabajos/III%20-%20isbd/646-CARWICC2006.pdf** (16/10/2008).

[6] Catalán, M., "Metodologías de evaluación de las interfaces gráficas de usuario", **http://eprints.rclis.org/archive/00004718/01/Metodologias_de_evaluaci%C3%B3n_de_interfaces_graficas_de_usuario.pdf** (10/11/2008).

[7] Coad, P. y Mayfield, M., "Java Design", *Prentice-Hall*, 1997.

[8] Cooper, S., Dann, W., Pausch, R., "Alice: a 3-D tool for introductory programming concepts", *Preceedings of the fifth annual CCSC northeastern conference on the Journal of computing in small colleges,* 107 – 116, 2000.

[9] Cooper, S., Dann, W., Pausch, R., "Teaching objects-first in introductory computer science", *ACM SIGCSE Bulletin,* 35(1): 191 – 195, 2003.

[10] Curbelo, A., "Uso de las Tablet PC en la Educación: Una Herramienta de Cambio", **http://ined.sagrado.edu/webedu/WebEdu05/Aury%20Curbelo.swf** (26/10/2008).

[11] English, B., Rainwater, S., "The effectiveness of animations in an undergraduate operating systems course", *Journal of Computing Sciences in Colleges,* 21(5): 53 – 59, 2006.

[12] French, J., "Beyond the tablet PC: using the tablet PC in a collaborative learning environment", *Journal of Computing Sciences in Colleges,* 23(2): 84 – 89, 2007.

[13] García, M., "¿Es conveniente la orientación a objetos en un primer curso de programación?", **http://bioinfo.uib.es/~joemiro/aenui/procJenui/ProcWeb/actas2001/gaesc43.pdf** (16/10/2008).

[14] Gayo, D. et al., "Reflexiones y experiencias sobre la enseñanza de POO como único paradigma", **http://www.di.uniovi.es/~dani/publications/jenui03.pdf** (16/10/2008).

[15] Gross, P., Powers, K., "Evaluating assessments of novice programming environments", *ACM SIGCSE International Computing Education Research Workshop,* 99 – 110, 2005.

[16] Haajanen, J. et al., "Animation of User Algorithms on the Web", *IEEE Symposium on Visual Languages*, 1997.

[17] Kamin, S. et al., "A system for developing tablet pc applications for education", *ACM SIGCSE Bulletin,* 40(1): 422 – 426, 2008.

[18] Kennewell, S., Morgan, A., "Student teachers' experiences and attitudes towards using interactive whiteboards in the teaching and learning of

young children", *ACM International Conference Proceeding Series; Vol. 98",* 65 – 69, 2003.

[19] Kölling, M., "The Problem of Teaching Object – Oriented Programming, Part I: Languages", *Journal of Object – Oriented Programming,* 11(8): 8 – 15, 1999.

[20] Kölling, M., "The Problem of Teaching Object – Oriented Programming, Part II: Environments", *Journal of Object – Oriented Programming,* 11(9): 6– 12, 1999.

[21] Kölling, M., Rosenberg, J., "Objects first with Java an BlueJ", *ACM SIGCSE Technical Symposium on Computer Science Education,* 429, 2000

[22] Kölling, M., Rosenberg, J., "Guidelines for Teaching Orientation with Java", *ACM SIGCSE Bulletin,* 33(3): 33 – 36, 2001.

[23] Kouznetsova, S., "Using BlueJ and Blackjack to teach object-oriented design concepts in CS1", *Journal of Computing Sciences in Colleges,* 22(4): 49 – 55, 2007.

[24] Leutenegger, S., Edgington, J., "A games first to teaching introductory programming", *ACM SIGCSE Technical Symposium on Computer Science Education,* 115 – 118, 2007.

[25] Li, Q., et al., "Technology supports for distributed and collaborative learning over the Internet", *ACM Transactions on Internet Technology,* 8(2): Artículo 5, 2008.

[26] Lindskov, J., Lehrmann, O., "Using Object – Orientation as a Common Basis for System Development Education", *ACM SIGPLAN Notices,* 31(12), 1996.

[27] Lorenzen, T., Sattar, A., "Objects first using Alice to introduce object constructs in CS1", *ACM SIGCSE Bulletin,* 40(2): 62 – 64, 2008.

[28] Machorro, R., "¿Cómo aprender el lenguaje Java a través de una aplicación visual", *Primer Encuentro Intrainstitucional de Tutorías, IPN,* Noviembre 2004.

[29] Macías, J., "Utilización de prácticas con gráficos 3D animados en la enseñanza de la programación orientada a objetos", **http://www.cc.uah.es/jagm/docs/2002/JENUIGR.pdf** (16/10/2008).

[30] Malinowsky, E., "Eseñanza de Programación Orientada a Objetos para Principiantes", *Ingeniería*, Enero 1999, **http://www.allbusiness.com/information/publishing-industries/399502-1.html** (/ /2008).

[31] Martínez, A., Santillán, R.A., "ENCICLOMEDIA. El uso del pizarrón interactivo", *Secretaría de Educación Pública,* 40 páginas, 2006.

[32] McKim, C., "Teaching object-oriented programming and design", *ACM SIGPLAN OOPS Messenger,* 4(2): 221, 1993.

[33] Moreno, A. et al., "Program animation in jeliot 3", *ACM SIGCSE, ACM SIGCSE Annual Joint Conference ITiCSE,* 265 – 265, 2004.

[34] Naharro-Berrocal, F. et al., "Automatic Generation of Algorithm Animations in a Programming Environment", *ASE/IEEE Frontiers in Education Conference,* 2000.

[35] Naps, T. et al., "Evaluating the educational impact of visualization", *ACM SIGCSE Annual Joint Conference ITiCSE working groups reports,* 124 – 136, 2003.

[36] Ponce, M., "Experiencias del uso del pizarrón electrónico en preparatoria", **http://dewey.uab.es/pmarques/dim/docs/teresaponce.pdf** (26/10/2008).

[37] Röbling, G., Freisleben, B., "Generación de visualizaciones de software usando AnimalScript", *Novática* 150, 2001.

[38] Sajaniem, J., Kuittienen, M., Tikansalo, T., "A study of the development of students' visualizations of program state during an elementary object-oriented programming course", *Journal on Educational Resources in Computing (JERIC),* 7(4): Artículo 3, 2008.

[39] Smith, P., Boyd, G., "Introducing OO concepts from a class user perspective", *Journal of Computing Sciences in Colleges,* 17(2): 152 – 158, 2001.

[40] Timmins, S., "Tablet PC: blackboard to the web", *ACM SIGUCCS User Services Conference,* 296 – 300, 2004.

[41] Thomasson, B., Ratcliffe, M., Thomas, L.; "Identifying Novice Difficulties in Object Oriented Design", *ACM SIGCSE Bulletin* 38(2): 28 – 32, 2006.

[42] Tomov, O., Smrikarov, A., "Exploring environments for creation of e-learning simulators", *ACM International Conference Proceeding Series on Computer Systems and technologies,* 287: Artículo 91, 2007.

[43] Torres, A. et al. "El pizarrón electrónico interactivo, otra tecnología para incorporar a la educación", **http://bibliotecadigital.conevyt.org.mx/colecciones/documentos/somece/02.pdf** (16/10/2008).

[44] Olan, M., "Dr. J vs. the bird: Java IDE's one-on-one", *Journal of Computing Sciences in Colleges,* 19(5): 44 – 52, 2004.

[45] Wikipedia, **http://es.wikipedia.org/wiki/Pizarra_electr%C3%B3nica** (16/10/2008).

[46] Willis, C., Miertschin, L., "Tablet PC's as instructional tools or the pen is mightier than the 'board!", *ACM Proceedings of the Conference On Information Technology Education,* 153 – 159, 2004.

[47] Xinogalos, S., Satratzemi, M., Dagdilelis, V., "Teaching java with BlueJ: a two-year experience", *ACM SIGCSE Annual Joint Conference ITiCSE,* Poster Session: 345, 2007.

[48] Yu, B., y Roh, S., "The effects of menu design on Information-seeking performance and user's attitude on the World Wide Web", *Journal of the American Society for information Science and Techonology,* 53(11):923-933, 2002.

[49] AMIPCI, "Hábitos de los usuarios de Internet en México, 2008", **http://www.amipci.org.mx/temp/pdf-031596700119342674OB.pdf** (10/11/2008).

[50] López, A. (2007) Introducción al desarrollo de programas con Java. Facultad de Ciencias, UNAM. 1ª edición, México. 253 pp.

APÉNDICE

FORMATO DE LAS PRUEBAS DE USABILIDAD

PROTOCOLO DE BIENVENIDA

¡Qué tal! Mi nombre es _____. Estaré trabajando con ustedes en esta sesión de prueba. Déjenme explicarles porque les he pedido que vengan este día.

Estoy aquí para probar lo fácil que es usar este nuevo Sistema de Software, una herramienta para apoyar la enseñanza de la POO y me agradaría contar con su ayuda. Es importante hacer notar que sólo voy a evaluar el software, no voy a calificar sus conocimientos y habilidades; no los estoy evaluando a ustedes.

Ustedes realizarán algunas tareas típicas con esta herramienta y me gustaría que las realicen como normalmente las harían. Por ejemplo, traten de trabajar a la misma velocidad y con la misma atención y detalle con que normalmente las efectuarían. Esta es una prueba para un sistema de software que se encuentra en su primera versión, es completamente funcional, sin embargo, podría no trabajar como ustedes esperan. En cualquier momento podrán hacer las preguntas que deseen, sin embargo, no puedo responderles ninguna, ya que se trata de un estudio sobre el uso una nueva herramienta para apoyar la enseñanza de la POO y me interesa observar cómo se comporta con usuarios normales.

Durante esta sesión les voy a pedir que llenen algunas formas y respondan algunas preguntas. Es importante que contesten con la verdad. Mi único papel aquí es que entre todos descubramos los desperfectos y ventajas de este sistema de software desde su perspectiva. Les pido por favor que no contesten las preguntas basados en lo que ustedes piensan que yo quisiera escuchar; necesito conocer exactamente lo que ustedes piensan y opinan.

Mientras ustedes están trabajando yo estaré sentado cerca tomando algunas notas y tiempos. Les pido por favor que actúen y hablen de forma natural.

¿Tienen alguna pregunta?

Si no, entonces vamos comenzar respondiendo un cuestionario de conocimientos previos. La información obtenida se utilizará sin fines de lucro y es completamente confidencial.

CUESTIONARIO DE ENTRADA

Nombre: _____

Sexo: ___M ___H Edad: _____ Fecha: _____

Por favor responda las preguntas siguientes para ayudarnos a entender sus conocimientos previos y experiencia.

EDUCACIÓN (Por favor marque su grado máximo de estudios)

( ) Bachillerato

( ) Licenciatura Título: _____

( ) Posgrado Área de estudio: _____

COMPUTACIÓN

1. **¿Has usado computadoras personales?**
___ Si ___ No

2. **¿Cuánto tiempo llevas ocupándolas?**_____

3. **En un día normal, con qué frecuencia usas la computadora para desarrollar tu trabajo.**
__ Todo el tiempo __ Frecuentemente __Ocasionalmente __No la ocupo

4. **De la lista siguiente, marca las aplicaciones que has usado:**

Aplicación	SI	NO
Procesadores de palabras (Word…)		
Hojas de cálculo (Excel…)		
Bases de datos (Oracle,…)		
Diseño web (Dreamweaver, Flash…)		
Programación (Java, C# …)		
Otros:		

5. **¿En un día normal, cuánto tiempo navegas en Internet?**
__ Todo el tiempo __ Frecuentemente __Ocasionalmente __ No navego

EXPERIENCIA CON LA PROGRAMACIÓN ORIENTADA A OBJETOS

6. **¿Has programado en alguna ocasión?**
___ Si ___ No

7. **Si lo has hecho, ¿En qué lenguaje has programado? (Puedes marcar más de uno)**
___ Java ___ TurboC ___ C# ___ C++ ___ Pascal
___ Otro: _____

8. **¿Has programado en un lenguaje que maneje orientación a objetos? ¿Cuál?**
___ Si ___ No
Lenguaje: _____

9. **¿Te ha resultado fácil programar en ese lenguaje?**
__ Totalmente __ Frecuentemente __Ocasionalmente __Me ha resultado difícil

10. **¿Has utilizado alguna herramienta visual para ayudarte a comprender mejor los conceptos de la programación orientada a objetos?**
___ Si ___No

11. **Si lo has hecho, ¿Qué herramienta has ocupado? (Puedes marcar más de una)**
__ Jeliot __ BlueJ __Alice __ Otra: _____

12. **¿Crees que la utilización de herramientas visuales ayude a los usuarios con poca o nula experiencia en programación a comprender mejor y más fácil los conceptos? (Brevemente)**

_____

_____

_____

13. **¿Recomendarías el uso de una herramienta visual para apoyar la enseñanza-aprendizaje de la programación orientada a objetos?**
___ Si ___ No

Gracias

LISTA DE TAREAS

Entrar al sistema
Teclear la dirección **www.ingenieria.unam.mx/~copadi/final/Intro.html**

Despliega la interfaz principal para el tema *Creación y uso de objetos*
Describe brevemente la pantalla que te muestra el sistema para realizar esta tarea.

Selecciona alguno de los subtemas mostrados en el menú principal.

Selecciona alguno de los subtemas de *Creación y uso de clases.*

Describe brevemente la pantalla que te muestra el sistema para realizar esta tarea.

Navega en la información para el subtema seleccionado hasta encontrar un botón para cargar una animación.

Observaciones

Ejecuta la animación

Observaciones

Vuelve a ejecutar la animación. Al finalizar cierra la ventana

Selecciona el subtema disponible del tema *Objetos como atributos*

Describe brevemente las pantallas que muestra el sistema para realizar esta tarea.

Navega dentro de la información disponible hasta llegar a la estructura de la clase *Linea* y ejecuta todas las animaciones disponibles.
En términos generales, describe brevemente cuales son las características generales de las animaciones.

Navega en los subtemas contenidos en el tema de *Herencia*

Describe en términos generales las pantallas que se muestran

Carga el entorno de programación

Describe en términos generales la pantalla que se muestra

Cierra la ventana

Cierra la ventana del explorador

Gracias

CUESTIONARIO DE SALIDA

Sistema de software para la utilización de pizarrones electrónicos para apoyar la enseñanza de la programación orientada a objetos con Java.

Nombre: _____

Por favor responde las preguntas siguientes basándote en tu experiencia usando el sistema de software para apoyar la enseñanza de la POO. Es importante que llenes las líneas de comentarios.

1. **En general, me resulto sencillo usar el sistema de software.**

___ Completamente de acuerdo
___ De acuerdo
___ Ni de acuerdo ni en desacuerdo
___ En desacuerdo
___ Completamente en desacuerdo

2. **El sistema de software me pareció un método efectivo para apoyar la enseñanza de la programación orientada a objetos.**

___ Completamente de acuerdo
___ De acuerdo
___ Ni de acuerdo ni en desacuerdo
___ En desacuerdo
___ Completamente en desacuerdo

3. **Los siguientes aspectos del sistema de software para apoyar la enseñanza de la POO, me parecieron fáciles de usar (Por favor lista de 1 - 3 aspectos).**

a. _____
b. _____
c. _____

4. **Los siguientes aspectos del sistema de software para apoyar la enseñanza de la POO, me parecieron difíciles de usar (Por favor lista de 1 - 3 aspectos).**

a. _____
b. _____
c. _____

5. **La interfaz del sistema de software me pareció un método sencillo para desplegar información relevante sobre algún concepto particular.**

___ Completamente de acuerdo
___ De acuerdo
___ Ni de acuerdo ni en desacuerdo
___ En desacuerdo
___ Completamente en desacuerdo

6. Usando la siguiente columna de intervalos, encierra el número más cercano a lo que sientas respecto a la característica del sistema de software.

Simple	3	2	1	0	1	2	3	Complejo
Tecnología alta	3	2	1	0	1	2	3	Tecnología baja
Confiable	3	2	1	0	1	2	3	No confiable
Fácil de usar	3	2	1	0	1	2	3	Difícil de usar
Amigable	3	2	1	0	1	2	3	No amigable
Profesional	3	2	1	0	1	2	3	No profesional
Seguro	3	2	1	0	1	2	3	No seguro
Durable	3	2	1	0	1	2	3	Frágil
Atractivo	3	2	1	0	1	2	3	No atractivo
Alta calidad	3	2	1	0	1	2	3	Baja calidad
Me gusta	3	2	1	0	1	2	3	No me gusta

7. Por favor selecciona la opción que más refleje tu opinión sobre el sistema.

   a. *Las tareas fueron fáciles de realizar.*
   ___ Completamente de acuerdo
   ___ De acuerdo
   ___ Ni de acuerdo ni en desacuerdo
   ___ En desacuerdo
   ___ Completamente en desacuerdo

   b. *La información en las pantallas está organizada de manera óptima*
   ___ Completamente de acuerdo
   ___ De acuerdo
   ___ Ni de acuerdo ni en desacuerdo
   ___ En desacuerdo
   ___ Completamente en desacuerdo

   c. *La cantidad de pasos para realizar una tarea es la correcta.*
   ___ Completamente de acuerdo
   ___ De acuerdo
   ___ Ni de acuerdo ni en desacuerdo
   ___ En desacuerdo
   ___ Completamente en desacuerdo

   d. *La terminología fue clara y precisa.*
   ___ Completamente de acuerdo
   ___ De acuerdo
   ___ Ni de acuerdo ni en desacuerdo
   ___ En desacuerdo
   ___ Completamente en desacuerdo

e. *Siempre pude realizar la operación que necesité.*

___ Completamente de acuerdo
___ De acuerdo
___ Ni de acuerdo ni en desacuerdo
___ En desacuerdo
___ Completamente en desacuerdo

f. *La cantidad de información en la pantalla fue adecuada para desarrollar la tarea.*

___ Completamente de acuerdo
___ De acuerdo
___ Ni de acuerdo ni en desacuerdo
___ En desacuerdo
___ Completamente en desacuerdo

8. **Por favor agrega cualquier comentario que creas que puede ayudarnos a mejorar el sistema de software. Agradeceremos tus opiniones específicamente en cuanto a:**
   - ✓ *Funciones que son esenciales o superfluas para la tarea.*
   - ✓ *Características que te gustaría ver en productos futuros.*

Gracias