UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Algunos elementos de la didáctica de la Programación y el pensamiento Computacional

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

CINTHIA RODRÍGUEZ MAYA

DIRECTOR DE TESIS:
DR. JOSÉ DE JESÚS GALAVIZ CASAS
2014

MÉXICO., D.F.

Hoja de datos del jurado

1. Datos del alumno
Rodríguez
Maya
Cinthia
57 55 57 28
Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
304036954

2. Datos del tutor
Dr
José de Jesús
Galaviz
Casas

3. Datos del sinodal 1
Dra
Amparo
López
Gaona

4. Datos del sinodal 2
Mat
Salvador
López
Mendoza

5. Datos del sinodal 3
M en I
Gerardo
Avilés
Rosas

6. Datos del sinodal 4
Dr
Canek
Peláez
Valdés

7. Datos del trabajo escrito
Algunos elementos de la didáctica de la Programación y el pensamiento Computacional
137p
2014

Agradecimientos

A Dios por haberme permitido llegar hasta este punto y haberme dado salud y uso de la razón para lograr mis objetivos, además de su infinita bondad y amor.

A mi padre Juan Rodríguez y a la memoria de mi madre Juana Maya, gracias por su apoyo, consejos, comprensión, amor y por brindarme todo lo que pudieron para que tuviera una carrera. Me han dado todo lo que soy como persona, mis valores, principios y empeño para lograr mis objetivos.

A mis hermanos por su apoyo incondicional y cariño.

A María del Socorro Díaz por todo el cariño, amistad y apoyo que me ha brindado desde que nací.

A René Adrián Dávila por su apoyo en momentos difíciles tanto académicos como personales.

A todos los profesores que he tenido a lo largo de mi vida como estudiante, en particular, agradezco a mi tutor el Dr. José Galaviz por su paciencia y apoyo en la realización de este trabajo.
Al Mat. Salvador López Mendoza por los valiosos conocimientos que adquirí en sus excelentes cursos de Sistemas Operativos y Redes de Computadoras, así como por darme la oportunidad de ser su ayudante y desempeñar una de mis labores favoritas que es dar clase.
Al M. en C. Agustín Ontiveros Pineda quien me asesoró y orientó incondicionalmente en la parte matemática de la carrera.

A mis sinodales por leer este trabajo y sus valiosas correcciones.

A la Universidad Nacional Autónoma de México, en especial a la Facultad de Ciencias por haberme dado acceso a tan excelentes estudios.

Índice general

1. Enseñanza y aprendizaje **7**
- 1.1. Concepto de aprendizaje humano . . . 7
- 1.2. Tipos de aprendizaje . . . 8
  - 1.2.1. Aprendizaje memorístico o repetitivo . . . 9
  - 1.2.2. Aprendizaje receptivo . . . 9
  - 1.2.3. Aprendizaje visual . . . 9
  - 1.2.4. Aprendizaje auditivo . . . 9
  - 1.2.5. Aprendizaje significativo . . . 9
  - 1.2.6. Aprendizaje por descubrimiento . . . 9
- 1.3. Mecanismos de enseñanza . . . 10
- 1.4. Mecanismos de enseñanza en Ciencias Exactas . . . 11
  - 1.4.1. Aprendizaje Basado en Problemas . . . 12

2. Enseñanza y aprendizaje de la programación **15**
- 2.1. ¿Qué es la programación? . . . 15
- 2.2. Paradigmas de los lenguajes de programación . . . 15
  - 2.2.1. Paradigma imperativo . . . 16
  - 2.2.2. Paradigma orientado a objetos . . . 16
  - 2.2.3. Paradigma lógico . . . 16
  - 2.2.4. Paradigma funcional . . . 17
- 2.3. Definición de problema . . . 17
  - 2.3.1. Cómo resolver problemas . . . 17
- 2.4. Pensamiento computacional . . . 18
- 2.5. Técnicas de enseñanza de la programación . . . 19
  - 2.5.1. Alice . . . 19
  - 2.5.2. Greenfoot . . . 20
  - 2.5.3. BlueJ . . . 21
  - 2.5.4. Sugerencias Didácticas . . . 21
  - 2.5.5. Buenas prácticas de programación . . . 22
- 2.6. Dificultades en la enseñanza de la programación . . . 23

3. Diseño de Proyectos de Programación **25**
- 3.1. Proyecto 1 . . . 26
- 3.2. Proyecto 2 . . . 29
- 3.3. Proyecto 3 . . . 31
- 3.4. Proyecto 4 . . . 34
- 3.5. Proyecto 5 . . . 36
- 3.6. Proyecto 6 . . . 41
- 3.7. Proyecto 7 . . . 44

3.8. Proyecto 8 48
3.9. Proyecto 9 50
3.10. Proyecto 10 55

A. Implementación del proyecto 1 **63**

B. Implementación del proyecto 2 **65**

C. Implementación del proyecto 3 **67**

D. Implementación del proyecto 4 **73**

E. Implementación del proyecto 5 **83**

F. Implementación del proyecto 6 **97**

G. Implementación del proyecto 7 **105**

H. Implementación del proyecto 8 **109**

I. Implementación del proyecto 9 **113**

J. Implementación del proyecto 10 **123**

Introducción

En el presente trabajo se mencionan algunos de los elementos didácticos a tomar en cuenta antes de dar un primer curso de Programación, por primer curso debemos entender que se habla de materias de Programación introductorias en la carrera de Ciencias de la Computación.

En el capítulo 1 se define lo que es el aprendizaje así como los tipos de aprendizaje más usuales en el ser humano, se hace un análisis sobre cómo un computólogo resuelve un problema tomando como referencia el método para resolver problemas propuesto por Pólya[1] en su libro *How to Solve it*. Se analiza el Aprendizaje Basado en Problemas como un mecanismo de enseñanza en las Ciencias exactas.

En el capítulo 2 se hace un breve resumen de los 4 principales paradigmas en los que se clasifican los lenguajes de programación, se da una definición de lo que es el pensamiento computacional y se brindan técnicas que pueden resultar de utilidad en la enseñanza de la programación.

El capítulo 3 es una propuesta de proyectos de programación que van desde aritmética hasta persistencia; cada uno de estos proyectos incluye código fuente disponible en los anexos de esta tesis.

[1] George Polya (1887-1985), matemático húngaro autor de libros sobre cómo debería enseñarse y aprender la manera de resolver problemas.

Capítulo 1

Enseñanza y aprendizaje

1.1. Concepto de aprendizaje humano

Toda la vida estamos en constante aprendizaje, es una de las facultades inherentes del ser humano. El aprendizaje se da en formas diversas, cualquiera de esas formas tiene el mismo fin, mejorar la calidad de vida del ser humano.

El aprendizaje es el proceso a través del cual se adquieren o modifican habilidades, destrezas, conocimientos o conductas como resultado del estudio, la experiencia, la instrucción, el razonamiento y la observación.[Rojas2001]

> "Podemos definir el aprendizaje como un proceso de cambio relativamente permanente en el comportamiento de una persona generado por la experiencia." Feldman, R.S. (2005) "Psicología: con aplicaciones en países de habla hispana". (Sexta Edición) México, McGrawHill.

En primer lugar, aprendizaje supone un cambio conductual o un cambio en la capacidad conductual. En segundo lugar, dicho cambio debe ser perdurable en el tiempo. En tercer lugar, otro criterio fundamental es que el aprendizaje ocurre a través de la práctica o de otras formas de experiencia.[Rojas2001]

Para Vygotsky(1896-1934), el contexto social influye en el aprendizaje más que las actitudes y las creencias; tiene una profunda influencia en cómo se piensa y en lo qué se piensa. El contexto forma parte del proceso de desarrollo y, como tal, moldea los procesos cognitivos. El contexto social debe ser considerado en diversos niveles:

1. El nivel interactivo inmediato
   Constituido por las personas con quienes el individuo interactúa en esos momentos.

2. El nivel estructural
   Constituido por las estructuras sociales que influyen en el individuo, tales como la familia y la escuela.

3. El nivel cultural o social general
   Constituido por la sociedad en general, como el lenguaje, el sistema numérico y la tecnología.[1]

[1]Bodrova Elena y Debora J. Leong. "La teoría de Vygotsky: principios de la psicología y la educación". En: Curso de Formación y Actualización Profesional para el Personal Docente de Educación Preescolar. Vol. I. SEP. México 2005, pag. 48.

La mente humana, de acuerdo con Piaget(1896-1980), opera en términos de dos funciones no cambiantes: asimilación y acomodación. La asimilación es establecimiento de relaciones entre los conocimientos previos y los nuevos, la acomodación es la reestructuración del propio conocimiento. Mediante la asimilación y la acomodación vamos reestructurando cognitivamente nuestro aprendizaje a lo largo de nuestro desarrollo.

Para Piaget la educación tiene como finalidad favorecer el crecimiento intelectual, afectivo y social del individuo, pero teniendo en cuenta que ese crecimiento es el resultado de procesos evolutivos naturales. La acción educativa, por tanto, ha de estructurarse de manera que favorezcan los procesos constructivos personales, mediante los cuales opera el crecimiento. Las actividades de descubrimiento deben ser entonces, prioritarias. Esto no implica que el individuo tenga que aprender en solitario. Al contrario, una de las características básicas del modelo pedagógico piagetiano es, justamente, el modo en que resaltan las interacciones sociales.

Ausubel(1918-2008) fue influenciado por los aspectos cognitivos de la teoría de Piaget, y planteó su Teoría del Aprendizaje Significativo por Recepción, este tipo de aprendizaje tiene las siguientes características:

- Presentar la información al alumno en su forma final; es decir, no requiere más investigación por parte del alumno para simplificar algún resultado o llegar a una conclusión.
- Presentar temas usando y aprovechando los esquemas previos del estudiante.
- Dar cierta información al estudiante permitiendo que éste por sí mismo descubra un conocimiento nuevo.
- Mostrar materiales pedagógicos de forma coloquial y organizada que no distraigan la concentración del estudiante.
- Hacer que haya una participación activa por parte del alumno.

Para lograr el aprendizaje de un nuevo concepto, según Ausubel, es necesario tender un puente cognitivo entre conocimientos adquiridos por el alumno, siendo éstos de carácter general y un nuevo concepto. De esta forma, la nueva información al ser relacionada con la anterior, es guardada en la memoria de largo plazo.[Pozo1987]

1.2. Tipos de aprendizaje

Cada persona tiene una forma o estilo particular de establecer relación con el mundo y por lo tanto para aprender, es por ello que se han desarrollado distintos modelos que aproximan una clasificación de estas distintas formas de aprendizaje.[Alonso1994]

Para la pedagogía, los tipos de aprendizaje más comunes son:[Rojas2001]

- Aprendizaje memorístico o repetitivo
- Aprendizaje receptivo
- Aprendizaje visual
- Aprendizaje auditivo
- Aprendizaje significativo
- Aprendizaje por descubrimiento

1.2.1. Aprendizaje memorístico o repetitivo

Generalmente, al referirnos al aprendizaje memorístico o repetitivo, nos referimos a un aprendizaje mecánico, en el que se obliga al individuo a recordar información a través de un proceso de repetición oral o escrita.

Desde la psicología cognitiva se considera que si el aprendizaje se logra sólo mediante la repetición al poco tiempo se olvidará, ya que los nuevos conocimientos se incorporan de forma arbitraria en la estructura cognitiva del alumno y no quedan almacenados en la memoria de largo plazo.[Gimeno1992]

1.2.2. Aprendizaje receptivo

En este tipo de aprendizaje el individuo sólo necesita comprender el contenido para poder reproducirlo de manera posterior. El alumno es un receptor pasivo de los conocimientos transmitidos por el profesor, y fiel reproductor de los mismos. El material de conocimiento se basa generalmente en un libro de texto. [Pozo1996]

1.2.3. Aprendizaje visual

El Aprendizaje Visual se define como un método de enseñanza/aprendizaje que utiliza un conjunto de organizadores gráficos (métodos visuales para ordenar información), con el objeto de ayudar a los estudiantes, mediante el trabajo con ideas y conceptos, a pensar y a aprender más efectivamente. Ejemplos de estos organizadores son: mapas conceptuales, diagramas causa-efecto y líneas de tiempo, entre otros.

Los individuos visuales poseen una conducta organizada, observadora y tranquila, su aprendizaje se basa en lo que ven y piensan mediante imágenes.[Pozo1996]

1.2.4. Aprendizaje auditivo

Un individuo cuyo aprendizaje es auditivo, aprende mejor escuchando los conceptos transmitidos. Este tipo de persona tiende a entender mejor las explicaciones orales y puede recordar y comprender mejor la información si lee en voz alta. [Pozo1996]

1.2.5. Aprendizaje significativo

El aprendizaje significativo se alcanza cuando los nuevos conocimientos se incorporan en forma sustantiva en la estructura cognitiva del individuo, ya que éste considera valiosa la información presentada y logra relacionar los conocimientos nuevos con los anteriores. [Valle1993]

Este tipo de aprendizaje, a diferencia del aprendizaje memorístico, produce una retención más duradera de la información. La nueva información al relacionarse con la anterior, es depositada en la llamada memoria de largo plazo, en la que se conservan desde elementos importantes hasta detalles secundarios concretos. Es activo, pues depende de la asimilación deliberada de las actividades de aprendizaje por parte del individuo.[Dávila2000]

1.2.6. Aprendizaje por descubrimiento

El aprendizaje por descubrimiento es el aprendizaje en el que los estudiantes construyen por sí mismos sus propios conocimientos, en contraste con la enseñanza tradicional o transmisora del

conocimiento, donde el docente pretende que la información sea simplemente recibida por los estudiantes.[Sprinthall1996]

Pozo y Gómez[2] mencionan que el aprendizaje por descubrimiento es especialmente efectivo en la enseñanza de las ciencias. Según resultados reportados en diversos estudios, los estudiantes que emplean estrategias que favorecen el aprendizaje por descubrimiento, obtienen mejores resultados que aquellos donde la enseñanza se basa en la transmisión de información.

1.3. Mecanismos de enseñanza

Previo a abordar los diversos mecanismos de enseñanza, se considera pertinente aclarar que tanto enseñanza como aprendizaje son conceptos diferentes. Para esto, [Fenstermacher2006] establece que la enseñanza es una actividad en la que debe haber al menos dos personas, una de las cuales posee un conocimiento o habilidad que la otra no posee, la primera intenta transmitir esos conocimientos o habilidades a la segunda, estableciéndose entre ambas una cierta relación a fin de que la segunda los adquiera. ¿En que difieren el aprendizaje y la enseñanza? El aprendizaje puede realizarlo uno mismo, la enseñanza por el contrario se produce por lo general estando presente una persona más. El aprendizaje implica la adquisición de algo, la enseñanza implica dar algo.

Un modelo de enseñanza es un plan estructurado que puede usarse para orientar la enseñanza en las aulas, así mismo, es de utilidad en la elaboración de materiales de apoyo que facilitan el proceso de aprendizaje. Hemos de considerar que no existe un único camino para el éxito pedagógico, ni la solución sin esfuerzo de los complejos problemas docentes, ni la descripción del *mejor método de enseñanza*. [Martínez2004]

A continuación se mencionan algunas de las clasificaciones más comunes sobre métodos de enseñanza según Renzo Titone[3] e Imideo Nérici.[4]

- Métodos en cuanto a la forma de razonamiento

  - Método deductivo

    Cuando el asunto estudiado procede de lo general a lo particular. El profesor presenta conceptos, principios, definiciones o afirmaciones de las que se van extrayendo conclusiones y consecuencias.

    El método deductivo es muy válido cuando los conceptos, definiciones, fórmulas, leyes y principios ya están muy asimilados por el alumno, pues a partir de ellos se generan las deducciones. Evita trabajo y ahorra tiempo.

  - Método inductivo

    Cuando el asunto estudiado se presenta por medio de casos particulares sugiriéndose que se descubra el principio general que los rige. Es el método activo por excelencia, que ha dado lugar a la mayoría de descubrimientos científicos. Se basa en la experiencia, la participación y los hechos; posibilita en gran medida la generalización y el razonamiento globalizado.

  - Método analógico o comparativo

    El pensamiento va de lo particular a lo particular. Es fundamentalmente la forma de razonar de los más pequeños, sin olvidar su importancia en todas las edades.

[2] Las ideas de los alumnos sobre la ciencia: Una interpretación desde la psicología cognitiva. *Enseñanza de las ciencias*,9, pp.35-52, 1991

[3] Psicodidáctica. Madrid, Narcea; 1986

[4] metodología de la enseñanza. México Kapelusz Mexicana; 1985

- Métodos en cuanto a la organización de la materia
  - Método basado en la lógica de la tradición o de la disciplina científica
    Cuando los datos o los hechos se presentan en orden de antecedente y consecuente, obedeciendo a una estructuración de hechos que va desde lo menos a lo más complejo o desde el origen hasta la actualidad.
  - Método basado en la psicología del alumno
    Cuando el orden seguido responde más bien a los intereses y experiencias del alumno. Se ciñe a la motivación del momento y va de lo conocido por el alumno a lo desconocido por él.
- Métodos en cuanto a las actividades externas del alumno
  - Método pasivo
    El alumno es el receptor de conocimiento del profesor, quien expone o realiza dictados. Es aprender antes de comprender. El profesor impone como verdad lo que enseña a sus alumnos.
  - Método activo
    El alumno participa en clase, realiza prácticas de campo o actividades de investigación que profundizan y refuerzan su aprendizaje.

1.4. Mecanismos de enseñanza en Ciencias Exactas

Los descubrimientos e inventos científicos y tecnológicos han tenido repercusiones considerables en la historia de la humanidad durante los últimos siglos. Por paradójico que parezca, la enseñanza científica y tecnológica no se ha desarrollado al mismo ritmo que los avances de la ciencia y la tecnología. En la mayoría de los países del mundo la enseñanza de la ciencia y la tecnología no figura entre los temas prioritarios de los programas de educación. Además, las políticas, planes de estudios, métodos y materiales pedagógicos relativos a las disciplinas científicas, así como la formación de los docentes especializados en las mismas, suelen ser obsoletos y poco interesantes. Por eso, no es sorprendente que en la enseñanza de materias de índole científico los docentes carezcan a menudo de motivación, al no haber sido suficientemente formados y no disponer del material necesario; y que los alumnos –en particular los pertenecientes a la educación básica– al no tener una motivación similar, suelan etiquetar a las disciplinas de índole científico como una elección poco acertada. [UNESCO2005]

La UNESCO [5] tiene plena conciencia de sus responsabilidades y del importante papel que debe desempeñar para aportar una respuesta a este problema. La Organización ha estimulado la realización de actividades en ámbitos como: [UNESCO2005]

- Fortalecimiento de capacidades
  Suministro de material didáctico y acceso a información científica que nutran el conocimiento tanto de docentes, como de los encargados de formar planes de estudio educativos.
- Mejora en calidad y pertinencia
  Actualizar planes de estudio conforme avancen los adelantos científicos y promover actividades que estimulen el interés de los estudiantes desde la educación básica.
- Elaboración de un marco de acción Internacional
  Organizar seminarios regionales y nacionales así como redes científicas.

[5] Organización de las Naciones Unidas para la Educación la Ciencia y la Cultura

Según la UNESCO, gracias a las actividades mencionadas así como a La Declaración de Budapest sobre la ciencia, se ha tenido un éxito considerable año con año en la difusión de la ciencia y se espera que más organizaciones y países se sumen al esfuerzo por resolver este problema de rezago educativo. [UNESCO2005]

Uno de los principales obstáculos en la enseñanza de las Ciencias es que los alumnos tienen la idea de que el conocimiento científico se presenta a través de ecuaciones y definiciones que sólo pueden comprender si logran memorizar. Esto da pauta a que el conocimiento científico no llegue a ser significativo para esos alumnos, cuyo procentaje puede llegar a ser alto, sobre todo en lo que abarca desde educación básica hasta nivel medio superior. [Linder1993]

Se ha observado que las técnicas y razonamientos empleados por la mayoría de los alumnos, al abordar problemas de índole científico, son incorrectos, superficiales o de dudosa utilidad[Carrascosa1985]. Un problema es que los alumnos no son capaces de comprender de manera exacta un problema, es decir, no saben que no saben. [Otero1990]

1.4.1. Aprendizaje Basado en Problemas

El *Aprendizaje Basado en Problemas* está centrado en el estudiante, pero promueve el desarrollo de una cultura de trabajo colectivo involucrando a todos los miembros del grupo en el proceso de aprendizaje cuando enfrentan un problema. Se busca que el profesor fomente en el grupo de estudiantes la libertad de que cada uno de ellos exprese sus diferentes puntos de vista que sirvan de guía en la solución de problemas apegados a la realidad. [Morales2004]

La ruta que siguen los estudiantes durante el desarrollo de este tipo de aprendizaje se puede sintetizar en los siguientes pasos: [Morales2004]

Paso 1 *Leer y analizar el escenario del problema*
Se busca que el alumno verifique su comprensión del escenario mediante la discusión del mismo dentro de su equipo de trabajo.

Paso 2 *Realizar una lluvia de ideas*
Los alumnos usualmente tienen teorías o hipótesis sobre las causas del problema así como ideas de cómo resolverlo. Estas deben de enlistarse y serán aceptadas o rechazadas según se avance en la investigación.

Paso 3 *Hacer una lista de aquello que se conoce*
Se debe hacer una lista de todo aquello que el equipo conoce acerca del problema o situación.

Paso 4 *Hacer una lista de aquello que se desconoce*
Se debe hacer una lista con todo aquello que el equipo cree se debe de saber para resolver el problema. Existen muy diversos tipos de preguntas que pueden ser adecuadas; algunas pueden relacionarse con conceptos o principios que deben estudiarse para resolver la situación.

Paso 5 *Hacer una lista de aquello que necesita hacerse para resolver el problema*
Planear las estrategias de investigación. Es aconsejable que en grupo los alumnos elaboren una lista de las acciones que deben realizarse.

Paso 6 *Definir el problema*
La definición del problema consiste en un par de declaraciones que expliquen claramente lo que

el equipo desea resolver, producir, responder, probar o demostrar.

Paso 7 *Obtener información*
El equipo localizará, acopiará, organizará, analizará e interpretará la información de diversas fuentes.

Paso 8 *Presentar resultados*
El equipo presentará un reporte o hará una presentación en la cual se muestren las recomendaciones, predicciones, inferencias o aquello que sea conveniente en relación a la solución del problema.

Se considera que la principal ventaja de este tipo de aprendizaje es que es más adecuado que los métodos tradicionales por transmisión(presentación del contenido del tema por parte del profesor al estudiante), ya que entre las situaciones más frecuentes que se deben afrontar en las ciencias experimentales se encuentra la búsqueda de soluciones a situaciones problemáticas.

> "El aprendizaje a partir de problemas es el mejor medio disponible para desarrollar las potencialidades generales de los alumnos." Birch, W. (1986). Towards a model for problem-based learning. *Studies in Higher Education*, 11, pp.73-82.

Entonces, podemos darnos cuenta que el *Aprendizaje Basado en Problemas*, a diferencia del aprendizaje tradicional, donde el profesor expone los conocimientos, brinda la posibilidad de desarrollar en el alumno la habilidad de preguntarse a sí mismo qué necesita saber y fomentar en él la disciplina de buscar medios que le brinden la información que haya considerado le será de utilidad y no limitarse únicamente a lo que el profesor le enseñe.

Este es el tipo de aprendizaje que considero adecuado fomentar en los alumnos de la Facultad de Ciencias, particularmente a quienes ingresan a la carrera Ciencias de la Computación, sobre todo durante los primeros semestres en las asignaturas que involucran la enseñanza de la programación, debido a que programar es una habilidad que se desarrolla através de la práctica constante y resolución de problemas.

> "Es del enfrentamiento con la dificultad, con la incertidumbre, con el problema..., de donde emerge el proceso reflexivo que obliga a extender, diferenciar, reformular, las teorías previas, para configurar otras nuevas". Dewey, J. (1933) How We Think. A restatement of the relation of reflective thinking to the educative process, Boston: D. C. Heath.

Capítulo 2

Enseñanza y aprendizaje de la programación

Programar es una de las habilidades que todo computólogo debe adquirir, debido a que muchos problemas en Ciencias de la Computación requieren el manejo de la programación para la automatización de diversas tareas. Considero que la paciencia y la creatividad son algunas de las harramientas fundamentales en el aprendizaje de esta disciplina.

2.1. ¿Qué es la programación?

Las instrucciones que entiende y con las que opera la computadora no es el lenguaje natural al que estamos acostumbrados los seres humanos; para que la computadora pueda entender una instrucción dada por nosotros, debemos utilizar un medio llamado lenguaje de programación.

Podemos ver a un lenguaje de programación como un idioma que entiende la computadora, y de la misma manera que las personas tenemos la capacidad de entender varios idiomas, la computadora es capaz de responder a diversos lenguajes de programación. En realidad nuestras computadoras son capaces de ejecutar las instrucciones dadas en un único lenguaje nativo llamado *lenguaje de máquina*, que depende del diseño particular de cada computadora. [Peter1997]

¿Qué es entonces la programación de computadoras? Podemos definirla a grandes rasgos como el proceso de diseñar algoritmos y escribirlos en algún lenguaje de programación. Un algoritmo es una secuencia ordenada y finita de pasos bien definidos para llevar a cabo una determinada labor. La programación de computadoras es una actividad imprescindible en muchos ámbitos de nuestra era moderna, ya que ésta se encuentra presente en aparatos cotidianos como cajeros automáticos, teléfonos celulares, los dispositivos que controlan nuestros medios de transporte o de telecomunicaciones.

El objetivo de la carrera de Ciencias de la Computación no es formar programadores, sin embargo, es imprescindible que cada computólogo domine la programación contando con esta herramienta en su formación.

2.2. Paradigmas de los lenguajes de programación

Un paradigma de programación es una colección de modelos conceptuales que en conjunto modelan el proceso de diseño y determinan la estructura de un programa. Esa estructura conceptual controla el modo en que el desarrollador de software piensa y formula soluciones que luego son

implementadas en un lenguaje de programación. [Loïc2005]

Un paradigma está constituido por los supuestos teóricos generales, las leyes y las técnicas para su aplicación que adoptan los miembros de una determinada comunidad científica.

Podemos decir que los paradigmas son marcos de referencia que imponen reglas sobre cómo se deben plantear las cosas, indican qué es válido dentro del paradigma y qué está fuera de sus límites. Un paradigma distinto implica nuevas reglas, elementos, límites y maneras de pensar.

Los paradigmas pueden ser considerados como patrones de pensamiento para la resolución de problemas. Representan un enfoque particular o filosofía para la construcción del software. No es mejor uno que otro sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

Aunque diversas fuentes de información nos mencionan la existencia de diversos paradigmas de lenguajes de programación, a continuación se mencionarán los más comunes.

2.2.1. Paradigma imperativo

Los lenguajes de programación que pertenecen a este paradigma se caracterizan por el uso de instrucciones ordenadas que paso a paso cambian el estado del programa, los programas imperativos suelen ser parecidos a una receta de cocina donde el seguimiento de cada uno de los pasos de manera ordenada es esencial. En este paradigma las instrucciones son justamente imperativas; órdenes que le damos a la computadora. [Loïc2005]

La implementación de hardware de la mayoría de las computadoras es imperativa; prácticamente todo el hardware está diseñado para ejecutar código de máquina, que es nativo a la computadora. Algunos lenguajes de programación que corresponden a este paradigma son: C, Pascal, Fortran y Algol.

2.2.2. Paradigma orientado a objetos

Graddy Booch[1] define a la programación orientada a objetos como *un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representan una instancia de alguna clase y cuyas clases son todos miembros de una jerarquía de clases unidos mediante relaciones de herencia.*

Este paradigma es muy parecido a la interacción humana con el mundo real y se caracteriza por el uso de: encapsulación, abstracción, herencia y polimorfismo; propiedades que no encontramos en los demás paradigmas. Algunos lenguajes de programación que corresponden a este paradigma son: Java, SIMULA, Ada, C++, entre otros.

2.2.3. Paradigma lógico

El paradigma lógico encaja muy bien cuando se aplica en dominios de problemas que tienen que ver con la extracción de conocimiento a partir de datos básicos y relaciones. [Loïc2005]

A diferencia de los demás paradigmas, trabajar en este significa especificar qué hacer y no cómo hacerlo. El proceso general de la programación lógica es que a partir de un conjunto de reglas (axio-

[1]Booch Grady, Análisis y diseño orientado a objetos con aplicaciones, segunda edición, Adisson-Wesley, California 2007

mas) e inferencias podamos comprobar nuevas proposiciones que nos sean relevantes. Este proceso está basado en reglas de lógica de primer orden. El lenguaje lógico más famoso es Prolog.

2.2.4. Paradigma funcional

La programación funcional se origina a partir de la teoría de funciones, todos los cálculos se realizan a través de llamadas a ellas. Estas funciones son valores de primera clase, que permiten que otras funciones reciban, almacenen y regresen funciones; lo que no poseen otros lenguajes de programación. [Loïc2005]

Los principales lenguajes de programación de este paradigma son: Haskell, LISP, Scheme y Miranda.

La programación funcional tiene sus raíces en el cálculo lambda, un sistema formal desarrollado en la década de 1930 por Alonzo Church[2] y Stephen Kleen[3] para investigar la definición de función, la aplicación de las funciones y la recursión.

2.3. Definición de problema

¿Qué es un problema?

> "Un problema es una situación, cuantitativa o no, que pide una solución para la cual los individuos implicados no conocen medios o caminos evidentes para obtenerla." Krulik, S. y Rudnick, K. Problem solving in school mathematics. National council of teachers of mathematics. Year Book. Virginia: Reston. (1980)

El concepto de problema no tiene un significado único y depende en particular del medio o contexto en que nos situemos. En esta tesis nos enfocaremos en problemas de carácter computacional.

¿Qué es un problema en Computación?

Un problema en computación es una relación entre un conjunto de instancias (los elementos que intervienen en el problema) y un conjunto de soluciones. Un problema computacional permite establecer formalmente la relación deseada entre la entrada de un algoritmo y su salida. Una solución algorítmica a un problema computacional consiste de un algoritmo que por cada instancia del problema calcula al menos una solución correspondiente (en caso de haberla) o demuestra que no existe solución alguna. [Cormen2009]

2.3.1. Cómo resolver problemas

Polya (1887-1985) menciona en su libro *How To Solve It*, los siguientes pasos a tomar en cuenta en la solución de problemas:

1. Se debe entender cuál es el problema. Es decir, debemos ser capaces de responder a la pregunta ¿Cuál es la incógnita o incógnitas?, así como determinar cuál es la condición que debe satisfacerse.

2. Encontrar una conexión entre los datos que se tengan y la incógnita, ya sea por medio de problemas similares vistos anteriormente o auxiliados de teoremas, fórmulas, etc.

3. Resolver el problema paso a paso, cuidando que cada uno de ellos sea claro y correcto.

[2](1903-1995) matemático y lógico norteamericano responsable por crear la base de la computación teórica.
[3](1909-1994) lógico y matemático estadounidense considerado uno de los fundadores de la lógica matemática.

Tomando como base a Polya, podemos decir que un computólogo puede resolver un problema mediante el uso de los siguientes pasos:

1. **Entender bien el problema**
   Hay que tener una idea clara de qué hay que hacer. Para ello debe haber una lectura o escucha atenta del planteamiento del problema, es de gran utilidad responder a las preguntas ¿Cuáles son los datos de entrada? y ¿Cuáles son los resultados de salida esperados?. Encontrar de ser posible una conexión entre el problema actual con problemas similares vistos anteriormente.

2. **Diseñar el algoritmo que resuelva el problema**
   Desarrollar la serie de pasos que resuelvan nuestro programa analizando cuál es el proceso por el que pasan los datos de entrada para llegar a la solución. Realizar pseudocódigo o diagramas de flujo en caso de ser necesario para plasmar la idea de nuestra solución.

3. **Implementar de ser posible, el algoritmo de solución**
   Codificar nuestro algoritmo a un lenguaje de programación obedeciendo las reglas de sintaxis y haciendo uso de las bibliotecas del mismo con el fin de ahorrar tiempo o reutilizar software. Una herramienta de gran utilidad antes de escribir código fuente es el pseudocódigo, el cual es un lenguaje informal que ayuda a los programadores a desarrollar algoritmos sin tener que preocuparse por los estrictos detalles de las síntaxis del lenguaje de programación empleado. El pseudocódigo es similar al lenguaje cotidiano.

4. **Realizar pruebas al código implementado**
   Poner a prueba el programa realizado con una buena y amplia selección de valores de entrada para asegurar el buen funcionamiento del programa.

   La mayoría de los estudiantes que comienzan a programar encuentran esta actividad difícil debido a que se concentran más en la sintaxis del lenguaje de programación que en entender bien el problema y en diseñar el algoritmo de solución. Esto trae como consecuencia que sus programas no funcionen como deben, o bien, terminen funcionando "a martillazos", lo cual da como resultado código fuente difícil de entender y que quizá no sea del todo correcto; esto ocasiona que muchos estudiantes pierdan motivación o incluso lleguen a odiar programar. [Cabral2012]

2.4. Pensamiento computacional

La Sociedad Internacional para la Tecnología en la Educación(ISTE) y la Asociación de Profesores de Ciencias de la Computación(CSTA) han desarrollado una definición de lo que es el pensamiento computacional, el cual se describe por los siguientes puntos: [ACM2011]

- Formulación de problemas de manera que la solución de estos se logre utilizando una computadora.
- Organización y análisis lógico de datos.
- Representación de los datos a través de abstracciones tales como modelos.
- Automatización de soluciones a través del pensamiento algorítmico.
- Identificar, analizar y aplicar las posibles soluciones con el objetivo de lograr la combinación entre los pasos y recursos que sea más eficaz.

- Generalizar la solución de un problema particular a una extensa variedad de problemas similares.

El pensamiento computacional se refuerza con las siguientes actitudes:

- Confianza con el trato de la complejidad.
- Persistencia en el trabajo con los problemas difíciles.
- Tolerancia a la ambigüedad.
- La capacidad de comunicarse y trabajar con otros para lograr un objetivo común o solución.

La ACM (Association for Computing Machinery) define las ciencias de la computación como: [Barr2011]

> "El estudio de las computadoras y sus procesos algorítmicos, incluyendo su diseño de software y hardware, así como sus aplicaciones, y su impacto en la sociedad."

Las Ciencias de la Computación incluyen: programación, diseño de hardware, redes, gráficación, bases de datos, seguridad, diseño de software, lenguajes de programación y paradigmas, lógica, niveles de abstracción, inteligencia artificial así como aplicaciones de las Ciencias de la Computación en diferentes campos.

2.5. Técnicas de enseñanza de la programación

Aprender a programar es una tarea difícil debido, entre otras cosas, a que no existe un procedimiento para ello. Para aprender a programar es necesario escribir programas, no basta con leer o entender programas ya escritos, es necesario enfrentar el reto de programar, tener tropiezos en el camino y aprender de ellos. Para lograrlo se debe aprender a analizar un problema, descomponerlo en sus partes y esbozar una solución. Una vez que se tiene el esbozo de solución se puede proceder a escribir en un lenguaje de programación los pasos que se deben seguir para llegar a la solución del problema. En el caso de la programación orientada a objetos estos pasos deben contener instrucciones que impliquen la interacción de objetos a través de mensajes.[4]

El uso que hacen de la computadora la mayoría de los jóvenes que recién ingresan a la carrera de Ciencias de la Computación es por lo general para trabajos escolares básicos y redes sociales y la mayoría de ellos tiene una idea muy vaga sobre cómo funciona una computadora (hardware) y qué es lo que la hace funcionar (software).

Existen un gran número de herramientas gratuitas que proveen una introducción a la programación y a la codificación de programas. Algunas de ellas son: Alice, Greenfoot, y BlueJ, las cuales representan una gran oportunidad para inculcar el pensamiento lógico en las escuelas. [Cabral2012]

2.5.1. Alice

Alice es un software educativo libre y abierto desarrollado en la Universidad Carnegie Mellon y está programado en Java. Alice se basa en un entrono de *arrastrar y soltar* para crear mundos virtuales mediante el uso de modelos en 3D. El diseño fundamental de Alice permite enseñar a los alumnos tanto la programación orientada a objetos como la orientada a eventos. El usuario va

[4]López Gaona Amparo, Introducción al desarrollo de programas con Java, 2a ed, México: UNAM, Facultad de Ciencias, 2011

creando su mundo virtual a través de la estructuración lógica de instrucciones que ya vienen pre-programadas en el IDE[5] de Alice. Este software cuenta con una gran cantidad de objetos (clases) pre-programados que el alumno puede combinar para la creación tanto de animaciones como historias en un mundo virtual. [Cabral2012]

Figura 2.1: Logotipo de Alice

Figura 2.2: Interfaz de Alice

La mayoría de los lenguajes de programación están diseñados para producir programas, cada vez más complejos. Alice está diseñado únicamente para enseñar a programar. Una ventaja del uso de Alice para introducir a los alumnos a la programación es la exigencia de la creatividad por parte del programador. El estudiante tiene que diseñar previamente en un *Story-board* su mundo virtual antes de iniciar con el proceso de creación del mismo. Otro punto a favor de Alice es que los alumnos se divierten al mismo tiempo que van, de manera intuitiva, aprendiendo los conceptos de instancia, atributos y métodos entre otras características propias de los lenguajes orientados a objetos como Java.

2.5.2. Greenfoot

Greenfoot es un ambiente interactivo de desarrollo programado en Java para propósitos educacionales. Permite la creación de aplicaciones gráficas en dos dimensiones, como simulaciones y juegos.

Greenfoot mantiene el atractivo visual y amigabilidad en cuanto a los diagramas que se pueden generar, pero su formalidad en cuanto al lenguaje es tan poderosa que permite la creación de aplicaciones desde sencillas hasta muy complejas tales como videojuegos. El estudiante puede interactuar con un editor de texto para la realización de sus proyectos.

Figura 2.3: Logotipo de Greenfoot

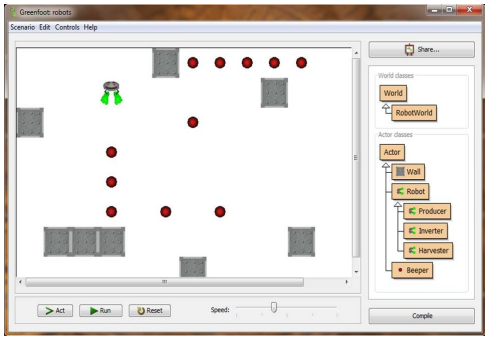


Figura 2.4: Interfaz de Greenfoot

[5]Entorno de Desarrollo Integrado *Integrated Development Environment*

2.5.3. BlueJ

Es un IDE para el lenguaje de programación Java desarrollado principalmente con propósitos educativos, pero también es adecuado para el desarrollo de software a pequeña escala.

BlueJ fue desarrollado para apoyar la enseñanza y el aprendizaje de la programación orientada a objetos. La pantalla principal muestra gráficamente la estructura de clases de una aplicación en desarrollo (en un diagrama muy parecido a UML), y los objetos pueden ser creados y probados interactivamente. Combinado con una interfaz de usuario simple, esta facilidad de interacción permite experimentar de manera sencilla con los objetos en desarrollo. Los conceptos de la orientación a objetos (clases, objetos, comunicación a través de llamadas a métodos) son representados visualmente en el diseño de interacción en la interfaz.

El enfoque pedagógico representado en BlueJ está basado en constructivismo y visualización. Trabajar con el entorno BlueJ proporciona experiencias concretas para entornos abstractos, tales como relaciones entre objetos, llamadas a métodos, y paso de parámetros.

Figura 2.5: Logotipo de BlueJ

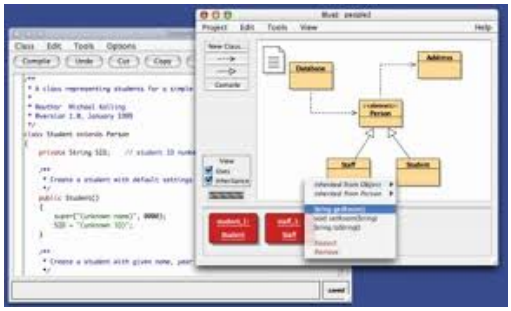


Figura 2.6: Interfaz de BlueJ

2.5.4. Sugerencias Didácticas

Con base en mi experiencia como ayudante de profesor en la materia de Introducción a las Ciencias de la Computación, he notado que la mayoría de los alumnos de primer ingreso llegan a la carrera pensando que todo se resuelve de manera mecánica y cuando entran en contacto con la programación esperan escribir programas siguiendo alguna fórmula o una serie de pasos similar a resolver ecuaciones, dada esta situación considero que algunas técnicas útiles en la enseñanza de la programación son:

- Evitar convertir los problemas en situaciones que se resuelvan de manera mecánica similar a *recetas de cocina*, ya que un pequeño cambio en el problema puede poner a los alumnos en dificultades.
- Plantear a los alumnos las mismas preguntas que uno mismo como docente se hizo al estar frente al problema por primera vez.
- No olvidar que el objetivo de un curso de programación es que el alumno sea capaz de: especificar, diseñar y analizar un algoritmo que sea correcto y eficiente a partir de un problema dado.
- Estimular la utilización de *pseudocódigo* ya que permite mantener cierta independencia del lenguaje de programación que se vaya a utilizar.

- Fomentar en los alumnos la disciplina de realizar pruebas unitarias a sus programas.

La Universidad de los Andes en Colombia, desarrolló un proyecto llamado Cupi2, el cual, desde el año 2004 tiene como objetivo el desarrollo de nuevas estrategias en el proceso de enseñanza-aprendizaje de la programación, centrándose en la motivación de los estudiantes. [Villalobos2009]

El objetivo de los cursos de programación no es únicamente que el estudiante aprenda a escribir un programa de computadora. Estos cursos deben generar una gran cantidad de habilidades en los estudiantes; ellos deben aprender a entender un problema (abstraer, modelar, analizar), a plantear soluciones efectivas (reflexionar acerca de una abstracción, definir estrategias, seguir un proceso, aplicar una metodología, descomponer en subproblemas), a manejar lenguajes de programación para expresar una solución (codificar, entender y respetar una sintaxis), a utilizar herramientas que entiendan esos lenguajes (programar, compilar, ejecutar, depurar), a probar que la solución sea válida (entender el concepto de corrección y de prueba) y a justificar las decisiones tomadas (medir, argumentar). [Villalobos2009]

El proyecto Cupi2 considera que los siguientes puntos podrían ser de utilidad en la enseñanza de la programación:

- **Aprendizaje activo**
  Lograr que el estudiante deje de ser un receptor pasivo, haciéndolo participar en clase por medio de la lectura, discusión y propuesta de soluciones.

- **Aprendizaje basado en problemas**
  Plantear y resolver problemas del mundo real para lograr que el alumno se sienta motivado al ver que lo que puede aprender en un curso de programación tiene aplicaciones en la vida real.

- **Aprendizaje incremental**
  Introducir conceptos nuevos a medida que avanza el curso no es suficiente, se recomienda poner al alumno desde el primer día de clases frente a un programa para que poco a poco vaya entendiendo las partes que forman el código y que observe lo que puede suceder si modifica este; se busca seguir con programas incompletos que el alumno tenga que completar con el uso de temas y conceptos vistos en clase, y finalmente se pretende que una vez que el alumno es capaz de crear programas simples por él mismo se puedan exponer problemas con mayor grado de dificultad.

El enfoque propuesto en Cupi2, más que una solución terminada, representa un espacio de soluciones posibles adaptables a los contextos y necesidades para los profesores de las distintas Universidades.

2.5.5. Buenas prácticas de programación

Es importante fomentar en los alumnos de un primer curso de programación buenos hábitos al momento de programar, de esta manera, estarán menos expuestos a pasar por alguna de las siguientes situaciones: [Kernighan1999]

- Desperdiciar todo un día buscando un error que debía de haber tomado máximo cinco minutos.

- Tardar demasiado tiempo en entender su propio código transcurrido cierto tiempo sin consultarlo.

- Reescribir todo un programa por no saber como añadirle una pequeña modificación.

A continuación se enlistan una serie de prácticas que se consideran recomendables al momento de programar, algunas de estas prácticas son aplicables en la programación en general y otras son exclusivas del paradigma Orientado a Objetos. [Kernighan1999]

- Es conveniente que todo programa comience con un comentario que explique su propósito, el autor, la fecha y la hora de la última modificación del mismo.
- Seleccionar nombres de variables significativas para autodocumentar el código.
- Hacer uso de paréntesis en operaciones complejas aunque estos no parezcan necesarios, con el fin de mejorar la legibilidad de la expresión.
- Hacer uso de la notación CamelCase[6] para nombres largos de variables.
- Evitar la incorporación de más de una instrucción en una sola línea.
- Cuidar que el tamaño de las sangrías sean regulares.
- Poner un espacio después de cada *coma* (,) facilita la legibilidad del código.
- En caso de usar operadores binarios, se recomienda poner una espacio a los extremos de cada operador.
- Evitar colocar comentarios de manera innecesaria en situaciones obvias.
- Colocar siempre llaves en cualquier instrucción de control.
- En un ciclo controlado por *centinela*[7], los indicadores que solicitan la introducción de datos deben recordar explítamente al usuario el valor que representa el *centinela*.
- Reutilizar software mediante el uso de métodos y otras ventajas que proporcionan las bibliotecas de un lenguaje de programación, evitando así *reinventar la rueda*.
- Utilizar mayúsculas para los nombres de las constantes de enumeración las hará resaltar y permitirá recordar que dichas contantes no son variables.

2.6. Dificultades en la enseñanza de la programación

Además de las dificultades propias que presenta impartir un curso de programación. No es raro que muchos de los alumnos que recién ingresan a la carrera de Ciencias de la Computación presenten los siguientes problemas.

- Nociones nulas de lo que es programar.
- Poca habilidad para resolver problemas.
- Razonamiento lógico poco desarrollado.
- Prejuicios de lo que implica estudiar.

[6]Estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello. Ejemplo: numero de cuenta en CamelCase sería numeroDeCuenta

[7]Valor especial de datos que indica que ya no se deben introducir más datos

- Desinterés por sus estudios en general.
- Desconocimiento de lo que estudia la Ciencia de la Computación.

Además, el nuevo grado de libertad con el que se encuentran los estudiantes así como las relaciones personales que comienzan a surgir con nuevos amigos terminan distrayendo a los alumnos de sus estudios.

Otra cuestión que puede resultar un obstáculo en la enseñanza de la programación, son los problemas de infraestructura: laboratorios mal equipados.

La mayoría de los estudiantes de nuevo ingreso vienen de escuelas públicas en las cuales no se promueve una cultura creativa para resolver problemas; el tipo de enseñanza mecanizado no desarrolla plenamente el pensamiento abstracto en el alumno. Esto provoca que los estudiantes descubran sus deficiencias, entren en confusión y pongan en duda su elección de carrera.

Capítulo 3

Diseño de Proyectos de Programación

En los primeros cursos de programación de la Facultad de Ciencias se estudia el paradigma Orientado a Objetos, el lenguaje más utilizado por lo profesores para impartir estos cursos es Java, por este motivo los proyectos presentados en este capítulo están implementados en dicho lenguaje de programación.

Cada proyecto consta de las siguientes partes:

- Título
- Objetivo
- Introducción al tema
  En esta parte se mencionan brevemente los conceptos principales que conviene presentar con mayor detalle en clase.
- Planteamiento del problema
  Es la presentación del problema, nos indica qué es lo que tenemos que resolver.
- Solución
  Es la solución al problema utilizando la metodología para resolver un problema presentada en el capítulo 2.
- Actividades sugeridas
  Son una serie de problemas similares al presentado que podrían servir de ejercicio a los alumnos para reforzar el tema.

Los proyectos que se tratan en este capítulo se eligieron con base al temario de Introducción a las Ciencias de la Computación y tienen la finalidad de reforzar los temas vistos en sesiones de clase teórica y de laboratorio.

3.1. Proyecto 1

Operadores en Java

Objetivo:
Introducir al alumno a la aritmética y precedencia de operadores en Java así como a la creación y uso de datos primitivos.

Introducción al tema

En un programa en Java es preciso asignar un identificador a cada elemento que se defina. Un identificador de variable se construye como una sucesión de caracteres alfanuméricos que inicia con letra o con guión bajo pero jamás con número. Existe un conjunto de palabras que no pueden ser usadas como identificadores porque tienen un significado especial en Java, cada una de estas se denomina *palabra reservada*.

Java es un lenguaje de programación fuertemente tipado, lo cual significa que cada variable utilizada debe tener un tipo. Los tipos definidos en Java y llamados primitivos son: byte, short, int, long, float, double, char y boolean.
A continuación se presenta una tabla que contiene una mayor descripción de los tipos primitivos en Java:

Nombre	Tipo	Tamaño	Rango
char	caracter	16 bits	0 ... 65,535
boolean	lógico	1 bit	True o False
byte	entero	8 bits	-128 ... 127
short	entero	16 bits	-32,768 ... 32,767
int	entero	32 bits	-2,147,483,648 ... 2,147,483,647
long	entero	64 bits	$-9*10^{18}$ ... $9*10^{18}-1$
float	real	32 bits	$-3.4*10^{38}$ ... $3.4*10^{38}-1$
double	real	64 bits	$-1.79*10^{308}$ ... $1.79*10^{308}-1$

A continuación se muestra una tabla que contiene las palabras reservadas de Java más usuales:

abstract	else	interface	static
boolean	extends	long	super
break	false	main	switch
byte	final	native	synchronized
case	finaly	new	this
catch	float	null	throw
char	for	package	throws
class	if	private	true
continue	implements	protected	try
default	import	public	void
do	instanceof	return	volatile
double	int	short	while

Es recomendable elegir nombres de variables apropiadas relacionadas con el concepto que representan para que el código sea más entendible tanto para el programador como para quien lea el

programa.
Java cuenta con operadores tanto aritméticos, relacionales y lógicos a continuación se muestra una tabla que describe cada uno de estos operadores

Operadores aritméticos	
Operador	Nombre
+	suma
-	resta
*	multiplicación
/	división
%	módulo
Operadores relacionales	
Operador	Nombre
<	menor que
>	mayor que
<=	menor o igual
>=	mayor o igual
! =	distinto
==	igual
Operadores lógicos	
Operador	Nombre
&	AND
|	OR
!	NOT

Las expresiones aritméticas en Java deben escribirse en un mismo renglón. Es decir, la expresión *a divide a b* debe escribirse como *a/b*.
Java aplica los operadores en expresiones aritméticas en una secuencia precisa, determinada por las siguientes reglas de precedencia de operadores.

1. Las operaciones de multiplicación, división y residuo se aplican primero. Si una expresión contiene varias de esas operaciones, los operadores se aplican de izquierda a derecha.

2. Las operaciones de suma y resta se aplican después de las operaciones de multiplicación, división y residuo. Si una expresión contiene varias de esas operaciones, los operadores se aplican de izquierda a derecha.

La expresión 3+5*8 es igual a 43, primero se efectúa el producto y hasta el último la suma, si quisiéramos que primero se evaluara la suma y al final el producto debemos colocar paréntesis (3+5)*8 obteniendo como resultado 64.

Es importante mencionar que para realizar cálculos más avanzados como una raíz cuadrada o una potencia, o bien, hacer uso de constantes matemáticas existe la clase *Math*[1] que se importa por defecto en cada clase escrita en Java.

Planteamiento del problema

Escribe un programa que calcule las siguientes fórmulas matemáticas e imprima en pantalla el resultado de cada una de ellas.

[1] http://docs.oracle.com/javase/6/docs/api/java/lang/Math.html

Considera los siguientes valores:
x=5
y=2
z=4

- $\frac{x+3}{5y^2} - \sqrt{y} + 10$
- $\sqrt{4x + 9y} + (\frac{x^5}{y^2})^2$
- $(\frac{6x+2z}{2})^5$
- $\sqrt{4x - 5yz} + 9y - \frac{1}{x}$
- $\frac{x+y}{z}^2 * \frac{x^3}{(y+2)^5}$

Solución

1. Entender bien el problema
   Se solicita que dadas tres variables *x=5, y=2, z=4* se calculen cinco fórmulas algebraicas y se muestre el resultado de cada uno de los cálculos en pantalla. Como podemos ver, la entrada de nuestro programa son esas tres variables las cuales pasan por el proceso de evaluación en cada respectiva fórmula y la salida son los resultados de cada una de esas fórmulas.

2. Diseñar el algoritmo que resuelva el problema
   Definir variables x,y,z
   Definir variable resultado
   Para cada fórmula
   resultado = fórmula algebraica
   Imprimir resultado

3. Implementar de ser posible, el algoritmo de solución.
   El código fuente de este proyecto se encuentra en el apéndice A.

4. Realizar pruebas al código implementado
   Para probar nuestros programas podemos realizar pruebas unitarias, pero este tipo de pruebas se realizarán cuando estemos trabajando con la creación de objetos. Así que para probar el código anterior bastará con hacer un par de pruebas a cada fórmula sustituyendo las variables con números y efectuando los cálculos correspondientes en papel, o haciendo uso de la calculadora.

Actividades sugeridas

- Dadas las siguientes declaraciones de variables:
  int x,y;
  float z = 3.1313f;
  boolean verdad = true;
  long l = 45L, m;

  Evaluar las siguientes expresiones siempre y cuando sea posible. En caso de no ser posible la evaluación justificar la respuesta.
  a) x = 6;
  b) y = 1000;
  c) y = 2.3333;

d) 25++;
e) w = 175;
f) verdad = 1;
g) z = 3.1416;
h) m = x * 250;
i) (x+y)++;
j) y = 1;

- Si el costo de la gasolina es de $7.50 por litro, escribir una expresión que defina el valor de la variable real *total* como el total a pagar por n litro de gasolina.

- ¿Cuáles de las siguientes cadenas no pueden ser identificadores en Java? ¿Porqué?
  a) java.awt.Graphics
  b) rayos-x
  c) _123
  d) valor calculado
  e) valorCalculado
  f) void
  g) Void
  h) 2dias
  i) segundaBase
  j) x

3.2. Proyecto 2

El problema de *3n+1*

Objetivo:
Presentar al alumno las instrucciones de control en Java las cuales son imprescindibles en el desarrollo de programas.

Introducción al tema

El trabajo de Bohm y Jacopini demostró que todos los programas podían escribirse en términos de tres estructuras de control solamente: la **estructura de secuencia**, la **estructura de selección** y la **estructura de repetición**. [Bohm1966]

Estructura de secuencia en Java
La computadora ejecuta las instrucciones en Java una después de la otra, en el orden en que estén escritas; es decir, en secuencia.

Estructura de selección en Java
Java tiene tres tipos de instrucciones de selección:

- Instrucción de selección simple *if*
  Esta instrucción realiza (selecciona) una acción si la condición es verdadera, o evita la acción si la condición es falsa.

- Instrucción de selección *if...else*

Esta instrucción realiza una acción si la condición es verdadera o realiza otra acción si la condición es falsa.

- Instrucción de selección múltiple *switch*
  Esta instrucción selecciona una acción entre diversas opciones.

Estructura de repetición en Java

Java cuenta con tres instrucciones de repetición (también llamadas instrucciones de ciclo) que permiten a los programas ejecutar instrucciones en forma repetida, siempre y cuando una condición (llamada la condición de continuación del ciclo) siga siendo verdadera.

- Las instrucciones de control *while* y *for*
  Realizan la acción o grupo de acciones en sus cuerpos, cero o más veces; si la condición de continuación del ciclo es inicialmente falsa, no se ejecutará la acción.

- La instrucción de control *do...while*
  Realiza la acción o grupo de acciones contenidas en el cuerpo de la instrucción una o más veces.

Planteamiento del problema

Consideremos el siguiente algoritmo para generar una secuencia de números. Comenzando con un entero n: si n es par, se divide entre 2; si n es impar, se multiplica por 3 y se le suma 1. Este proceso se debe repetir para cada nuevo valor de n, finalizando cuando $n=1$. Por ejemplo, para $n=22$ se genera la siguiente secuencia de números:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

La cantidad de números desde n hasta 1 se conoce como longitud de ciclo; para $n=22$ la longitud de ciclo es 16. Se conjetura (aunque no está demostrado) que este algoritmo termina en $n=1$ para cualquier entero n. Dicha conjetura se cumple, al menos para cualquier entero hasta 1,000,000.

Dados dos números cualesquiera, i y j, se debe determinar la máxima longitud de ciclo correspondiente a un número comprendido entre i y j, incluyendo ambos extremos.

Solución

1. Entender bien el problema

   Se tiene un intervalo de i a j, para cada uno de esos números en el rango, si el valor actual es par, se debe dividir entre 2, si es impar, se debe multiplicar por 3 y sumar 1, al número total de operaciones que se realizan para llegar a la unidad se le conoce como longitud de ciclo. Lo que se nos pide encontrar es un valor entero que indica la longitud de ciclo mayor para un entero comprendido en el rango de i a j.

   La entrada consta de dos enteros i y j los enteros serán menores de 1,000,000 y mayores a 0. Se debe cumplir que i<=j. La salida debe ser el valor entero s tal que s es la máxima longitud de ciclo para algún entero k tal que $i<=k<=j$.

2. Diseñar el algoritmo que resuelva el problema

   Leer valor de i
   Leer valor de j

```
temporal=0
maxima=0
contador=0
Mientras i<=j
        temporal=i
        contador=1
        Mientras temporal>1
                si temporal es par
                        temporal=temporal/2
                        contador++
                si no
                        temporal=temporal*3+1
                        contador++
        Si contador>maxima
                maxima=contador
Imprimimos el valor de maxima
```

No es posible garantizar que el algoritmo para resolver **el problema 3n+1** siempre termine, dado que a la fecha no se ha podido demostrar que para cualquier n exista una longitud de ciclo finita.

3. Implementar el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice B.

4. Realizar pruebas al código implementado
   Podemos probar nuestro código con varios ejemplos, y también darnos cuenta de que nuestro ciclo *while* más externo recorre el intervalo desde i hasta j, mientras que el ciclo interno, toma el valor actual de i y realiza los calculos correspondientes de dividirlo entre 2 en caso de que sea par o multiplicarlo por 3 en caso contrario hasta llegar al número 1.

Actividades sugeridas

- Escribe un programa que calcule todos los números primos contenidos en el rango de 2 a 1,000
- Un número perfecto es un número natural que es igual a la suma de sus divisores propios positivos, sin incluirse él mismo. Así, 6 es un número perfecto porque sus divisores propios son 1, 2 y 3; y $6 = 1 + 2 + 3$.
  Escribe un programa que permita calcular todos los números perfectos en el rango de 1 a 10,000

3.3. Proyecto 3

El sistema binario

Objetivo:
El alumno aprenderá a segmentar un programa, a través del uso de métodos.

Introducción al tema

Métodos en Java
Los métodos (también conocidos como funciones o procedimientos en otros lenguajes) permiten

al programador dividir un programa en módulos, por medio de la separación de sus tareas en unidades autónomas. Las instrucciones en los cuerpos de los métodos se escriben sólo una vez, y se pueden reutilizar desde varias ubicaciones en un programa.

Una razón para dividir un programa en módulos mediante el uso de métodos es la metodología *divide y vencerás*, que hace que el desarrollo de un programa sea más fácil de administrar.

Un método se invoca mediante una llamada, y cuando el método que llamó completa su tarea, devuelve un resultado o simplemente el control al método que lo llamó. Hay tres formas de llamar a un método.

- Utilizando el nombre de un método por sí sólo para llamar a otro método de la misma clase.
- Utilizando una variable que contiene una referencia a un objeto, seguida de un punto (.) y del nombre del método para llamar a un método del objeto al que se hace referencia.
- Utilizando el nombre de la clase y un punto(.) para llamar a un método *static* de una clase.

Hay ocasiones en las que un método necesita información adicional para realizar su labor, estos datos se conocen como parámetros, los parámetros pueden ser tipos primitivos o bien, tipos referenciados.

Planteamiento del problema

El sistema de numeración decimal, también llamado sistema decimal, es un sistema de numeración posicional en el que las cantidades se representan utilizando como base aritmética las potencias del número diez. El conjunto de símbolos utilizado (sistema de numeración arábiga) se compone de diez cifras diferentes: cero (0), uno (1), dos (2), tres (3), cuatro (4), cinco (5), seis (6), siete (7), ocho (8) y nueve (9).

El sistema binario es un sistema de numeración en el que los números se representan utilizando solamente las cifras cero y uno (0 y 1). Es el que se utiliza en las computadoras, debido a que trabajan internamente con dos niveles de voltaje, por lo cual su sistema de numeración natural es el sistema binario (encendido 1, apagado 0). Es posible realizar conversiones entre ambos sistemas de numeración.

Se desea programar una aplicación capaz de convertir números en base 10 a base 2 y viceversa. Para poder realizar estas operaciones se deberá mostrar un menú con las siguientes opciones:

1. Convertir un número en base 10 a base 2
2. Convertir un número en base 2 a base 10

Solución

1. Entender bien el problema

   Se desea resolver un problema que consta de dos partes: La primera es convertir un número en base 10 a base 2, la segunda parte es convertir un número en base 2 a base 10.

2. Diseñar el algoritmo que resuelva el problema

   Para resolver la primera parte del problema (Conversión Sistema Decimal - Sistema Binario), se conoce un algoritmo que realiza esta labor el cual consiste en hacer dividir el número original entre 2, luego dividir el cociente de esa división entre 2 y así sucesivamente ir diviendo el cociente de las divisiones entre 2 hasta que el cociente sea 0, una vez que se llega a este

punto, se toman todos los residuos de las divisiones desde último hasta el primero y lo que se obtenga es la representación binaria del número decimal.

decimal-binario(numeroDecimal):
cadena
Mientras numeroDecimal >= 0
residuo = numeroDecimal %2
concatenar a cadena el residuo
numeroDecimal=numeroDecimal/2
Calcular y regresar la reversa de cadena

La segunda parte del problema es convertir un número del sistema binario a su representación decimal, y para ello, existe otro algoritmo que nos dice que debemos tomar dígito a dígito el número binario de derecha a izquierda y multiplicarlo por 2^n desde *n=0* incrementando el valor de n por cada dígito que avancemos en el número binario que estamos recorriendo, finalmente sumamos el resultado de todas las multiplicaciones que se hayan realizado.

binario-decimal(numeroBinario):
n=0
suma=0
i=cantidad de dígitos de numeroBinario-1
Mientras i >= 0
j = (caracter de número binario en posición i) 2^n
suma=suma+j
n++
i=i-1
regresar suma

3. Implementar el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice C.

4. Realizar pruebas al código implementado
   En programación, una prueba unitaria es una forma de probar el correcto funcionamiento de un fragmento de código. Esto sirve para asegurar que cada uno de nuestros métodos funcione correctamente por separado.

JUnit es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. JUnit es un framework que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. [Gamma2006]

Para poder realizar pruebas a nuestro código usando JUnit debemos reunir nuestras clases en paquetes. Un paquete en Java es un contenedor de clases que permite agrupar las distintas partes de un programa cuya funcionalidad tienen elementos comunes. En los archivos de código Java se usa la palabra reservada *package* para especificar a qué paquete pertenecen, debe

indicarse como primera sentencia. Para usar un paquete dentro del código se usa la declaración *import*.

Para hacer uso de JUnit, utilizaremos *Apache Ant.*[2] Además, tendremos que crear una clase donde realicemos las pruebas unitarias de cada uno de los métodos de nuestro proyecto, esta clase que contiene las pruebas unitarias también se encuentra en el apéndice C.

Es importante mencionar que en esta tesis haremos uso de pruebas unitarias para la realización de pruebas a nuestros programas, aunque existen diversas herramientas de gran utilidad para este propósito. (Referencias bibliográficas sobre Sofware Testing [MYERS2011], [Hetzel1988]).

Actividades sugeridas

- Escribir un programa que juegue dados con las siguientes reglas: Un jugador tira dos dados. Se calcula la suma de los puntos de las caras superiores. Si la suma es 7 u 11 en el primer tiro, el jugador gana. Si la suma es 2, 3 o 12 en el primer tiro, el jugador pierde, si la suma es 4, 5, 6, 8, 9 o 10 en el primer tiro, esta suma se convierte en el punto del jugador. Para ganar el jugador debe seguir tirando los dados hasta que salga otra vez su punto. El jugador pierde si tira un 7 antes de llegar a su punto.
- Escribe una aplicación que juegue a adivinar el número de la siguiente manera: Su programa elige el número a adivinar de forma aleatoria en el rango de 1 a 1,000. La aplicación muestra el mensaje "Adivine un número entre 1 y 1,000". El jugador escribe su primer intento, si la respuesta del jugador es incorrecta, su programa debe mostrar el mensaje: "Tú número es más alto que el mío, intenta de nuevo" o "Tú número es más bajo que el mío, intenta de nuevo", y solicitar un nuevo valor. Si el jugador adivina el número, el programa debe imprimir: "Adivinaste". Este programa debe dar cómo máximo 10 intentos para adivinar el número.

3.4. Proyecto 4

Objetos en Java

Objetivo:
Introducir al alumno a la creación de objetos en Java.

Introducción al tema

Creación de objetos en Java

El primer paso para trabajar con un objeto en Java es su creación. Para crear un objeto se requiere utilizar el operador **new** seguido del nombre de la clase a la que pertencerá tal objeto.

Una referencia es la intancia de una clase. El valor que se almacena en una variable o constante de tipo referencia no es el objeto en sí, es la dirección o referencia del objeto. Una vez creado un objeto se le pueden enviar mensajes, ya sea para conocer su estado, modificar su estado, realizar algún cálculo, etc. [Barnes2007]

Existe un tipo especial de métodos llamados *métodos Constructores* que permiten inicializar los atributos de un objeto, o bien, cada que se crea un objeto se hace una llamada a este tipo de

[2] http://ant.apache.org/

método.

Planteamiento del problema

Crea una clase llamada *Recta* que servirá para representar una recta en el plano. La representación debe ser de forma analítica, es decir, se debe representar una recta de la forma: $y=mx+b$. Los métodos que debes incluir en tu clase deben ser:

- public Recta(double m, double b)
  Constructor que inicializa los valores de la recta (pendiente m y ordenada b)
- public boolean esta(Punto p)
  Regresa *true* si el punto p es parte de la recta desde donde se invoca el método, en otro caso regresa *false*.
- public boolean esIgual(Recta r)
  Regresa *true* si la recta desde la que se invoca al método es igual a la recta r, en caso contrario regresa *false*.
- public boolean esParalela(Recta r)
  Regresa *true* si la recta desde la que se invoca al método es paralela a la recta r, en caso contrario regresa *false*.
- public boolean esOrtogonal(Recta r)
  Regresa *true* si la recta desde la que se invoca al método es ortogonal a la recta r, en caso contrario regresa *false*.
- public Punto intersecta(Recta r1)
  Método que regresa el punto de intersección de la recta *r1* con la recta que invoca el método.
- public Recta ortogonal(Punto p)
  Regresa una recta ortogonal a la recta que invoca el método y que pasa por el punto p.

Solución

1. Entender bien el problema

   Debemos desarrollar un programa en Java que nos permita trabajar con rectas y con algunas de las operaciones asociadas a ellas, para tal propósito se nos da una serie de métodos que debemos implementar.

2. Diseñar el algoritmo que resuelva el problema

   En primer lugar podemos darnos cuenta que necesitaremos crear dos clases: la clase *Recta* y la clase *Punto*, ya que podemos observar que algunos de los métodos solicitados tienen como argumento un objeto de tipo Punto.

   ¿Cómo sabemos que un punto pertenece a una recta?
   Se dice que un punto $P(x1, y1)$ pertenece a una recta $y = mx+b$ si al sustituir las coordenadas del punto P en la ecuación se cumple que $y1 = mx1 + b$

   Para revisar que dos rectas son iguales debemos revisar que tengan la misma pendiente y la misma ordenada al origen.

   Dos rectas son ortogonales cuando el producto de sus pendientes es igual a -1.

   Dos rectas son paralelas cuando tienen la misma pendiente.

   Para obtener el punto de intersección $P(x1, y1)$ de dos rectas, debemos igualar las dos rectas que son de la forma $y = mx + b$ y despejar el valor de x, de esta forma obtenemos el punto

$x1$, luego sustituimos este valor en cualquieras de la ecuaciones para obtener $y1$.

Para calcular la ecuación de una recta R_1 que es ortogonal a otra recta R_2 podemos notar primeramente que la pendiente de R_1 multiplicada por la pendiente de R_2 debe ser igual a -1; es decir $pendiente R_1 * pendiente R_2 = -1$, despejando $pendiente R_1$ obtenemos $pendiente R_1 = -1/pendiente R_2$. Entonces ahora para calcular la ordenada al origen de la recta R_1 partimos del hecho de que la ecuación de esta recta es de la forma $y = mx + b$, despejamos el valor de b y obtenemos $b = y - mx$, lo cual podemos resolver sustituyendo las coordenas del punto $P(x1, y1)$ por el que pasa R_1 así como el valor de $pendiente R_1$

3. Implementar de ser posible, el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice D.

4. Realizar pruebas al código implementado
   El código de pruebas también se encuentra en el apéndice D.

Actividades sugeridas

- Crea un programa en Java que maneje objetos de tipo *Racional* e implementa los siguientes métodos:
  - public Racional suma(Racional r)
    Este método calcula la suma de dos objetos racionales.
  - public Racional resta(Racional r).
    Este método calcula la resta de dos objetos racionales. El que hace la llamada **menos** el de la lista de parámetros.
  - public Racional producto(Racional r)
    Este método calcula el producto de dos objetos racionales.
  - public Racional cociente(Racional r)
    Este método calcula la división de dos objetos racionales. El que hace la llamada **entre** el de la lista de parámetros.
  - public Racional simplifica(Racional r)
    Este método reduce un objeto Racional a su mínima expresión.
  - public Racional mayor(Racional r)
    Este método compara dos objetos racionales y regresa el de mayor valor.
  - public Racional iguales(Racional r)
    Este método regresa *true* en caso de que dos objetos racionales sean iguales.
- Escribe un programa que simule el comportamiento de un robot.
  El robot tiene nombre, un número de serie y un estado que puede ser encendido o apagado. Crea la clase *Robot.java* que contenga los métodos necesarios para crear un robot, apagarse y encenderse, además el robot es capaz de: hacer operaciones aritméticas simples(suma, resta, multiplicación y división) y saludar diciendo su nombre y número de serie.
  Siempre que un robot acaba de ser creado su estado inicial es apagado.
  Escribe una clase *PruebaRobot.java* donde crees 3 objetos de tipo Robot y mándales instrucciones que sepan hacer. Es importante que tengas en cuenta que mientras el robot está apagado no atiende ningún mensaje a menos que sea encender.

3.5. Proyecto 5

Arreglos

Objetivo:

Introducir al alumno a la creación y manejo de arreglos en el lenguaje de programación Java.

Introducción al tema

Los arreglos son objetos que almacenan elementos del mismo tipo; son entidades de longitud fija; permanecen con la misma longitud una vez que se crean. Los elementos de un arreglo pueden ser tipos primitivos o tipos por referencia (incluyendo arreglos).

Para crear un objeto tipo arreglo, el programador especifica el tipo de los elementos del arreglo y la cantidad de elementos como parte de una expresión de creación de arreglo, que utiliza la palabra clave *new*. La siguiente expresión de creación de arreglo crea un arreglo de 100 valores int:
int[] b = new int[100];

Cuando se crea un arreglo, cada elemento del mismo recibe un valor predeterminado: cero para los elementos numericos de tipo primitivo, *false* para los elementos booleanos, *null* para las referencias y '\u0000' para las variables de tipo char.

Todo objeto tipo arreglo conoce su propia longitud y mantiene esta información en un campo **length**. La manera de acceder a cada elemento de un arreglo es através de sus índices, los cuales van desde 0 hasta la longitud del arreglo menos uno.

Planteamiento del problema

Una pequeña aerolínea acaba de comprar una computadora para su nuevo sistema de reservaciones automatizado. Se te ha pedido que escribas una aplicación para asignar asientos en cada vuelo del único avión con el que cuenta la aerolínea (capacidad: 10 asientos).

Tu aplicación deberá mostrar las siguientes alternativas: *Por favor escriba 1 para primera clase* y *Por favor escriba 2 para clase economica*. Si el usuario escribe 1 tu aplicación le deberá mostrar los asientos disponibles del 1 al 5, si el usuario escribe 2 tu aplicación deberá mostrarle los asiento desocupados del 6 a 10.

Una vez que se elige un asiento se debe preguntar el nombre del pasajero. Tu aplicación deberá imprimir entonces un pase de abordaje, indicando el nombre de la persona y número de asiento y si se encuentra en la sección de primera clase o clase económica del avión. Tu programa nunca deberá asignar un asiento que ya esté ocupado. Cuando esté llena la sección económica, tu programa deberá preguntar a la persona si acepta ser colocada en la sección de primera clase (y viceversa). Si la persona acepta, se deberá realizar la asignación de asiento apropiada. Si no acepta, imprime el mensaje *El próximo vuelo sale en 3 horas*.

El vendedor de boletos puede consultar en cualquier momento la información de cada asiento, si este está disponible o el nombre de la persona que lo compró. Además, se permite que un cliente cancele su vuelo, en caso de hacer eso, se pondrá ese lugar como desocupado y podrá comprarlo alguien más.

Solución

1. Entender bien el problema

   Podemos simular la venta de boletos de una aerolínea mediante un arreglo que representará asientos que serán ocupados por personas, los asientos se dividen en dos clases: primera clase y clase económica. Cada vez que que se realice una venta se debe marcar el asiento como ocupado y además solicitar el nombre de la persona que compró dicho asiento, con la finalidad de que el vendedor pueda consultar la información de los lugares del avión y así mismo, imprimir el ticket de compra.

   Es posible comprar boletos en la clase económica si los de la primera clase se agotaron y

viceversa. También será posible cancelar un boleto en caso de que ya se haya comprado.

2. Diseñar el algoritmo que resuelva el problema

   Vamos a necesitar una clase *Persona* para modelar a los clientes. Ya que el nombre es lo único que nos interesa para este problema esta clase sólo contendrá el atributo nombre.

   Vamos a necesitar un método que funcione como menú para realizar las actividades del vendedor de boletos de la aerolínea:

menu
Mientras la opcion no sea salir:
    Mostrar opciones
        vender boleto(Primera clase o clase económica)
        cancelar boleto
        consultar asientos
    Leer opción

A continuación se detallan cada una de las opciones posibles:

vender boleto:
    imprimir: Por favor escriba 1 para primera clase
    imprimir: Por favor escriba 2 para clase económica
    Leer opción
    Si opción es 1:
        Revisar si hay lugares en sección 1
        Si hay lugares en sección 1
            llamar método de venta en primera clase
        Si no hay lugares en sección 1
            revisar si hay lugar en sección 2
            Si hay lugar en sección 2
                preguntar si compra en sección 2
                Si elige comprar en sección 2
                    llamar método de venta en clase económica
                Si no elige comprar en sección 2
                    imprimir: El próximo vuelo sale en 3 horas
            Si no hay lugar en sección 2
                imprimir: Lo sentimos, se agotaron los boletos
                imprimir: El próximo vuelo sale en 3 horas
    Si opción es 2:
        Revisar si hay lugares en sección 2
        Si hay lugares en sección 2
            llamar método de compra en clase económica
        Si no hay lugares en sección 2
            revisar si hay lugar en sección 1
            Si hay lugar en sección 1
                preguntar si compra en sección 1
                Si elige comprar en sección 1
                    llamar método de compra en primera clase
                Si no elige comprar en sección 1
                    imprimir: El próximo vuelo sale en 3 horas
            Si no hay lugar en sección 1
                imprimir: Lo sentimos, se agotaron los boletos
                imprimir: El próximo vuelo sale en 3 horas

Si opción es diferente de 1 o 2:
imprimir: Introduce una opción válida

Método de venta de boletos en primera clase:

venderBoletoPrimeraClase
mostrar asientos desocupados en primera clase
leer número de asiento
Si número de asiento es válido
Si el asiento está desocupado
preguntar nombre pasajero
asignar pasajero a asiento en avión
llamar método que imprime boleto
Si el asiento no está desocupado
imprimir: Ese asiento ya se encuentra ocupado
Si no es válido
imprimir: Asiento inválido

Método de venta de boletos en clase económica:

venderBoletoClaseEconomica
mostrar asientos desocupados en clase economica
leer número de asiento
Si número de asiento es válido
Si el asiento está desocupado
preguntar nombre pasajero
asignar pasajero a asiento en avión
llamar método que imprime boleto
Si el asiento no está desocupado
imprimir: Ese asiento ya se encuentra ocupado
Si no es válido
imprimir: Asiento inválido

Método para cancelar boletos:

cancelarBoleto:
solicitar nombre
solicitar número de asiento
Si en el asiento está el nombre leído
asignar null al asiento
imprimir: El boleto ha sido cancelado
Si no está
imprimir: Datos incorrectos
imprimir: No se pudo cancelar el boleto

Método para ver la información de los asientos del avión:

verTodo:

imprimir: Información del avión
Recorrer todo el arreglo del avion desde i=0
Si i>= 0 y i<= 4
imprimir: Primera clase
Si i>= 5 y i<= 9
imprimir: Clase económica
Si la posición i del avión es diferente de null
imprimir la información del avión en esa posición
Si no
imprimir: número de asiento Desocupado

Método que imprime boleto:

imprimirBoleto:
Si asiento < 5
*imprimir: ************************
imprimir nombre del pasajero
imprimir número de asiento
imprimir: PRIMERA CLASE
imprimir: Gracias por preferir nuestra aerolinea
*imprimir: ************************
Si no
*imprimir: ************************
imprimir nombre del pasajero
imprimir número de asiento
imprimir: CLASE ECONOMICA
imprimir: Gracias por preferir nuestra aerolinea
*imprimir: ************************

3. Implementar de ser posible, el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice E.

4. Realizar pruebas al código implementado
   Como el flujo del programa depende de la información proporcionada por el usuario desde la entrada estándar y es posible que el estado de los asientos del avión cambie continuamente debido a la posibilidad de cancelar boletos, por esta ocasión no se realizarán pruebas unitarias y se considerará la ejecución del programa como un conjunto de pruebas a medida que se interactúa con él.

Actividades sugeridas

- Escribir un programa que simule votaciones.
  Se tienen 3 candidatos: Hugo, Paco y Luis, el programa debe de generar 1000 votos de manera aleatoria; el 1 indica que es un voto para Hugo, el 2 es un voto para Paco y el 3 un voto para Luis. En caso de que haya un empate entre dos candidatos se debe generar un voto aleatorio que los desempate. Finalmente se debe mostrar quien ganó las elecciones.

- Tiro de dados
  Escribe una aplicación para simular el tiro de dos dados. La aplicación debe utilizar un objeto de la clase *Random* una vez para tirar el primer dado, y de nuevo para tirar el segundo dado,. Después debe calcularse la suma de los dos valores. Cada dado puede mostrar un valor

entero del 1 al 6, por lo que la suma de los valores variará del 2 al 12. El programa debe tirar los dados 36,000 veces. Utiliza un arreglo unidimensional para registrar el número de veces que aparezca cada una de las posibles sumas.

- Simulación de la tortuga y la liebre

  En este problema de recreará la clasica carrera de la tortuga y la liebre. Nuestros competidores empezarán la carrera en la posición 1 de 70, el primero en llegar a la posición 70 ganará una cubeta de zanahorias y lechugas frescas. El recorrido se abre paso hasta la cima de una resbalosa montaña por lo que ocasionalmente los competidores pierden terreno.

  A continuación se muestra una tabla con las reglas que ajustan las posiciones de la tortuga y la liebre.

Animal	Tipo de movimiento	Porcentaje del tiempo	Acción
Tortuga	paso rápido	50	avanza 3 posiciones
	resbalón	20	retrocede 6 posiciones
	paso lento	30	avanza 1 posición
Liebre	dormir	20	ningún movimiento
	gran salto	20	avanza 9 posiciones
	gran resbalón	10	retrocede 12 posiciones
	pequeño salto	30	avanza 1 posición
	pequeño resbalón	20	retrocede 2 posiciones

Empieza la carrera imprimiendo el mensaje
PUM!!!
Y ARRAACAN!!!

Tu programa deberá imprimir durante su ejecución el estado de la carrera en cada momento mientras no haya ganador, utiliza la letra T para indicar la posición de la tortuga y una L para indicar la posición de la liebre. Una vez que haya un ganador imprime su nombre.

3.6. Proyecto 6

Listas

Objetivo:

Que el alumno aprenda el manejo de listas mediante el uso de la estructura *ArrayList* proporcionada por la API[3] de Java.

Introducción al tema

Conjuntos

Un conjunto es una colección bien definida de objetos, entendiendo que dichos objetos pueden ser cualquier cosa: números, personas, letras, otros conjuntos, etc. Algunos ejemplos son:

- A es el conjunto de las letras del alfabeto latino
- B es el conjunto de cartas de la baraja española
- C es el conjunto de los planetas del Sistema Solar

[3] http://docs.oracle.com/javase/7/docs/api/

Los conjuntos pueden ser finitos o infinitos. El conjunto de los números naturales es infinito, pero el conjunto de los días de la semana es finito.

ArrayList en Java

La clase ArrayList es un objeto que actúa como una lista que implementa la interfaz Collection de Java. Esta clase permite contener y ordenar objetos, incluso, puede almacenar objetos duplicados. Su tamaño es dinámico, es decir, esta lista crecerá a medida que se inserten en ella más elementos. Debememos recordar que el primer elemento del ArrayList tiene como índice el 0.

De forma general un ArrayList en Java se crea de la siguiente forma:
ArrayList nombreArray = new ArrayList();
Un ArrayList declarado así puede contener objetos de cualquier tipo, o bien, objetos de varios tipos.

Algunos métodos que proporciona ArrayList son:

MÉTODO	DESCRIPCIÓN
size()	Devuelve la cantidad de elementos contenidos en el ArrayList
add(x)	Añade un elemento al final
add(posicion,x)	Añade el elemento x en la posición indicada
get(posicion)	Devuelve el elemento que está en la posición indicada
remove(posicion)	Elimina el elemento que se encuentra en la posición indicada
remove(x)	Elimina la primera ocurrencia del elemento x
clear()	Elimina todos los elementos del ArrayList
set(posicion, x)	Sustituye el elemento de la posición indicada por el elemento x
contains(x)	Verifica si el elemento x está contenido en el ArrayList
indexOf(x)	Devuelve la posición del objeto X. Si no existe devuelve -1

Métodos estáticos en Java

Los métodos que no requieren de ningún objeto para ser llamados se llaman métodos *static*.

Planteamiento del problema

Se desea programar una aplicación que trabaje con conjuntos, nuestro Universo estará formado por números enteros que no excedan el límite del rango de un número entero en el lenguaje de programación Java.

Se deberá hacer uso de la clase ArrayList de Java para representar a nuestros conjuntos como listas, además se deberán implementar métodos *static* para realizar cada una de las siguientes operaciones:

- Unión
- Intersección
- Diferencia
- Cardinalidad
- Pertenencia

Solución

1. Entender bien el problema
   Se desea programar una aplicación que trabaje con conjuntos, estos conjuntos serán representados por medio de listas, para trabajar con listas haremos uso de la clase ArrayList de Java. Debemos realizar operaciones básicas con los conjuntos (unión, intersección...) mediante el uso de métodos de tipo *static*.

2. Diseñar el algoritmo que resuelva el problema
   A continuación se muestra el algoritmo de cada método:

   Método de la operación unión (recibe dos conjuntos)
   Crear una lista UNION
   Recorrer todos los elementos del primer conjunto y agregarlos a UNION
   Recorrer todos los elementos del segundo conjunto y si no están en UNION añadirlos
   regresar UNION

   Método de la operación intersección (recibe dos conjuntos)
   Crear una lista INTERSECCION
   Para cada elemento x en el primer conjunto
   si x está en el segundo conjunto
   agregar x a INTERSECCION
   regresar INTERSECCION

   Método de la operación diferencia (recibe dos conjuntos)
   Crear un conjunto DIFERENCIA
   Para cada elemento x en el primer conjunto
   Si x no está contenido en el segundo conjunto
   Agregar x a DIFERENCIA
   regresar DIFERENCIA

   Para los métodos de cardinalidad y pertenecia podemos usar los métodos *size()* y *contains(x)* que proporciona la clase ArrayList de Java.

3. Implementar de ser posible, el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice F.

4. Realizar pruebas al código implementado
   El código de las pruebas unitarias se encuentran también en el apéndice F.

Actividades sugeridas

- Escribe un programa en Java que genere una lista de longitud 15 que contenga valores enteros en el rango de 0 a 25 generados de manera aleatoria, ordena los elementos de la lista utilizando el algoritmo *Quick Sort*. [4]

- Escribe un programa en Java que simule una pastelería, para esto deberás tener las siguientes clases:

  - Pastel
    Esta clase modela un pastel, el cual consta de sabor (fresa, zarzamora, piña o durazno) y fecha de elaboración.

[4] Cormen, Thomas H., et al. Introduction to algorithms. Vol. 2. Cambridge: MIT press, 2001., pag. 145

- Fecha
  Esta clase deberá modelar la fecha de elaboración de un pastel en formato:
  dd/mm/aaaa

Tu programa deberá generar 50 pasteles con sabor y fecha de elaboración aleatorios (cuidando que la fecha sea menor o igual al día actual). Si la fecha de elaboración del pastel es igual al día actual o menor por 1 día el precio del pastel deberá ser de $150, si la fecha de elaboración del pastel es menor a la fecha actual por 2 o 3 días el precio del pastel será de $100. Si un pastel excede de más de 3 días desde la fecha de su elaboración esté será desechado y representará una pérdida de $80.

Tu programa debe permitir al usuario hacer lo siguiente:

- 1. Consultar los pasteles frescos ($150)
- 2. Consultar los pasteles en rebaja ($100)
- 3. Consultar los pasteles que ya caducaron
- 4. Salir

Las opciones 1 a 3 mostrarán un listado de cada pastel de la forma:

Pastel de **sabor** elaborado el **Fecha**

La opción 3 deberá incluir además, la suma total de dinero ocasionado por las pérdidas.
La opción 4 termina con la ejecución del programa.

- Escribe un programa en Java que permita llevar el control de altas y bajas de los libros en una biblioteca.

Cada libro consta de ISBN, título, autor, tema, año, edición y número de páginas.
Debe ser posible que el bibliotecario realize lo siguiente:

- Registrar un nuevo libro
- Buscar libros por autor
- Buscar libros por tema
- Buscar libro por título
- Ver todos los libros
- Borrar un libro

Puede haber más de un ejemplar de un mismo libro. Borrar un libro solicita solamente el ISBN y elimina únicamente una pieza en caso de haber más de un sólo ejemplar.

3.7. Proyecto 7

Arreglos de arreglos

Objetivo:
El objetivo de este proyecto es que una vez que el alumno ha aprendido arreglos de una dimensión aprenda lo que son las matrices o arreglos bidimensionales en el leguanje de programación Java.

Introducción al tema

Los arreglos bidimensionales o matrices se utilizan con frecuencia para representar tablas de valores, las cuales consisten en información ordenada en filas y columnas. Para identificar un elemento

específico de una tabla debemos especificar dos subíndices. El primero especifica la fila del elemento y el segundo la columna.

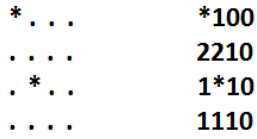
Un arreglo bidimensional con el mismo número de columnas en cada fila puede crearse mediante una expresión de creación de arreglos. Por ejemplo, en la siguiente línea se declara el arreglo **b** de tres renglones por cuatro columnas.
int[][] b = new int[3][4];

La forma en cómo se representan los arreglos bidimensionales los hace bastante flexibles. De hecho, la cantidad de columnas en cada renglón dentro de una matriz no tienen que ser todas iguales. Por ejemplo,
int[][] b = {{1,2},{3,4,5}};
Crea la matriz cuyo renglón 1 contiene las columnas con los valores 1 y 2; y cuyo renglón 2 contiene al 3, 4 y 5.

La manera de recorrer una matriz es mediante dos instrucciones *for* anidadas. El *for* más externo recorre los renglones mientras que el *for* más interno recorre las columnas.

Planteamiento del problema

¿Quien no ha jugado al *Buscaminas*? El objetivo del juego es encontrar todas las minas ubicadas en un campo de dimensiones m renglones por n columnas. El juego muestra un número en un recuadro que indica la cantidad de minas adyacentes a ese recuadro. Cada recuadro tiene, como mucho, ocho recuadros adyacentes. Considera las siguientes matrices:



La matriz de tamaño 4x4 de la izquierda, contiene dos minas, cada una de ellas representada por el caracter "*". Si representamos la misma matriz con los números descritos anteriormente, tendremos la matriz de la derecha.

Se desea escribir un programa que genere un campo de dimensiones *mxn* que contendrá minas de manera aleatoria. Para hacer el programa más dináico, m y n serán valores generados al azar entre 4 y 8, así mismo, cada casilla dentro de la matriz tendrá una probabilidad de 1/7 de tener una mina. Las minas estarán representadas con * y los espacios sin mina se representarán con un punto.

Solución

1. Entender bien el problema

   Queremos simular uno de los muchos comportamientos del juego Buscaminas, el cual consiste en calcular el número de minas adyacentes a cada recuadro; pasando de la representación de una matriz con minas y sin minas a una representación de un campo que incluye la cantidad de minas adyacentes a cada recuadro.

2. Diseñar el algoritmo que resuelva el problema

   La entrada será una matriz de m renglones por n columnas, donde m y n serán generados de manera aleatoria en el rango de entre 4 y 8.

   La salida será una nueva matriz que además de contener las minas, contendrá valores numéricos para indicar la cantidad de minas adyacentes a cada recuadro.

Lo que necesitamos hacer es recorrer la matriz de entrada a la par que vamos a ir modificando y mapeando las minas a la matriz de salida. Las minas se transladarán por el entero 9 debido a que una casilla tiene como máximo 8 minas, entonces el 9 será nuestro número especial. En el momento que nos encontremos con una mina todas las casillas vecinas de la matriz de enteros aumentarán una unidad, debemos tener cuidado con las casillas de las orillas, ya que tienen menos vecinos y la modificación de los índices i y j podría llevarnos a posiciones fuera del rango de la matriz.

Entonces, el algoritmo de solución sería el siguiente:

Generar m y n de manera aleatoria
Crear una matriz de tipo char de dimensiones m x n cuyo nombre sea entrada
Recorrer la matriz entrada y por cada casilla generar un valor entre 0 y 6
    Si el valor aleatorio es 0, asignamos a esa entrada el valor ∗
    En otro caso, asignamos a esa entrada, el valor .
Imprimimos la matriz entrada
Crear una matriz de nombre salida cuyas dimensiones sean m y n
Recorrer la matriz de entrada
    Si el valor leído en entrada es ∗
        Asignar a la matriz salida en la posición i y j el valor 9
            Si i-1>=0 y j-1>=0
                Incrementamos en uno, la casilla superior izquierda de la matriz salida
            Si i-1>=0
                Incrementamos en uno, la casilla superior de la matriz salida
            Si i-1>=0 y j+1<n
                Incrementamos en uno, la casilla superior derecha de la matriz salida
            Si j-1>=0
                Incrementamos en uno, la casilla lateral izquierda de la matriz salida
            Si j+1<n
                Incrementamos en uno, la casilla lateral derecha de la matriz salida
            Si i+1<m y j-1>=0
                Incrementamos en uno, la casilla inferior izquierda de la matriz salida
            Si i+1<m
                Incrementamos en uno, la casilla inferior de la matriz salida
            Si i+1<m y j+1<n
                Incrementamos en uno, la casilla inferior derecha de la matriz salida
Imprimos la matriz salida, cuando encontremos un valor mayor o igual a 9 en lugar de imprimir el valor contenido en la matriz imprimimos ∗

3. Implementar de ser posible, el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice G.

4. Realizar pruebas al código implementado

Podemos darnos cuenta que para cada casilla *i,j* dentro de la matriz se tienen las siguientes casillas vecinas:

i-1,j-1	i-1,j	i-1,j+1
i,j-1	i,j	i,j+1
i+1,j-1	i+1,j	i+1,j+1

Excepto en las orillas de nuestra matriz donde debemos tener cuidado de no acceder a posiciones inválidas, pero ese problema se resuelve en cada *if* de nuestro algoritmo que antes de incrementar las casillas vecinas a una mina verifica que sean accesibles. Por lo tanto nuestro programa es correcto.

Actividades sugeridas

- Paseo del caballo

  ¿Puede la pieza de ajedrez conocida como caballo moverse alrededor de un tablero de ajedrez vacío y tocar cada una de las 64 posiciones una y sólo una vez?

  El caballo realiza solamente movimientos en forma de **L** (dos espacios en una dirección y un espacio en una dirección perpendicular). Utiliza la generación de números aleatorios para permitir que el caballo se desplace a lo largo del tablero (mediante sus movimientos en forma de **L**). Tu programa debe ejecutar un paseo e imprimir el tablero final. ¿Qué tan lejos llegó el caballo?

- Ventas totales

  Una compañía tiene cuatro vendedores que venden cinco productos distintos. Al final del día cada vendedor pasa una nota por cada tipo de producto vendido. Cada nota contiene lo siguiente:

  - El número del vendedor
  - El número del producto
  - El valor total en pesos de ese producto vendido en ese día

  Así cada vendedor pasa entre 0 y 5 notas de venta por día. Suponga que está disponible la información sobre todas las notas del mes pasado. Escribe una aplicación que lea toda esa información del mes pasado y que resuma las ventas totales por vendedor, por producto. Tu programa debe imprimir en formato tabular la información de dichas ventas.

- Gráficos de tortuga

  Imagina a una tortuga mecánica que camina por todo un cuarto bajo el control de una aplicación en Java. La tortuga sostiene una pluma en una de dos posiciones, arriba o abajo. Mientras la pluma está abajo, la tortuga va trazando figuras a medida que se va moviendo, y mientras la pluma está arriba, la tortuga se mueve libremente sin trazar nada.

  Crea un arreglo de 20x20 llamado *piso*. Supón que la tortuga siempre empieza en la posición (0,0) del piso, con la pluma hacia arriba. El conjunto de comandos que la tortuga procesa se muestra a continuación.

Comando	Significado
1	pluma arriba
2	pluma abajo
3	voltear a la derecha
4	voltear a la izquierda
5, x	avanzar al frente x posiciones
6	imprimir el piso
9	término del programa

El siguiente ejemplo dibuja e imprime un cuadro de 12x12 dejando la pluma en posición levantada.

2
5,12
3
5,12
3
5,12
3
5,12
1
6
9

3.8. Proyecto 8

Herencia

Objetivo:

Introducir al alumno al manejo de la herencia en Java como una de las propiedades características de los lenguajes de programación orientados a objetos.

Introducción al tema

La idea de la herencia es permitir la creación de nuevas clases basadas en clases existentes. Cuando heredamos de una clase existente reusamos métodos y variables globales.

La clase ya existente es llamada superclase, clase base o clase padre. La clase nueva es llamada subclase, clase derivada o clase hija.

La superclase directa de una subclase (que se espeficia mediante la palabra *extends* en la primera línea de una declaración de clase) es la superclase a partir de la cual hereda la subclase. Una superclase indirecta de una subclase se encuentra dos o más niveles arriba de esa subclase en la jerarquía de herencia.

Cada objeto de una subclase es también un objeto de la superclase. Sin embargo, el objeto de una superclase no es un objeto de las subclases de su clase.

Los miembros *public* de una superclase son accesibles en cualquier parte en donde el programa tenga una referencia a un objeto de esa superclase, o de una sus subclases. Los miembros *private* de una superclase son accesibles sólo dentro de la declaración de esa superclase. Los miembros *protected* de una superclase tienen un nivel intermedio de protección entre acceso *public* y *private*; pueden ser utilizados por el miembros de la superclase, los miembros de sus subclases y los miembros de otras clases en el mismo paquete.

Cuando un método de una subclase sobrescribe a un método de una superclase, se puede acceder al método de la superclase desde la subclase si se antepone al nombre del método de la subclase la palabra clave **super** y un punto.

Una subclase no puede acceder o heredar los miembros *private* de su superclase; al permitir esto se violaría el encapsulamiento de la superclase. Sin embargo, una subclase puede heredar los miembros no *private* de su superclase.

Una subclase puede invocar de forma explícita a un constructor de su superclase, para ello utiliza la sintaxis de llamada del constructor de la superclase: la palabra: **super**, seguida de los argumentos del constructor de la superclase.

Planteamiento del problema

La empresa de Computación *Compu Technology* necesita llevar un registro de todos sus empleados que se encuentran laborando actualmente, cada empelado tiene los siguientes atributos:

- nombre completo
- cédula profesional
- edad, (nigún empleado tiene menos de 18 ni más de 45 años)
- salario, (ningún empleado gana menos de $3,000 ni más de $15,000)

La empresa cuenta con programadores de computadoras, los cuales tienen las siguientes características adicionales:

- líneas de código por hora
- cantidad de lenguajes de programación que domina

Tu programa debe imprimir los datos de cada programador y además debes implementar y mandar llamar al método *obtenerTipoProgramador()* que imprima el mensaje:
Intermedio: En caso de que el programador domine entre 3 y 5 lenguajes de programación
Avanzado: En caso de que el programador domine más de 5 lenguajes de programación
Ningún programador domina menos de 3 lenguajes de programación.

Tu programa debe imprimir los datos de cada empleado y además debes mandar llamar al método *obtenerTipo()* que regrese la cadena:
Becario si gana entre $3,000 y $4,999
Empleado A si gana entre $5,000 y $9,999
Empleado B si gana entre $10,000 y $14,999
Empleado Base si gana $15,000

Solución

1. Entender bien el problema
   Dada la información sobre los empleados de una empresa de computación debemos mostrar el tipo de empleado y tipo de programador según sueldo y conocimientos.

2. Diseñar el algoritmo que resuelva el problema
   Para modelar un objeto de tipo empleado debemos revisar que la edad esté comprendida entre 18 y 45 así como que el salario esté entre $3,000 y $15,000 estas validaciones irán en el constructor.

Para obtener el tipo de empleado debemos hacer lo siguiente:
Si salario < 5000
Becario
Si salario > 5000 y salario < 10000
Empleado A
Si salario > 10000 y salario < 15000
Empleado B
Si salario = 15000
Empleado Base
Para modelar objetos de la clase Programador debemos revisar que el número de lenguajes de programación dominados sean 3 o más. Si domina más de 5 es Avanzado en caso contrario es Intermedio.

3. Implementar de ser posible, el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice H.

4. Realizar pruebas al código implementado
   Para verificar que nuestro programa es correcto podemos probar a crear más objetos y ejecutar más veces nuestro programa. Debido a que la herencia permite que el estado de algunos objetos cambie no haremos uso de JUnit para realizar pruebas unitarias.

Actividades sugeridas

- Actas
  Escribir un programa para imprimir actas en un registro civil. Las actas pueden ser: acta de nacimiento (fecha, nombre, ciudad), acta de matrimonio (fecha, nombre1, nombre2, ciudad), cartilla de vacunación (fecha, nombre, arreglo con las vacunas).

- Figuras
  Hacer un programa que permita trabajar con figuras geométricas como rectángulos y cuadrados. Se sugiere primero escribir la clase para los rectángulos con largo y ancho. Luego escribir una clase para manejo de los cuadrados como subclase de los rectángulos. Para cada clase escribir métodos para encontrar la altura, el ancho, el perímetro y el área.

3.9. Proyecto 9

Excepciones

Objetivo:
Que el alumno repase objetos, arreglos unidimensionales y además aprenda el uso de excepciones en Java.

Introducción al tema

Una **excepción** es un evento que ocurre en cualquier momento de ejecución de un programa y que modifica el flujo normal de éste. Las excepciones son objetos de la clase **Exception** que almacenan información que se regresa en caso de que ocurra una anormalidad.

Una excepción se activa para indicar que ocurrió una falla durante la ejecución de un método. La excepción se propaga hasta encontrar un método en el cual se indica qué se debe hacer en circunstancias anómalas. Este comportamiento se describe en términos técnicos definidos como estados por los que pasan las excepciones. Estos estados son:

- Disparo
  Al ocurrir un error se activa una excepción creando un objeto de la clase **Exception** con información como: tipo de excepción y estado del programa.

- Atrapado
  Una vez que un método dispara una excepción, el sistema de ejecución busca las instrucciones que especifican qué hacer para esa excepción. Por ejemplo, imprimir un mensaje de error y no hacer nada más, pedir la corrección de manera interactiva, etcétera.

- Terminación
  Si no se encuentra un método que maneje la excepción, termina la ejecución del programa. En caso contrario, se dice que el manejador de excepciones atrapó la excepción y el programa se recupera de la excepción, pues la ejecución de éste continúa de forma normal.

Para disparar una excepción se utiliza la instrucción **throw** con un objeto de la clase *Exception*. La ejecución de esta instrucción significa que ocurrió un error en el método y por lo tanto no puede continuar su ejecución.

Si en un método se puede disparar una excepción y ésta no es de la clase *RuntimeException* ni de sus descendientes entonces al final de la firma del método se debe incluir la cláusula **throws** seguida del nombre de la excepción que se puede disparar.

Para tratar con excepciones se requiere escribir un manejador se excepciones utilizando la cláusula **try**, que tiene la siguiente sintaxis:

```
try {
    instrucciones
}
catch(Excepcion_1 e_1) {
    instrucciones
}
...
catch(Excepcion_n e_n) {
    instrucciones
}
finally {
    instrucciones
}
```

La cláusula *try* contiene código que incluye las instrucciones que pueden disparar la(s) excepción(es).

Las cláusulas *catch* tienen como parámetro un objeto de alguna clase de excepción. En una instrucción *try* puede haber varias cláusulas *catch*; en el cuerpo de cada *try* se coloca el código que implementa la acción a realizar en caso de que ocurra una excepción del tipo de su parámetro.

Por último, la cláusula *finally* contiene el código para establecer un estado adecuado para continuar la ejecución del método donde aparece la instrucción *try*. Incluir la cláusula *finally* es opcional.

La clase *Exception* permite al programador crear sus propias excepciones según sus necesidades.

Planteamiento del problema

Se desea desarrollar un programa que simule el comportamiento de una contestadora telefónica,

la cual almacenará mensajes, un mensaje contiene Remitente y el texto del mensaje en sí. Cada vez que un teléfono suena más de 5 veces, esta contestadora se activa automáticamente y solicita el nombre del remitente y el mensaje a dejar almacenado.
Al ejecutar el programa se debe mostrar un menú con las siguientes opciones:

1. Guarda Mensaje
2. Borra mensaje según remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir

La opción 1 solicitará el remitente y el mensaje a grabar.
La opción 2 permitirá borrar un mensaje de la contestadora solicitando para ello, el nombre del remitente.
La opción 3 borrará todos los mensajes almacenados.
La opción 4 debe mostrar todos los mensajes contenidos en la contestadora.

El límite de almacenamiento de la contestadora es de 10 mensajes. Aunque en la vida real se requiere de un emisor y un receptor ambos en ubicaciones geográficas distintas, para este programa, un mismo usuario podrá guardar y administrar los mensajes a la vez en una sola Terminal. No se requiere que la información sea persistente hasta el momento.

Deberás crear tus propias excepciones para los siguientes casos: Cuando se quiere escuchar un mensaje pero la contestadora está vacía y cuando se desea almacenar un mensaje pero la contestadora ya está llena.

Solución

1. Entender bien el problema

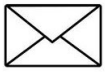
   Lo que necesitamos hacer es simular el comportamiento de una contestadora através de un menú que permita hacer operaciones tales como grabar mensaje, ver todos los mensajes, borrar mensaje y borrar todos los mensajes.

   La entrada serán cadenas de texto y valores numéricos proporcionadas desde la entrada estándar para poder acceder al menú que despliegue las opciones posibles de realizar con la Contestadora. La salida serán mensajes de texto que variarán según la opción solicitada por el usuario. Estos mensajes serán desde los mensajes alojados en la Contestadora como avisos de confirmación a las peticiones del usuario.

2. Diseñar el algoritmo que resuelva el problema

   Debemos pensar cuales son los elementos que intervienen en este programa y podríamos mencionar que algunos de ellos son: Contestadora, Remitente y Mensaje. Estos elementos representan a los objetos que modelarán la solución a nuestro proyecto. Dado que el Remitente no jugará un papel fundamental, no crearemos un objeto de este tipo. Por lo tanto, nos concentraremos en los siguientes objetos y enfocaremos en las acciones que estos pueden realizar:

CONTESTADORA:
Atributos: marca, color, peso
Comportamiento: encender, apagar, grabar mensaje, borrar mensaje, escuchar mensaje



MENSAJE:
Atributos: tamaño, destinatario, remitente, texto
Comportamiento: escribirse, enviarse, eliminarse

Para fines de este ejercicio, sólo consideraremos los atributos y métodos necesarios para resolver nuestro problema.

Ya que la capacidad máxima de la contestadora es de 10, necesitaremos de un arreglo para poder llevar la cuenta y manipular los mensajes, ya sea creándolos, consultándolos o borrándolos. Crearemos la clase *Mensaje* cuyos atributos serán *remitente* y *asunto* (el texto del mensaje).

Tendremos un método *menu* que nos muestre las opciones de lo que podemos hacer con la contestadora.

La opción 1 permitirá guardar un nuevo mensaje, solicitando entonces el nombre del remitente y el texto del mensaje. Una vez que se hayan solicitado estas dos cadenas de texto, se mandará llamar un método de nombre *guardaMensaje*, el cual verifique primero si hay espacio disponible de guardar un nuevo mensaje, en caso de ser así, en la primera posición libre del arreglo se creará un objeto de tipo mensaje con las dos cadenas de tipo String recibidas como argumento.

La opción 2 borra un mensaje de la contestadora, utilizando como clave el nombre del remitente, entonces, solicitaremos el nombre del remitente que se desea borrar desde la entrada estándar y con esta cadena de texto llamaremos al método *borraRemitente* el cual buscará entrada a entrada de la contestadora un mensaje cuyo atributo remitente será el nombre de la persona cuyo mensaje que queremos borrar, cuando lo encuentra, libera ese espacio que ocupaba el mensaje asignándole el valor *null*. Si hay varios mensajes del mismo remitente borra todos los que encuentre. En caso de que no se haya encontrado ninguna coincidencia, se mostrará un mensaje que indique que no se pudo eliminar el mensaje, en caso contrario se mostrará un mensaje avisando que la eliminación ha sido exitosa.

La opción 3 recorre todo el arreglo y asigna todas las entradas con el valor *null*.

La opción 4 recorre todo el arreglo e imprime el contenido en cada posición.

La opción 5, termina con la ejecución del programa.

El menú anterior se ciclará mientras la opción del menú no sea 5 y cuando se proporcione una opción distinta de 1, 2, 3 o 4, se pedirá una opción válida.

Este ejercicio nos pide crear nuestras propias excepciones por lo que tendremos dos clases en donde las definiremos.

3. Implementar de ser posible, el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice I.

4. Realizar pruebas al código implementado

   Para probar que nuestro programa funcione de manera correcta podemos ejecutarlo varias veces con una amplia variedad de casos de prueba, los cuales se pueden proporcionar de manera interactiva desde la entrada estándar.

Actividades sugeridas

- Añade al programa de la contestadora una nueva excepción llamada **NoExisteRemitenteException** la cual se lanzará cada vez queda vez que queramos escuchar los mensajes de un remitente equivocado (que no esté en la lista de remitentes).

- Operaciones Bancarias
  Las operaciones del empleado de un banco son las siguientes:

  - Crear una Cuenta: Crea una nueva cuenta con los siguientes datos:
    - Identificador de la Cuenta: Debe ser un numero o una cadena que identifique a cada cuenta.
    - Nombre del Cliente: Debe ser una cadena que contenga el nombre del cliente.
    - Password de la Cuenta: Debe ser una cadena con el password de la cuenta; la contrase ña debe ser generada de manera aleatoria por el programa con una longitud de 8 caracteres.
    - Monto inicial de la cuenta: Debe ser una flotante que represente el monto inicial de la cuenta.
  - Borrar una Cuenta: Debe permitir borrar una cuenta a partir del identificador de dicha cuenta.
  - Buscar por nombre: Busca las cuentas a partir del nombre del Cliente.
  - Ver todas las Cuentas: Muestra la información de todas las cuentas que han sido creadas.

  Para acceder a estas opciones el programa debe pedir un password de empleado el cual será único y no se podrá modificar.

  Operaciones del Cliente:

  - Consultar Datos: Muestra la información de su cuenta.
  - Retirar: Permite retirar una cantidad de la cuenta (siempre y cuando haya saldo suficiente).
  - Depositar: Permite depositar una cantidad de dinero a la cuenta.
  - Cambiar contraseña: El cliente puede cambiar la contraseña que recibió de manera inicial, esta debe seguir siendo de longitud 8.

  Para acceder a estas opciones el programa debe pedir el ID de la cuenta y el password de la misma.

  Cuando el programa inicie debe mostrar un menú que permita seleccionar las operaciones de empleado o las operaciones de cliente.

  Tu programa debe contener excepciones para los siguientes casos:

  - Intentar borrar una cuenta que no existe
  - Proporcionar información incorrecta al momento de intentar crear una cuenta
  - Tratar de retirar dinero cuando los fondos de la cuenta son insuficientes
  - Cambiar una contraseña proporcionando una longitud distinta de 8

3.10. Proyecto 10

Persistencia

Objetivo:

El objetivo de este proyecto es que una vez que el alumno ya sabe como crear sus propios objetos, aprenda a que estos sean persistentes, para ello se utilizará la serialización de objetos en Java.

Introducción al tema

Los datos que se almacenan en variables y arreglos es temporales; se pierden cuando una variable local queda fuera de alcance o cuando el programa termina. Los programas utilizan archivos para la retención a largo plazo de datos, estos datos que se mantienen en archivos existen más allá de la duración de la ejecución del programa.

Las computadoras almacenan los archivos en dispositivos de almacenamiento secundario como los discos duros.

La clase **File** de Java se utiliza para obtener información acerca de los archivos y directorios. Java cuenta con un mecanismo llamado **serialización de objetos**, el cual permite escribir o leer objetos completos mediante un flujo. Un objeto serializado es un objeto que se representa como una secuencia de bytes e incluye los datos del objeto, así como información acerca del tipo de objeto y los tipos de datos almacenados en el mismo. Una vez que se escribe un objeto serializado en un archivo se puede leer del archivo y deserializarse; es decir, se puede utilizar la información de tipo y los bytes que representan al objeto para recrearlo en la memoria.

Las clases *ObjectInputStream* y *ObjectOutputStream* permiten leer o escribir objetos completos de un flujo. Sólo las clases que implementan la interfaz *Serializable* pueden serializarse y deserializarse con objetos *ObjectOutputStream* y *ObjectInputStream*.

La interfaz *ObjectInput* contiene el método *readObject*, que lee y devuelve una referencia a un objeto Object de un objeto *InputStream*; esta interfaz también contiene un método *writeObject* el cual recibe un objeto Object que implementa a la interfaz *Serializable* como argumento y escribe su información en un objeto *ObjectOutputStream*.

Planteamiento del problema

Hoy en día las agendas electrónicas donde podemos almacenar los datos necesarios(nombre, *e-mail* y teléfono celular) para contactar a una persona, es tan imprescindible que podemos encontrarlas en una gran variedad de dispositivos móviles y computadoras.

Se desea realizar un programa capaz de simular el comportamiento de una agenda, la finalidad de este proyecto será la persistencia de la información, al hablar de persistencia, estamos hablando de programas que una vez terminada su ejecución son capaces de generar en memoria archivos de texto y archivos binarios a partir de los cuales podemos inicializar nuestros objetos al último estado que tenían, cada vez que ejecutemos nuestro programa.

Se desea que nuestra agenda muestre un menú principal que contenga las siguientes operaciones:

- Agregar un contacto
- Consultar toda la agenda
- Buscar un contacto
- Modificar un contacto
- Borrar un contacto

- Borrar todos los contactos
- Salir

La opción número 1 solicitará los siguientes datos:

- Nombre
- Apellido
- Teléfono
- e-mail
- Día de cumpleaños
- Mes de cumpleaños

La opción número 2 mostrará todos los contactos que tengamos registrados en la agenda.
La opción número 3 solicitará sólo el nombre de la persona que buscamos, en caso de que haya nombres repetidos mostrará a todos lo que conincidan con el nombre de búsqueda sin importar el apellido.
La opción número 4 solicita tanto el nombre como el apellido de la persona a la que le queremos realizar alguna modificación en su información, ya que en caso de tener a dos o más personas con el mismo nombre, debemos hacer la distinción del apellido para saber a cual de ellas modificar. Una vez que tenemos los datos de la persona a quien queremos hacer algún cambio, mostramos otro menú donde solicitamos que campo queremos cambiar.
La opción número 5, pide de igual manera el nombre y el apellido de la persona que queremos borrar de nuestra agenda.
La opción número 6 borra todos los registros que tengamos en el agenda registrados.
La opción número 7, termina la ejecución del programa.

Este programa debe ser peristente guardando la información de la agenda en un archivo *agenda.ser*
La capacidad de la agenda es de 100 contactos. Como el objetivo de este programa es la persistencia de objetos y por lo tanto validar que un *e-mail* tenga una estructura correcta, así como la fecha de cumpleaños, no serán tomados en cuenta.

Solución

1. Entender bien el problema

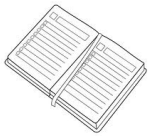
   Se desea programar una agenda capaz de recuperar a los contactos previamente almacenados evitando así tener que registrarlos cada vez que ejecutemos nuestro programa. Este proyecto mantiene una estructura similar al de la Contestadora con la diferencia de que ahora haremos uso de la serialización en el lenguaje Java.

   La entrada de nuestro programa son una serie de opciones de tipo entero para saber que operación debe hacer nuestra agenda, dependiendo dela opción seleccionada se solicitarán valores enteros o bien cadenas para crear objetos con los cuales nuestra agenda funcionará.

   La salida de nuestro programa será un archivo con extensión *.ser* el cual almacenará la información con la cual se cargará nuestra aplicación cada vez que la ejecutamos.

2. Diseñar el algoritmo que resuelva el problema

   Debemos pensar cuales son los objetos que intervienen en nuestro programa, los que resaltan son: Contacto y Agenda; los cuales podemos identificar debido a que un agenda almacena



contactos y deseamos simular el comportamiento de una agenda. Por lo tanto nos concentraremos en estos elementos.
AGENDA:
Atributos: Cantidad de hojas o capacidad de almacenamiento, color, peso
Comportamiento: Agregar contacto, borrar contacto, buscar contacto, ver todos los contactos, editar contacto, borrar toda la agenda.

CONTACTO:
Atributos: Nombre, dirección, teléfono de casa, teléfono de celular, e-mail, fecha de cumpleaños, facebook, ocupación.
Comportamiento: establecer y proporcionar la información de sus atributos.

Para fines de este proyecto consideraremos sólo los atributos y métodos necesarios para resolver nuestro problema. Debido a que la máxima capacidad de contactos que nuestra agenda puede almacenar es de 100 personas, utilizaremos un arreglo unidimensional de longitud 100.
Para realizar este proyecto crearemos 4 clases:
La clase *Contacto.java* contiene los atributos: nombre, apellido, celular, email y cumpleaños.
La clase *Agenda.java* contiene todos los métodos necesarios para trabajar con la agenda.
La clase *PruebaAgenda.java* contiene el método main.
La clase *Fecha.java* es una clase auxiliar para modelar el atributo cumpleaños de la clase *Contacto* de una mejor manera.

Necesitamos en primera instancia de un menú. La opción para guardar un contacto solicitará desde la entrada estándar cada valor con el cual se creará un objeto de tipo Contacto para guardarse en un arreglo que contendrá la información de cada contacto que vayamos registrando.
El contacto se almancerá en la primera posición que se encuentre vacía en la agenda, siempre y cuando aún haya espacio.

Si queremos agregar un contacto a la agenda

Iteramos desde i=0 hasta i=longitud de la agenda -1
    Si el valor de la agenda en la posición i está vacío
        nombre = solicitar nombre
        apellido = solicitar apellido
        telefono = solicitar telefono
        email = solicitar email
        diaCumpleaños = solicitar día de cumpleaños
        mesCumpleaños = solicitar mes de cumpleaños
        Creamos un objeto de tipo Contacto y lo guardamos en la i-ésima posición

Si el valor de la agenda en la posición i no está vacía
incrementamos una variabe contador
Si terminamos de recorrer la agenda y el contador es igual a la logitud de la agenda
La agenda está llena

La opción para consultar todos los contactos, iterará através de todo nuestro arreglo, si lo que haya en la posición actual contiene información, se imprimirá en pantalla.

Iteramos desde i=0 hasta i=longitud de la agenda -1
Si el valor de la agenda en la posición i no está vacío
Imprimimos lo que la agenda tiene almacenado en la posición i
Si el valor de la agenda en la posición i está vacío
Incrementamos una variable contador
Si el valor del contador es igual a la longitud de la agenda
La agenda no contiene contactos

La búsqueda de un contacto itera por todo nuestro arreglo comparando el nombre de cada contacto que tengamos almacenado con el nombre del contacto que deseamos encontrar, este método imprime todos los contactos cuyo nombre coincida con el nombre buscado.

Solicitamos el nombre de la persona a buscar
Iteramos desde i=0 hasta i=longitud de la agenda -1
Si el valor de la agenda en la posición i no está vacío
Tomamos el nombre del contacto almacenado en la posición i
Si el nombre de ese contacto es igual al buscado imprimimos al contacto
Si el valor de la agenda en la posición i está vacío
Incrementamos una variable contador
Si el valor del contador es igual a la longitud de la agenda
El contacto buscado no existe

El método de modificar un contacto, requiere tanto el nombre como el apellido del contacto que queremos editar, luego solicitamos que atributo en particular que deseamos modificar.

Solicitamos el nombre de la persona a editar
Solicitamos el apellido de la persona a editar
Iteramos desde i=0 hasta i=longitud de la agenda -1
Si el valor de la agenda en la posición i no está vacío
Tomamos al contacto almacenado en la posición i
Si el nombre y apellido de ese contacto es igual al buscado
Solicitamos que valor queremos editar
Actualizamos el valor solicitado
Si el valor de la agenda en la posición i está vacío
Incrementamos una variable contador
Si el valor del contador es igual a la longitud de la agenda
El contacto a editar no existe

El método de eliminación de un contacto, funciona de manera similar al de modificación, con la diferencia de que el método de eliminación una vez que localiza al contacto buscado, pone en nulo el valor del arreglo donde este fue localizado.

Solicitamos el nombre de la persona a eliminar

Solicitamos el apellido de la persona a eliminar
Iteramos desde i=0 hasta i=longitud de la agenda -1
Si el valor de la agenda en la posición i no está vacío
Tomamos al contacto almacenado en la posición i
Si el nombre y apellido de ese contacto es igual al buscado
Establecemos el valor de la agenda en la posición i a nulo
Incrementamos una variable contador
Si el valor del contador es igual a la longitud de la agenda
El contacto a eliminar no existe

El método de borrar todo, recorre toda la agenda poniendo en nulo cada valor del arreglo.

Iteramos desde i=0 hasta i=longitud de la agenda -1
Ponemos en nulo el valor de la agenda en la posición i

El método de salida, termina con el ciclo del menú, pero no con la ejecución del programa, ya que antes de salir de la aplicación, se debe invocar un método que almacene toda la información contenida en el agenda para que esta esté disponible en la próxima ejecución. Para lograr recuperar el contenido de la agenda, necesitaremos que nuestro método constructor revise si existe un archivo *.ser* del cual extraiga los datos de nuestra agenda, en caso de no existir, será porque es la primera vez que ejecutamos el programa y entonces este archivo se creará y luego de terminar de manejarlo, escribirá la respectiva información en él.

3. Implementar de ser posible, el algoritmo de solución
   El código fuente de este proyecto se encuentra en el apéndice J.

4. Realizar pruebas al código implementado
   Para verificar que nuestro programa funciona de la manera deseada podemos realizar varias pruebas en tiempo de ejecución con diversos datos.

Actividades sugeridas

- Contestadora persistente
  Modifica el programa mostrado en la sección 5.9 para que los mensajes en la contestadora sean persistentes.

- Repite el ejercicio 2 de la sección 5.9 para que las cuentas bancarias sean persistentes.

Conclusiones

A lo largo de este trabajo se han discutido y analizado diversas aspectos del proceso de enseñanza-aprendizaje de la programación. Es claro que es mucho lo que aún se podría escribir acerca del tema y que en ciertos rubros habría muchas cosas que considerar aparte de las que se han tomado en cuenta aquí. En algunos temas no habrá, ciertamente, un único punto de vista correcto ni una única manera de hacer las cosas. Se ha procurado, sin embargo, proporcionar un análisis objetivo, así como estrategias probadas exitosamente en algunos cursos de programación.

Uno de las conclusiones más inmediatas que es posible formular, luego del análisis hecho, es que impartir un curso de programación poco tiene que ver con la manera en que se imparten otros cursos, aún los de disciplinas científicas o matemáticas. La diferencia estribe en que, salvo excepciones, no existe una única solución correcta a los problemas que se resuelven: no hay un único algoritmo, o una única manera de expresarlo o una única estructura de datos. El conjunto de posibles soluciones puede ser grande y cada una de ellas puede tener ventajas sobre las otras, cada una de ellas valiosa. En buena medida, el éxito de un programador dependerá de sus habilidades para encontrar varias de ellas, analizarlas, discriminarlas y decidir, de acuerdo al contexto, cuál debe programar. Un programador es tanto mejor, cuanto mayor sea su capacidad para formular soluciones alternativas y más fino sea su sentido de orientación para transitar por la selva que ellas constituyen:el matrimonio de la creatividad y el buen juicio.

Como hemos visto en el texto, esta dualidad es lo que constituye todo un reto para el docente ya que debe, simultáneamente: proporcionar las estrategias y herramientas de análisis de problemas, diseñar el conjunto de ejercicios que permitan al estudiante poseer un dominio aceptable de las herramientas de programación y hacer esto de forma que se estimule la creatividad del estudiante.

La labor de programación está basada en un proceso de abstracción, como lo están las disciplinas científicas en general: programar consiste en determinar el modelo más adecuado que captura la esencia del problema a resolver y luego expresar esa abstracción en el lenguaje adecuado. El pensamiento computacional, de acuerdo con lo que se revisó en este trabajo, consiste esencialmente en llevar a cabo este proceso de abstracción hasta ser capaz de instrumentar la solución de la manera más eficiente.

En función de la complejidad del problema, este proceso de abstracción puede transitar por varios niveles. Cuando el problema es suficientemente simple (y no existe un criterio definido para decidirlo a priori) se puede proceder por modelar la secuencia de acciones que deben llevarse a cabo para resolver el problema; se decide pues, el algoritmo que ha de programarse, típicamente desde una descripción inicial muy general, hasta los detalles, se lleva a cabo un diseño top down típico del paradigma de la programación estructurada. Cuando la complejidad del problema crece, el enfoque anterior se hace inmanejable, es entonces mucho mejor que la abstracción modele las entidades involucradas en el problema, los objetos, cuyo comportamiento estará, como antes, determinado por

algoritmos que han de usar de la programación estructurada, pero que permiten descomponer la complejidad del programa gracias a la encapsulación.

No es de ninguna manera trivial para el docente, instruir y guiar al alumno a lo largo de este camino de ida y vuelta desde el dominio concreto del problema hasta la abstracción que lo modela mejor y de regreso al ámbito de la solución concreta. Esperamos haber contribuido de alguna manera arrojando luz sobre algunos obstáculos típicos y sobre algunas estrategias que han probado ser útiles.

En mi experiencia como ayudante de cursos de programación en la Facultad de Ciencias, la mayoría de los estudiantes que no han tenido contacto previo con la programación de computadoras, parecen pensar que podrán escribir programas como si siguieran recetas de cocina, como si programar fuera un proceso mecánico. Paradójicamente, como si programar fuera ejecutar un programa. Claro que tarde o temprano este concepto de lo que es programar los lleva a fallar. Suelen entonces atribuir el problema a una falta de manejo de la herramienta: del lenguaje de programación; se enfocan entonces en la sintaxis y las bibliotecas usadas en lugar de poner más atención al modelo diseñado o al desarrollo del algoritmo que resuelve el problema; tienden a cuestionarse más acerca de cómo expresaron su modelo que acerca de si el modelo elegido es el adecuado. La competencia en la programación no proviene, ciertamente, del buen manejo y el conocimiento del un lenguaje de programación (lo que es importante, claro, pero no es lo más importante). La competencia reside en la capacidad de abstraer lo esencial del problema, modelarlo en términos formales y expresarlo en la sintaxis del lenguaje de programación de forma que los resultados obtenidos se puedan mapear en soluciones al problema real. En todo curso de programación es el diseño de la solución lo que se debe enfatizar.

El presente trabajo espera contribuir a la reflexión que sobre la programación debe llevar a cabo el docente de la computación. Pero, como he señalado, esta reflexión dista mucho de estar completa o de contemplar todos los aspectos relevantes. Habría que continuar analizando y discutiendo, por ejemplo, la mejor estrategia pedagógica para enseñar el pensamiento computacional o decidir cúal es el tipo de pensamiento abstracto que resulta más útil para la programación y cómo es posible enfatizarlo. Quedan allí un par de hilos que habría que jalar, habrá que ver qué madeja se puede formar con ellos.

Apéndice A

Implementación del proyecto 1

Clase *Formulas.java*

```
/**
 * Programa que muestra el uso de variables en Java y la precendencia de
    operadores
 * mediante el calculo de expresiones algebraicas
 * @author: Cinthia Rodriguez Maya
 * @version: Marzo de 2014
 */
public class Formulas {

   public static void main(String[] args) {
      //variables para sustuir en las formulas
      int x=5;
      int y=2;
      int z=4;
      //variable que almacena el resultado de cada operacion
      double resultado;

      //Formula 1
      resultado = ((x+3)/(5*Math.pow(y,2)))-Math.sqrt(y)+10;
      System.out.println("Resultado de la formula 1: " + resultado);

      //Formula 2
      resultado = Math.sqrt(4*x+9*y)+Math.pow(Math.pow(x,5)/Math.pow(y,2)
         ,2);
      System.out.println("Resultado de la formula 2: " + resultado);

      //Formula 3
      resultado = Math.pow((6*x+2*z)/2,5);
      System.out.println("Resultado de la formula 3: " + resultado);

      //Formula 4
      resultado = Math.sqrt(4*x+5*y*z)+9*y-(1/x);
      System.out.println("Resultado de la formula 4: " + resultado);

      //Formula 5
      resultado = ((x+Math.pow(y,2))/z)*(Math.pow(x,3)/Math.pow((y+2),5));
      System.out.println("Resultado de la formula 5: " + resultado);
   }
}
```

Ejecución de nuestro proyecto

```
Resultado de la formula 1: 8.985786437626905
Resultado de la formula 2: 610357.726914003
Resultado de la formula 3: 2476099.0
Resultado de la formula 4: 25.745966692414832
Resultado de la formula 5: 0.274658203125
```

Apéndice B

Implementación del proyecto 2

Clase *Collatz.java*

```
/**
 * Programa para calcular la longitud mas grande de secuencias Collatz
 * @author: Cinthia Rodriguez Maya
 * @version: Marzo de 2014
 */
import java.util.Scanner;

public class Collatz {

   public static void main(String[] args) {
      Scanner entrada = new Scanner(System.in);
      //Leemos los valores del teclado
      System.out.println("Introduce el primer valor");
      int val1=entrada.nextInt();
      System.out.println("Introduce el segundo valor");
      int val2=entrada.nextInt();
      int i=val1;
      int j=val2;
      int temporal=0;
      int maxima=0;
      int contador=0;
      //iteramos a traves del intervalo i a j
      while(i<=j) {
         temporal=i;
         contador=1;
         while(temporal>1) {
            if(temporal%2==0) {
               temporal=temporal/2;
               contador++;
            } else {
               temporal=(temporal*3)+1;
               contador++;
            }
         }
         if(contador>maxima) {
            maxima=contador;
         }
         i++;
      }
```

```
40          System.out.println("Maxima: " + maxima);
41      }
42  }
```

Ejecución de nuestro proyecto

```
Introduce el primer valor
5
Introduce el segundo valor
90
Maxima: 116
```

```
Introduce el primer valor
67
Introduce el segundo valor
1289
Maxima: 182
```

Apéndice C

Implementación del proyecto 3

Clase *ConversorBinario.java*

```
1  /**
2   * Clase ConversorBinario
3   * Programa que realiza conversiones del sistema decimal a binario y de
        binario a decimal asi como
4   * el calculo del complemento a dos de un numero decimal
5   *
6   * @author Cinthia Rodriguez Maya
7   * @version: Marzo de 2014
8   */
9
10 package binario;
11
12 import java.util.Scanner;
13
14 public class ConversorBinario {
15
16    Scanner entrada = new Scanner(System.in);
17
18    /**
19     * Metodo menu
20     * muestra las posibles acciones a llevar a cabo, termina con la
           ejecucion
21     * del programa al poner la opcion 3 -> Salir
22     */
23    public void menu() {
24       int opcion=0;
25       do {
26          String numero="";
27          String conversion="";
28          System.out.println("Selecciona una opcion");
29          System.out.println("1. Convertir un numero en base 10 a base 2");
30          System.out.println("2. Convertir un numero en base 2 a base 10");
31          System.out.println("3. Salir");
32          opcion=entrada.nextInt();
33          switch(opcion) {
34             case 1:
35                System.out.println("Introduce el numero en base 10 que
                      quieres convertir a binario");
36                numero=entrada.next();
```

```
37                  conversion=decimal_binario(numero);
38                  System.out.println("La representacion binaria de " + numero
                        + " es " + conversion);
39                  break;
40              case 2:
41                  System.out.println("Introduce el numero en base 2 que
                       quieres convertir a decimal");
42                  numero=entrada.next();
43                  conversion=binario_decimal(numero);
44                  System.out.println("La representacion decimal de " + numero
                        + " es " + conversion);
45                  break;
46              case 3:
47                  System.exit(1);
48                  break;
49              default:
50                  System.out.println("Introduce una opcion valida");
51                  break;
52          }
53      }while(opcion!=3);
54  }
55
56  /**
57   * Metodo decimal_binario
58   * realiza la conversion de un numero decimal a su representacion
         binaria
59   * @param cadena, el numero decimal a convertir
60   */
61  public String decimal_binario(String cadena) {
62      String respuesta="";
63      String reversa="";
64      int numero=Integer.parseInt(cadena);
65      int cociente=0;
66      int residuo=0;
67      if(numero==0) {
68          reversa="0";
69      }
70      while(numero>=1) {
71          residuo=numero%2;
72          respuesta=respuesta+residuo;
73          numero=numero/2;
74      }
75      //Calculamos la reversa
76      for(int i=respuesta.length()-1; i>=0; i--) {
77          char c = respuesta.charAt(i);
78          String temp=Character.toString(c);
79          reversa=reversa+temp;
80      }
81      return reversa;
82  }
83
84  /**
85   * Metodo binario_decimal
86   * realiza la conversion de un numero binario a decimal
87   * @param cadena, el numero binario a convertir
88   */
```

```
89      public String binario_decimal(String cadena) {
90          int n=0;
91          int resultado=0;
92          for(int i=cadena.length()-1; i>=0; i--) {
93              char c = cadena.charAt(i);
94              String var = Character.toString(c);
95              int numero=Integer.parseInt(var);
96              int mult = numero * (int)Math.pow(2,n);
97              resultado=resultado+mult;
98              n++;
99          }
100         String respuesta=Integer.toString(resultado);
101         return respuesta;
102     }
103
104 }
```

Clase *UsoConversorBinario.java*

```
1  /**
2   * Clase UsoConversorBinario
3   * Contiene el m todo main para la clase ConversorBinario
4   *
5   * @author Cinthia Rodriguez Maya
6   * @version: Marzo de 2014
7   */
8
9  package binario;
10
11 public class UsoConversorBinario {
12
13     public static void main(String[] args) {
14
15         ConversorBinario cb = new ConversorBinario();
16         cb.menu();
17     }
18 }
```

Ejecución de nuestro proyecto

```
Selecciona una opcion
1. Convertir un numero en base 10 a base 2
2. Convertir un numero en base 2 a base 10
3. Salir
1
Introduce el numero en base 10 que quieres convertir a binario
34
La representacion binaria de 34 es 100010
Selecciona una opcion
1. Convertir un numero en base 10 a base 2
2. Convertir un numero en base 2 a base 10
3. Salir
2
Introduce el numero en base 2 que quieres convertir a decimal
1110
La representacion decimal de 1110 es 14
Selecciona una opcion
1. Convertir un numero en base 10 a base 2
2. Convertir un numero en base 2 a base 10
3. Salir
```

Clase *TestBinario.java* para las pruebas unitarias

```
/**
 * Clase para pruebas unitarias de la clase {@link ConversorBinario.java}.
 * @author: Cinthia Rodriguez Maya
 * @version: Marzo de 2014
 */

package binario.test;

import binario.*;
import org.junit.Assert;
import org.junit.Test;
import java.util.Random;

public class TestBinario {

   Random numerosAleatorios = new Random();
   ConversorBinario cb = new ConversorBinario();

   /**
    * Prueba unitaria para {@link decimal_binario#ConversorBinario}.
    */
   @Test public void testDecimalBinario() {
      int i=1;
      while(i<=100) {
         int valor = numerosAleatorios.nextInt(1000)+1;
         String binario_api = Integer.toBinaryString(valor);
         String decimal = Integer.toString(valor);
         String respuestaMetodo = cb.decimal_binario(decimal);
         Assert.assertTrue(respuestaMetodo.equals(binario_api));
         i++;
      }
   }

   /**
    * Prueba unitaria para {@link binario_decimal#ConversorBinario}.
    */
   @Test public void testBinarioDecimal() {
      int j=1;
      while(j<=100) {
         int valor = numerosAleatorios.nextInt(1000)+1;
         String binario_api = Integer.toBinaryString(valor);
         String respuestaMetodo = cb.binario_decimal(binario_api);
         int decimal=Integer.parseInt(respuestaMetodo);
         Assert.assertTrue(decimal==valor);
         j++;
      }
   }

}
```

Ejecución de las pruebas unitarias:

```
compile.ConversorB:
    [javac] Compiling 3 source files to /home/cindy/Escritorio/ConversorB/build

ConversorB.jar:
      [jar] Building jar: /home/cindy/Escritorio/ConversorB/ConversorB.jar

test:
    [junit] Running binario.test.TestBinario
    [junit] Testsuite: binario.test.TestBinario
    [junit] Tests run: 2, Failures: 0, Errors: 0, Time elapsed: 0.08 sec
    [junit] Tests run: 2, Failures: 0, Errors: 0, Time elapsed: 0.08 sec
    [junit]
    [junit] Testcase: testDecimalBinario took 0.021 sec
    [junit] Testcase: testBinarioDecimal took 0.012 sec

BUILD SUCCESSFUL
Total time: 4 seconds
```

Apéndice D

Implementación del proyecto 4

Clase *Punto.java*

```
/**
 * Clase Punto
 * Un punto se crea con dos coordenadas 'x' y 'y'
 * @author Amparo L pez Gaona
 * @version: Marzo de 2014
 */
package recta;

public class Punto {

   private double x;
   private double y;

   /**
    * Metodo constructor
    * @param x, coordenada en el eje x
    * @param y, coordenada en el eje y
    */
   public Punto(double x, double y) {
      this.x = x;
      this.y = y;
   }

   public double getX() {
      return x;
   }

   public double getY() {
      return y;
   }

   /**
    * Metodo toString
    * @return una representacion en cadena de un objeto Punto
    */
   public String toString() {
      return "(" + x +"," + y + ")";
   }
}
```

Clase *Recta.java*

```java
/**
 * Clase que representa un recta de forma analitica, es decir de la forma
 * y = mx + b
 *
 * @author Cinthia Rodriguez Maya
 * @version: Marzo de 2014
 */

package recta;

public class Recta {

   private double m;   // Pendiente.
   private double b;   // Desplazamiento.

   /**
    * Metodo constructor
    */
   public Recta(double m, double b) {
      this.m = m;
      this.b = b;
   }

   public double obtenerPendiente() {
      return m;
   }

   public double obtenerOrdenada() {
      return b;
   }

   /**
    * Metodo que determina si el punto P forma parte de una recta
    * @param p, el Punto
    * @return true si P forma parte de la recta
    */
   public boolean esta(Punto p) {
      return p.getY() == (this.getM() * p.getX()) + this.getB();
   }

   /**
    * Metodo que determina si dos rectas son iguales
    * @param r, la recta a comparar con la que hizo la llamada
    * @return true si ambas rectas son iguales
    */
   public boolean esIgual(Recta r) {
      return((this.getM() == r.getM()) && (this.getB() == r.getB()));
   }

   /**
    * Metodo que determina si dos rectas son paralelas
    * @param r, la recta a comparar con la que hizo la llamada
    * @return true si las rectas son paralelas
    */
```

```
55      public boolean esParalela(Recta r) {
56         return (this.getM() == r.getM() && r!=this);
57      }
58
59      /**
60       * Metodo que determina si dos rectas son ortogonales
61       * @param r, la recta a comparar con la que hizo la llamada
62       * @return true si las rectas son ortogonales
63       */
64      public boolean esOrtogonal(Recta r) {
65         return ((this.getM() * r.getM()) == -1);
66      }
67
68      /**
69       * Metodo que calcula el punto de interseccion de dos rectas que se
             cruzan entre si
70       * @param r1, una de las rectas
71       * @param r2, la otra recta
72       * @return pi, el punto de interseccion
73       * se debe verificar que las rectas no sean paralelas para que pueda
             haber cruce
74       */
75      public static Punto intersecta(Recta r1, Recta r2) {
76         Punto pi;
77         double xi, yi;
78         double aux = (r1.getM() + (r2.getM() * -1));
79         double aux2 = (r2.getB() + (r1.getB() * -1));
80         xi = (aux2 / aux);
81         yi = (((r1.getM() * xi) + r1.getB()));
82         pi = new Punto(xi, yi);
83         return pi;
84      }
85
86      /**
87       * Metodo que calcula la ecuacion de una recta que pasa por un punto y
             es
88       * ortogonal a la recta que invoca al metodo
89       * @param p, el Punto por el que pasa la recta
90       * @return orto, la ecuacion de la recta
91       */
92      public Recta ortogonal(Punto p) {
93         Recta orto;
94         double m2 = ((1/(this.getM())) * -1);
95         double b2 = ((this.getM()* p.getX()) + (p.getY()));
96         orto = new Recta(m2, b2);
97         return orto;
98      }
99
100     /**
101      * Metodo toString
102      * @return una representacion en cadena de un objeto Recta
103      */
104     public String toString() {
105        String cad="";
106        if(b==0) {
107           cad = "y = " + m + "x";
```

```
108         } else if (b<0) {
109            cad = "y = " + m + "x" + b;
110         } else {
111            cad = "y = " + m + "x+" + b;
112         }
113         return cad;
114      }
115   }
```

Clase *UsoRecta.java*

```
1   /**
2    * Clase de uso de Recta
3    *
4    * @author Cinthia Rodriguez Maya
5    * @version: Marzo de 2014
6    */
7
8   package recta;
9
10  public class UsoRecta {
11
12     public static void main(String[] args) {
13
14        Recta r1 = new Recta(1,-2);
15        Recta r2 = new Recta(-1,1);
16        Recta r3 = new Recta(-1,1);
17        Recta r4 = new Recta(1,-8);
18        Punto p1 = new Punto(-1,5);
19
20        System.out.print("Recta 1: ");
21        System.out.println(r1);
22        System.out.print("Recta 2: ");
23        System.out.println(r2);
24        System.out.print("Recta 3: ");
25        System.out.println(r3);
26        System.out.print("Recta 4: ");
27        System.out.println(r4);
28        System.out.print("Punto 1: ");
29        System.out.println(p1);
30
31        System.out.println("Esta el punto p1 en r1");
32        System.out.println(r1.esta(p1));
33        System.out.println("Son iguales r1 y r2");
34        System.out.println(r1.esIgual(r2));
35        System.out.println("Son iguales r2 y r3");
36        System.out.println(r2.esIgual(r3));
37        System.out.println("Son paralelas r1 y r4");
38        System.out.println(r2.esParalela(r3));
39        System.out.println("Son ortogonales r1 y r4");
40        System.out.println(r2.esOrtogonal(r3));
41        System.out.println("Punto de interseccion de r1 con r2");
42        System.out.println(Recta.intersecta(r1,r2));
43        System.out.println("Ecuacion de la recta que para por p1 y es
             ortogonal a r4");
44        System.out.println(r4.ortogonal(p1));
```

```
45      }
46  }
```

Ejecución de nuestro proyecto:

```
Recta 1: y = 1.0x-2.0
Recta 2: y = -1.0x+1.0
Recta 3: y = -1.0x+1.0
Recta 4: y = 1.0x-8.0
Punto 1: (-1.0,5.0)
Esta el punto p1 en r1
false
Son iguales r1 y r2
false
Son iguales r2 y r3
true
Son paralelas r1 y r4
true
Son ortogonales r1 y r4
false
Punto de interseccion de r1 con r2
(1.5,-0.5)
Ecuacion de la recta que para por p1 y es ortogonal a r4
y = -1.0x+4.0
```

Clase *TestRecta.java* para las pruebas unitarias

```
1   package recta.test;
2
3   import recta.*;
4   import org.junit.Assert;
5   import org.junit.Test;
6
7   /**
8    * Clase para pruebas unitarias de la clase {@link Recta.java}.
9    * @author: Cinthia Rodriguez Maya
10   * @version: Marzo de 2014
11   */
12
13  public class TestRecta{
14
15     //Rectas
16     Recta r1 = new Recta(1.0,3.0);
17     Recta r2 = new Recta(6.0,-13.0);
18     Recta r3= new Recta(-0.25,-3.0);
19     Recta r4 = new Recta(5.0,-2.0);
20     Recta r5 = new Recta(4.0,1.0);
21     //Puntos
22     Punto p1 = new Punto(0.0,0.0);
23     Punto p2 = new Punto(3.0,5.0);
24     Punto p3 = new Punto(2.0,-1.0);
25
26     /**
27      * Prueba unitaria para {@link Recta#Recta}.
28      */
29     @Test public void testRecta() {
30        Assert.assertTrue(r1.obtenerPendiente() == 1.0);
31        Assert.assertTrue(r1.obtenerOrdenada() == 3.0);
32        Assert.assertTrue(r2.obtenerPendiente() == 6.0);
33        Assert.assertTrue(r2.obtenerOrdenada() == -13.0);
```

```
34          Assert.assertTrue(r3.obtenerPendiente() == -0.25);
35          Assert.assertTrue(r3.obtenerOrdenada() == -3.0);
36          Assert.assertTrue(r4.obtenerPendiente() == 5.0);
37          Assert.assertTrue(r4.obtenerOrdenada() == -2.0);
38          Assert.assertTrue(r5.obtenerPendiente() == 4.0);
39          Assert.assertTrue(r5.obtenerOrdenada() == 1.0);
40      }
41
42      /**
43       * Prueba unitaria para {@link Recta#getM}.
44       */
45      @Test public void testGetM() {
46          Assert.assertTrue(r1.obtenerPendiente() == 1.0);
47          Assert.assertTrue(r2.obtenerPendiente() == 6.0);
48          Assert.assertTrue(r3.obtenerPendiente() == -0.25);
49          Assert.assertTrue(r4.obtenerPendiente() == 5.0);
50          Assert.assertTrue(r5.obtenerPendiente() == 4.0);
51      }
52
53      /**
54       * Prueba unitaria para {@link Recta#getB}.
55       */
56      @Test public void testGetB() {
57          Assert.assertTrue(r1.obtenerOrdenada() == 3.0);
58          Assert.assertTrue(r2.obtenerOrdenada() == -13.0);
59          Assert.assertTrue(r3.obtenerOrdenada() == -3.0);
60          Assert.assertTrue(r4.obtenerOrdenada() == -2.0);
61          Assert.assertTrue(r5.obtenerOrdenada() == 1.0);
62      }
63
64      /**
65       * Prueba unitaria para {@link Punto#Punto}.
66       */
67      @Test public void testPunto() {
68          Assert.assertTrue(p1.getX() == 0.0);
69          Assert.assertTrue(p1.getY() == 0.0);
70          Assert.assertTrue(p2.getX() == 3.0);
71          Assert.assertTrue(p2.getY() == 5.0);
72          Assert.assertTrue(p3.getX() == 2.0);
73          Assert.assertTrue(p3.getY() == -1.0);
74      }
75
76      /**
77       * Prueba unitaria para {@link Punto#getX}.
78       */
79      @Test public void testGetX() {
80          Assert.assertTrue(p1.getX() == 0.0);
81          Assert.assertTrue(p2.getX() == 3.0);
82          Assert.assertTrue(p3.getX() == 2.0);
83      }
84
85      /**
86       * Prueba unitaria para {@link Punto#getY}.
87       */
88      @Test public void testGetY() {
89          Assert.assertTrue(p1.getY() == 0.0);
```

```
90          Assert.assertTrue(p2.getY() == 5.0);
91          Assert.assertTrue(p3.getY() == -1.0);
92      }
93
94      /**
95       * Prueba unitaria para {@link Punto#toString}.
96       */
97      @Test public void testToStringPunto() {
98          String pto1="(0.0,0.0)";
99          String pto2="(3.0,5.0)";
100         String pto3="(2.0,-1.0)";
101
102         Assert.assertTrue(pto1.equals(p1.toString()));
103         Assert.assertTrue(pto2.equals(p2.toString()));
104         Assert.assertTrue(pto3.equals(p3.toString()));
105     }
106
107     /**
108      * Prueba unitaria para {@link Recta#toString}.
109      */
110     @Test public void testToStringRecta() {
111         String rec1="y = 1.0x+3.0";
112         String rec2="y = 6.0x-13.0";
113         String rec3="y = -0.25x-3.0";
114         String rec4="y = 5.0x-2.0";
115         String rec5="y = 4.0x+1.0";
116
117         Assert.assertTrue(rec1.equals(r1.toString()));
118         Assert.assertTrue(rec2.equals(r2.toString()));
119         Assert.assertTrue(rec3.equals(r3.toString()));
120         Assert.assertTrue(rec4.equals(r4.toString()));
121         Assert.assertTrue(rec5.equals(r5.toString()));
122     }
123
124     /**
125      * Prueba unitaria para {@link Recta#esta}.
126      */
127     @Test public void testEsta() {
128         Assert.assertTrue(r1.esta(p1)==false);
129         Assert.assertTrue(r2.esta(p1)==false);
130         Assert.assertTrue(r3.esta(p1)==false);
131         Assert.assertTrue(r4.esta(p1)==false);
132         Assert.assertTrue(r5.esta(p1)==false);
133         Assert.assertTrue(r1.esta(p2)==false);
134         Assert.assertTrue(r2.esta(p2)==true);
135         Assert.assertTrue(r3.esta(p2)==false);
136         Assert.assertTrue(r4.esta(p2)==false);
137         Assert.assertTrue(r5.esta(p2)==false);
138         Assert.assertTrue(r1.esta(p3)==false);
139         Assert.assertTrue(r2.esta(p3)==true);
140         Assert.assertTrue(r3.esta(p3)==false);
141         Assert.assertTrue(r4.esta(p3)==false);
142         Assert.assertTrue(r5.esta(p3)==false);
143     }
144
145     /**
```

```
146         * Prueba unitaria para {@link Recta#esIgual}.
147         */
148        @Test public void testEsIgual() {
149           Assert.assertTrue(r1.esIgual(r1)==true);
150           Assert.assertTrue(r1.esIgual(r2)==false);
151           Assert.assertTrue(r1.esIgual(r3)==false);
152           Assert.assertTrue(r1.esIgual(r4)==false);
153           Assert.assertTrue(r1.esIgual(r5)==false);
154           Assert.assertTrue(r2.esIgual(r2)==true);
155           Assert.assertTrue(r2.esIgual(r3)==false);
156           Assert.assertTrue(r2.esIgual(r4)==false);
157           Assert.assertTrue(r2.esIgual(r5)==false);
158           Assert.assertTrue(r3.esIgual(r3)==true);
159           Assert.assertTrue(r3.esIgual(r4)==false);
160           Assert.assertTrue(r3.esIgual(r5)==false);
161           Assert.assertTrue(r4.esIgual(r4)==true);
162           Assert.assertTrue(r4.esIgual(r5)==false);
163           Assert.assertTrue(r5.esIgual(r5)==true);
164        }
165
166        /**
167         * Prueba unitaria para {@link Recta#esParalela}.
168         */
169        @Test public void testEsParalela() {
170           Assert.assertTrue(r1.esParalela(r2)==false);
171           Assert.assertTrue(r1.esParalela(r3)==false);
172           Assert.assertTrue(r1.esParalela(r4)==false);
173           Assert.assertTrue(r1.esParalela(r5)==false);
174           Assert.assertTrue(r2.esParalela(r3)==false);
175           Assert.assertTrue(r2.esParalela(r4)==false);
176           Assert.assertTrue(r2.esParalela(r5)==false);
177           Assert.assertTrue(r3.esParalela(r4)==false);
178           Assert.assertTrue(r3.esParalela(r5)==false);
179           Assert.assertTrue(r4.esParalela(r5)==false);
180        }
181
182        /**
183         * Prueba unitaria para {@link Recta#esOrtogonal}.
184         */
185        @Test public void testEsOrtogonal() {
186           Assert.assertTrue(r1.esOrtogonal(r2)==false);
187           Assert.assertTrue(r1.esOrtogonal(r3)==false);
188           Assert.assertTrue(r1.esOrtogonal(r4)==false);
189           Assert.assertTrue(r1.esOrtogonal(r5)==false);
190           Assert.assertTrue(r2.esOrtogonal(r3)==false);
191           Assert.assertTrue(r2.esOrtogonal(r4)==false);
192           Assert.assertTrue(r2.esOrtogonal(r5)==false);
193           Assert.assertTrue(r3.esOrtogonal(r4)==false);
194           Assert.assertTrue(r3.esOrtogonal(r5)==true);
195           Assert.assertTrue(r4.esOrtogonal(r5)==false);
196        }
197
198        /**
199         * Prueba unitaria para {@link Recta#intersecta}.
200         */
201        @Test public void testIntersecta() {
```

```
202         Punto r1_r2 = Recta.intersecta(r1,r2);
203         Punto r1_r3 = Recta.intersecta(r1,r3);
204         Punto r1_r4 = Recta.intersecta(r1,r4);
205         Punto r2_r4 = Recta.intersecta(r2,r4);
206         Punto r2_r5 = Recta.intersecta(r2,r5);
207         Punto r3_r1 = Recta.intersecta(r3,r1);
208         Punto r3_r2 = Recta.intersecta(r3,r2);
209         Punto r4_r1 = Recta.intersecta(r4,r1);
210         Punto r4_r2 = Recta.intersecta(r4,r2);
211         Punto r4_r5 = Recta.intersecta(r4,r5);
212         Punto r5_r2 = Recta.intersecta(r5,r2);
213         Punto r5_r4 = Recta.intersecta(r5,r4);
214 
215         Assert.assertTrue(r1_r2.getX()==3.2);
216         Assert.assertTrue(r1_r2.getY()==6.2);
217         Assert.assertTrue(r1_r3.getX()==-6/1.25);
218         Assert.assertTrue(r1_r3.getY()==(-6/1.25)+3);
219         Assert.assertTrue(r1_r4.getX()==1.25);
220         Assert.assertTrue(r1_r4.getY()==4.25);
221         Assert.assertTrue(r2_r4.getX()==11.0);
222         Assert.assertTrue(r2_r4.getY()==53.0);
223         Assert.assertTrue(r2_r5.getX()==7.0);
224         Assert.assertTrue(r2_r5.getY()==29.0);
225         Assert.assertTrue(r3_r1.getX()==-4.8);
226         Assert.assertTrue(r3_r1.getY()==-1.8);
227         Assert.assertTrue(r3_r2.getX()==1.6);
228         Assert.assertTrue(r3_r2.getY()==-3.4);
229         Assert.assertTrue(r4_r1.getX()==1.25);
230         Assert.assertTrue(r4_r1.getY()==4.25);
231         Assert.assertTrue(r4_r2.getX()==11.0);
232         Assert.assertTrue(r4_r2.getY()==53.0);
233         Assert.assertTrue(r4_r5.getX()==3.0);
234         Assert.assertTrue(r4_r5.getY()==13.0);
235         Assert.assertTrue(r5_r2.getX()==7.0);
236         Assert.assertTrue(r5_r2.getY()==29.0);
237         Assert.assertTrue(r5_r4.getX()==3.0);
238         Assert.assertTrue(r5_r4.getY()==13.0);
239     }
240 
241     /**
242      * Prueba unitaria para {@link Recta#ortogonal}.
243      */
244     @Test public void testOrtogonal() {
245         Recta or1 = r1.ortogonal(p1);
246         Recta or2 = r1.ortogonal(p2);
247         Recta or3 = r1.ortogonal(p3);
248         Recta or4 = r3.ortogonal(p1);
249         Recta or5 = r3.ortogonal(p2);
250         Recta or6 = r3.ortogonal(p3);
251         Recta or7 = r4.ortogonal(p1);
252         Recta or8 = r4.ortogonal(p2);
253         Recta or9 = r4.ortogonal(p3);
254         Recta or10 = r5.ortogonal(p1);
255         Recta or11 = r5.ortogonal(p2);
256         Recta or12 = r5.ortogonal(p3);
257 
```

```
258         Assert.assertTrue(or1.getM()==-1.0);
259         Assert.assertTrue(or1.getB()==0.0);
260         Assert.assertTrue(or2.getM()==-1.0);
261         Assert.assertTrue(or2.getB()==8.0);
262         Assert.assertTrue(or3.getM()==-1.0);
263         Assert.assertTrue(or3.getB()==1.0);
264         Assert.assertTrue(or4.getM()==4.0);
265         Assert.assertTrue(or4.getB()==0.0);
266         Assert.assertTrue(or5.getM()==4.0);
267         Assert.assertTrue(or5.getB()==4.25);
268         Assert.assertTrue(or6.getM()==4.0);
269         Assert.assertTrue(or6.getB()==-1.5);
270         Assert.assertTrue(or7.getM()==-0.2);
271         Assert.assertTrue(or7.getB()==0.0);
272         Assert.assertTrue(or8.getM()==-0.2);
273         Assert.assertTrue(or8.getB()==20);
274         Assert.assertTrue(or9.getM()==-0.2);
275         Assert.assertTrue(or9.getB()==9);
276         Assert.assertTrue(or10.getM()==-0.25);
277         Assert.assertTrue(or10.getB()==0.0);
278         Assert.assertTrue(or11.getM()==-0.25);
279         Assert.assertTrue(or11.getB()==17);
280         Assert.assertTrue(or12.getM()==-0.25);
281         Assert.assertTrue(or12.getB()==7);
282     }
283 }
```

Ejecución de las pruebas unitarias:

```
compile.proyecto3:
    [javac] Compiling 3 source files to /home/cindy/Escritorio/proyecto3/build

proyecto3.jar:
      [jar] Building jar: /home/cindy/Escritorio/proyecto3/practica3.jar

test:
    [junit] Running recta.test.TestRecta
    [junit] Testsuite: recta.test.TestRecta
    [junit] Tests run: 14, Failures: 0, Errors: 0, Time elapsed: 0.117 sec
    [junit] Tests run: 14, Failures: 0, Errors: 0, Time elapsed: 0.117 sec
    [junit]
    [junit] Testcase: testRecta took 0.01 sec
    [junit] Testcase: testObtenerPendiente took 0.001 sec
    [junit] Testcase: testObtenerOrdenada took 0 sec
    [junit] Testcase: testPunto took 0.001 sec
    [junit] Testcase: testGetX took 0.001 sec
    [junit] Testcase: testGetY took 0.001 sec
    [junit] Testcase: testToStringPunto took 0.001 sec
    [junit] Testcase: testToStringRecta took 0.001 sec
    [junit] Testcase: testEsta took 0 sec
    [junit] Testcase: testEsIgual took 0 sec
    [junit] Testcase: testEsParalela took 0.001 sec
    [junit] Testcase: testEsOrtogonal took 0 sec
    [junit] Testcase: testIntersecta took 0 sec
    [junit] Testcase: testOrtogonal took 0.001 sec

BUILD SUCCESSFUL
Total time: 4 seconds
```

Apéndice E

Implementación del proyecto 5

Clase *Persona.java*

```
/**
 * Clase que modela objetos de tipo Persona
 * una persona solo tiene nombre
 * @author: Cinthia Rodriguez Maya
 * @Fecha: Abril de 2014
 */

public class Persona {

   //nombre de la persona
   private String nombre;

   /**
    *  Metodo constructor
    *  @param: nom inicializa el nombre de la persona
    */
   public Persona(String nom) {
      nombre=nom;
   }

   /**
    *  Metodo getNombre
    *  return el nombre
    */
   public String getNombre() {
      return nombre;
   }

   /**
    *  Metodo toString
    *  return una representacion en cadena del nombre
    */
   public String toString() {
      return nombre;
   }

}
```

Clase *Aerolinea.java*

```
/**
 * Clase que simula la venta de boletos de una aerolinea
 * esta aerolinea cuenta con un solo avion
 * @author: Cinthia Rodriguez Maya
 * @Fecha: Abril de 2014
 */
import java.util.Scanner;

public class Aerolinea {

   //variables de entrada para leer de la entrada estandar
   Scanner entrada = new Scanner(System.in);
   Scanner entrada2 = new Scanner(System.in);
   //arreglo unidimensional que representa los asientos del avion
   Persona[] avion = new Persona[10];
   int asiento=0;
   String nombre;
   /**
    *  Metodo menu
    *  pregunta al usuario lo que quiere hacer:
    *  vender un boleto
    *  cancelar un boleto
    *  ver todos los asientos
    *  salir del programa
    */
   public void menu() {
      boolean bandera=true;
      int opcion=0;
      while(bandera) {
         System.out.println("Selecciona una opcion");
         System.out.println("1. Vender boleto");
         System.out.println("2. Cancelar boleto");
         System.out.println("3. Consultar asientos");
         System.out.println("4. Salir");
         opcion=entrada.nextInt();
         switch(opcion) {
            case 1:
               venderBoleto();
               break;
            case 2:
               cancelarBoleto();
               break;
            case 3:
               verTodo();
               break;
            case 4:
               System.exit(1);
               break;
            default:
               System.out.println("Opcion invalida");
               break;
         }
      }
   }
```

```
55
56      /**
57       * Metodo venderBoleto
58       * asigna un boleto a un cliente, el cual puede ser en primera clase
59       * o en clase economica
60       */
61      public void venderBoleto() {
62         boolean bandera;
63         int opcion=0;
64         int cambio=0;
65         int seccion=0;
66         int contador=0;
67         System.out.println("Venta de boletos");
68         System.out.println("Por favor escriba 1 para primera clase");
69         System.out.println("Por favor escriba 2 para clase economica");
70         opcion=entrada.nextInt();
71         if(opcion==1) {
72            for(int i=0; i<avion.length/2; i++) {
73               if(avion[i]==null) {
74                  contador++;
75               }
76            }
77            //Aun hay asientos disponibles en primera clase
78            if(contador!=0) {
79               venderBoletoPrimeraClase();
80            } else {
81               contador=0;
82               for(int i=5; i<avion.length; i++) {
83                  if(avion[i]==null) {
84                     contador++;
85                  }
86               }
87               //Posibilidad de compra en clase economica
88               if(contador!=0) {
89                  bandera=true;
90                  while(bandera) {
91                     System.out.println("Ya no hay asientos disponibles en
                        primera clase pero si en clase economica");
92                     System.out.println("1. Comprar boleto en clase economica
                        ");
93                     System.out.println("2. Cancelar compra");
94                     cambio=entrada.nextInt();
95                     if(cambio==1) {
96                        venderBoletoClaseEconomica();
97                        bandera=false;
98                     } else if(cambio==2) {
99                        System.out.println("El proximo vuelo sale en 3 horas"
                           );
100                       bandera=false;
101                    } else {
102                       System.out.println("Introduce una opcion valida");
103                    }
104                 }
105              } else {
106                 System.out.println("Lo sentimos, se agotaron los boletos");
107                 System.out.println("El proximo vuelo sale en 3 horas");
```

```
108             }
109          }
110       } else if(opcion==2) {
111          contador=0;
112          for(int i=5; i<avion.length; i++) {
113             if(avion[i]==null) {
114                contador++;
115             }
116          }
117          //Aun hay asientos disponibles en clase economica
118          if(contador!=0) {
119             venderBoletoClaseEconomica();
120          } else {
121             bandera=true;
122             contador=0;
123             for(int i=0; i<avion.length/2; i++) {
124                if(avion[i]==null) {
125                   contador++;
126                }
127             }
128             if(contador!=0) {
129                while(bandera) {
130                   System.out.println("Ya no hay asientos disponibles en
                         clase economica pero si en primera clase");
131                   System.out.println("1. Comprar boleto en primera clase")
                         ;
132                   System.out.println("2. Cancelar compra");
133                   cambio=entrada.nextInt();
134                   if(cambio==1) {
135                      venderBoletoPrimeraClase();
136                      bandera=false;
137                   } else if(cambio==2) {
138                      System.out.println("El proximo vuelo sale en 3 horas"
                            );
139                      bandera=false;
140                   } else {
141                      System.out.println("Introduce una opcion valida");
142                   }
143                }
144             } else {
145                System.out.println("Lo sentimos, se agotaron los boletos");
146                System.out.println("El proximo vuelo sale en 3 horas");
147             }
148          }
149       } else {
150          System.out.println("Introduce una opcion valida");
151       }
152    }
153
154    /**
155     * Metodo venderBoletoPrimeraClase
156     * asigna un asiento de los lugares 1 a 5
157     * imprime boleto de compra
158     */
159    public void venderBoletoPrimeraClase() {
160       System.out.println("Asientos disponibles");
```

```
161         for(int i=0; i<avion.length/2; i++) {
162            if(avion[i]==null) {
163               System.out.println(i+1);
164            }
165         }
166         System.out.println("Elige uno de los asientos disponibles");
167         asiento=entrada.nextInt();
168         asiento--;
169         //asiento valido
170         if(asiento>=0 && asiento<5) {
171            //asiento desocupado
172            if(avion[asiento]==null) {
173               System.out.println("Introduce el nombre del cliente");
174               nombre=entrada2.nextLine();
175               Persona p = new Persona(nombre);
176               avion[asiento]=p;
177               imprimirBoleto(p,asiento);
178            } else {
179               System.out.println("Ese asiento ya se encuentra ocupado");
180            }
181         } else {
182            System.out.println("Asiento invalido");
183         }
184      }
185
186      /**
187       * Metodo venderBoletoClaseEconomica
188       * asigna un asiento de los lugares 6 a 10
189       * imprime boleto de compra
190       */
191      public void venderBoletoClaseEconomica() {
192         System.out.println("Asientos disponibles");
193         for(int i=5; i<avion.length; i++) {
194            if(avion[i]==null) {
195               System.out.println(i+1);
196            }
197         }
198         System.out.println("Elige uno de los asientos disponibles");
199         asiento=entrada.nextInt();
200         asiento--;
201         //asiento valido
202         if(asiento>4 && asiento<10) {
203            //asiento desocupado
204            if(avion[asiento]==null) {
205               System.out.println("Introduce el nombre del cliente");
206               nombre=entrada2.nextLine();
207               Persona p = new Persona(nombre);
208               avion[asiento]=p;
209               imprimirBoleto(p,asiento);
210            } else {
211               System.out.println("Ese asiento ya se encuentra ocupado");
212            }
213         } else {
214            System.out.println("Asiento invalido");
215         }
216      }
```

```
217
218     /**
219      * Metodo cancelarBoleto
220      * solicita nombre y lugar para dejar este asiento como vacio
221      */
222     public void cancelarBoleto() {
223        String nombre;
224        int lugar=0;
225        Persona p;
226        boolean sePudo=false;
227        System.out.println("Para cancelar un boleto proporciona la siguiente
               informacion");
228        System.out.println("Nombre del cliente");
229        nombre=entrada2.nextLine();
230        System.out.println("Numero de asiento");
231        lugar=entrada.nextInt();
232        for(int i=0; i<avion.length; i++) {
233           p=avion[i];
234           if(p!=null) {
235              if(p.getNombre().equals(nombre) && i==(lugar-1)) {
236                 avion[i]=null;
237                 sePudo=true;
238                 System.out.println("El boleto ha sido cancelado");
239              }
240           }
241        }
242        if(!sePudo) {
243           System.out.println("Datos incorrectos");
244           System.out.println("No se pudo cancelar el boleto");
245        }
246     }
247
248     /**
249      * Metodo verTodo
250      * muestra la informacion de cada asiento del avion
251      * si el lugar esta ocupado muestra quien lo compro
252      * si el lugar esta vacio indica que el asiento esta desocupado
253      */
254     public void verTodo() {
255        System.out.println("Informacion del avion");
256        for(int i=0; i<avion.length; i++) {
257           if(i==0) {
258              System.out.println("Primera clase");
259           }
260           if(i==5) {
261              System.out.println("Clase economica");
262           }
263           if(avion[i]!=null) {
264              System.out.print(i+1 + " ");
265              System.out.println(avion[i]);
266           } else {
267              System.out.println((i+1) + " Desocupado");
268           }
269        }
270     }
271
```

```
272     /**
273      * Metodo imprimirBoleto
274      * imprime el boleto de compra
275      * @param: p, persona que compro el boleto
276      * @param: lugar, asiento solicitado
277      */
278     public void imprimirBoleto(Persona p, int lugar) {
279        if(lugar<5) {
280           System.out.println("**************************************");
281           System.out.println("Pasajero: " + p.getNombre());
282           System.out.println("Asiento: " + (lugar+1));
283           System.out.println("PRIMERA CLASE");
284           System.out.println("Gracias por preferir nuestra aerolinea");
285           System.out.println("**************************************");
286        } else {
287           System.out.println("**************************************");
288           System.out.println("Pasajero: " + p.getNombre());
289           System.out.println("Asiento: " + (lugar+1));
290           System.out.println("CLASE ECONOMICA");
291           System.out.println("Gracias por preferir nuestra aerolinea");
292           System.out.println("**************************************");
293        }
294     }
295  }
```

Clase *UsaAerolinea.java*

```
1  /**
2   * Clase que contiene el metodo main del programa de la Aerolinea
3   * @author: Cinthia Rodriguez Maya
4   * @Fecha: Abril de 2014
5   */
6
7  public class UsaAerolinea {
8
9     public static void main(String[] args) {
10
11       Aerolinea a = new Aerolinea();
12       a.menu();
13    }
14 }
```

Ejecución de nuestro proyecto:

```
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
3
Informacion del avion
Primera clase
1 Desocupado
2 Desocupado
3 Desocupado
4 Desocupado
5 Desocupado
Clase economica
6 Desocupado
```

```
7 Desocupado
8 Desocupado
9 Desocupado
10 Desocupado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
Asientos disponibles
1
2
3
4
5
Elige uno de los asientos disponibles
3
Introduce el nombre del cliente
cinthia rodriguez
**************************************************
Pasajero: cinthia rodriguez
Asiento: 3
PRIMERA CLASE
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
3
Informacion del avion
Primera clase
1 Desocupado
2 Desocupado
3 cinthia rodriguez
4 Desocupado
5 Desocupado
Clase economica
6 Desocupado
7 Desocupado
8 Desocupado
9 Desocupado
10 Desocupado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
Asientos disponibles
1
2
4
5
Elige uno de los asientos disponibles
3
Ese asiento ya se encuentra ocupado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
```

```
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
2
Asientos disponibles
6
7
8
9
10
Elige uno de los asientos disponibles
11
Asiento invalido
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
2
Asientos disponibles
6
7
8
9
10
Elige uno de los asientos disponibles
6
Introduce el nombre del cliente
fulanito
**************************************************
Pasajero: fulanito
Asiento: 6
CLASE ECONOMICA
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
3
Informacion del avion
Primera clase
1 Desocupado
2 Desocupado
3 cinthia rodriguez
4 Desocupado
5 Desocupado
Clase economica
6 fulanito
7 Desocupado
8 Desocupado
9 Desocupado
10 Desocupado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
Asientos disponibles
```

```
1
2
4
5
Elige uno de los asientos disponibles
1
Introduce el nombre del cliente
jose galaviz
**************************************************
Pasajero: jose galaviz
Asiento: 1
PRIMERA CLASE
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
Asientos disponibles
2
4
5
Elige uno de los asientos disponibles
2
Introduce el nombre del cliente
amparo lopez
**************************************************
Pasajero: amparo lopez
Asiento: 2
PRIMERA CLASE
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
Asientos disponibles
4
5
Elige uno de los asientos disponibles
4
Introduce el nombre del cliente
amparo lopez
**************************************************
Pasajero: amparo lopez
Asiento: 4
PRIMERA CLASE
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
```

```
Asientos disponibles
5
Elige uno de los asientos disponibles
5
Introduce el nombre del cliente
gerardo aviles
**************************************************
Pasajero: gerardo aviles
Asiento: 5
PRIMERA CLASE
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
Ya no hay asientos disponibles en primera clase pero si en clase economica
1. Comprar boleto en clase economica
2. Cancelar compra
1
Asientos disponibles
7
8
9
10
Elige uno de los asientos disponibles
7
Introduce el nombre del cliente
canek pelaez
**************************************************
Pasajero: canek pelaez
Asiento: 7
CLASE ECONOMICA
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
3
Informacion del avion
Primera clase
1 jose galaviz
2 amparo lopez
3 cinthia rodriguez
4 amparo lopez
5 gerardo aviles
Clase economica
6 fulanito
7 canek pelaez
8 Desocupado
9 Desocupado
10 Desocupado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
2
Para cancelar un boleto proporciona la siguiente informacion
Nombre del cliente
sutanito
Numero de asiento
1
```

```
Datos incorrectos
No se pudo cancelar el boleto
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
3
Informacion del avion
Primera clase
1 jose galaviz
2 amparo lopez
3 cinthia rodriguez
4 amparo lopez
5 gerardo aviles
Clase economica
6 fulanito
7 canek pelaez
8 Desocupado
9 Desocupado
10 Desocupado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
2
Para cancelar un boleto proporciona la siguiente informacion
Nombre del cliente
amparo lopez
Numero de asiento
2
El boleto ha sido cancelado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
3
Informacion del avion
Primera clase
1 jose galaviz
2 Desocupado
3 cinthia rodriguez
4 amparo lopez
5 gerardo aviles
Clase economica
6 fulanito
7 canek pelaez
8 Desocupado
9 Desocupado
10 Desocupado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
Asientos disponibles
2
Elige uno de los asientos disponibles
2
Introduce el nombre del cliente
salvador lopez
**************************************************
Pasajero: salvador lopez
Asiento: 2
```

```
PRIMERA CLASE
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
3
Informacion del avion
Primera clase
1 jose galaviz
2 salvador lopez
3 cinthia rodriguez
4 amparo lopez
5 gerardo aviles
Clase economica
6 fulanito
7 canek pelaez
8 Desocupado
9 Desocupado
10 Desocupado
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
2
Asientos disponibles
8
9
10
Elige uno de los asientos disponibles
8
Introduce el nombre del cliente
sutano
**************************************************
Pasajero: sutano
Asiento: 8
CLASE ECONOMICA
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
2
Asientos disponibles
9
10
Elige uno de los asientos disponibles
9
Introduce el nombre del cliente
mengano
**************************************************
Pasajero: mengano
Asiento: 9
CLASE ECONOMICA
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
```

```
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
2
Asientos disponibles
10
Elige uno de los asientos disponibles
10
Introduce el nombre del cliente
cinthia maya
**************************************************
Pasajero: cinthia maya
Asiento: 10
CLASE ECONOMICA
Gracias por preferir nuestra aerolinea
**************************************************
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
1
Venta de boletos
Por favor escriba 1 para primera clase
Por favor escriba 2 para clase economica
1
Lo sentimos, se agotaron los boletos
El proximo vuelo sale en 3 horas
Selecciona una opcion
1. Vender boleto
2. Cancelar boleto
3. Consultar asientos
4. Salir
4
```

Apéndice F

Implementación del proyecto 6

Clase *UsoConjunto.java*

```
/**
 * Clase UsaConjunto
 * Programa que realiza operaciones entre conjuntos mediante listas
    utilizando la clase ArrayList de Java
 *
 * @author Cinthia Rodriguez Maya
 * @version: Marzo de 2014
 */
package conjuntos;

import java.util.ArrayList;

public class Conjunto {

   /**
    * Metodo unir
    * Une dos conjuntos
    * @param c1, el primer conjunto
    * @param c2, el segundo conjunto
    * return el conjunto resultado de la union de c1 con c2
    */
   public static ArrayList unir(ArrayList c1, ArrayList c2) {
      ArrayList conjUnion = new ArrayList();
      for(int i=0; i<c1.size(); i++) {
         conjUnion.add(c1.get(i));
      }
      for(int j=0; j<c2.size(); j++) {
         if(!conjUnion.contains(c2.get(j))) {
            conjUnion.add(c2.get(j));
         }
      }
      return conjUnion;
   }

   /**
    * Metodo intersectar
    * Calcula la interseccion de dos conjuntos
    * @param c1, el primer conjunto
    * @param c2, el segundo conjunto
```

```
39          * return el conjunto resultado de la interseccion de c1 y c2
40          */
41         public static ArrayList intersectar(ArrayList c1, ArrayList c2) {
42            ArrayList conjInterseccion = new ArrayList();
43            for(int i=0; i<c1.size();i++) {
44               if(c2.contains(c1.get(i))) {
45                  conjInterseccion.add(c1.get(i));
46               }
47            }
48            return conjInterseccion;
49         }
50
51         /**
52          * Metodo calcularDiferencia
53          * Calcula la diferencia de dos conjuntos A\B
54          * @param c1, el primer conjunto
55          * @param c2, el segundo conjunto
56          * return el conjunto resultado de la diferencia del primer conjunto
                menos el segundo
57          */
58         public static ArrayList calcularDiferencia(ArrayList c1, ArrayList c2)
               {
59            ArrayList conjDif = new ArrayList();
60            for(int i=0; i<c2.size(); i++) {
61               Object v=c2.get(i);
62               if(!c1.contains(v)) {
63                  conjDif.add(v);
64               }
65            }
66            return conjDif;
67         }
68
69         /**
70          * Metodo obtenerCardinalidad
71          * Calcula el numero de elementos  que tiene un conjunto
72          * @param c, el conjunto
73          * return la cantidad de elementos contenidos en el conjunto
74          */
75         public static int obtenerCardinalidad(ArrayList c) {
76            return c.size();
77         }
78
79         /**
80          * Metodo pertenece
81          * Indica si un elemento pertenece a un conjunto
82          * @param c, el conjunto
83          * @param elem, el elemento a verificar si pertenece o no
84          * return true si elem pertenece a c, return false en caso contrario
85          */
86         public static boolean pertenece(ArrayList c, int elem) {
87            return c.contains(elem);
88         }
89
90         /**
91          * Metodo imprimirConjunto
92          * Imprime los elementos contenidos en un conjunto
```

```
93          * @param c, el conjunto a imprimir
94          */
95         public static void imprimeConjunto(ArrayList c) {
96            String cadena="";
97            for(int i=0; i<c.size(); i++) {
98               cadena=cadena+c.get(i)+ " ";
99            }
100           return cadena;
101        }
102
103        /**
104         * Metodo main
105         */
106        public static void main(String[] args) {
107           ArrayList c1 = new ArrayList();
108           ArrayList c2 = new ArrayList();
109           ArrayList c3 = new ArrayList();
110           ArrayList c4 = new ArrayList();
111           ArrayList c5 = new ArrayList();
112           c1.add(1);
113           c1.add(2);
114           c1.add(3);
115           c1.add(4);
116           c1.add(5);
117           c1.add(10);
118           c1.add(12);
119           c2.add(2);
120           c2.add(3);
121           c2.add(17);
122           System.out.println("Conjunto 1");
123           System.out.println(imprimirConjunto(c1));
124           System.out.println("Conjunto 2");
125           System.out.println(imprimirConjunto(c2));
126           System.out.println("Cardinalidad del conjunto 1");
127           System.out.println(obtenerCardinalidad(c1));
128           System.out.println("Cardinalidad del conjunto 2");
129           System.out.println(obtenerCardinalidad(c2));
130           c3=unir(c1,c2);
131           System.out.println("Uni n del Conjunto 1 y 2");
132           System.out.println(imprimirConjunto(c3));
133           c4=intersectar(c1,c2);
134           System.out.println("Intersecci n del Conjunto 1 y 2");
135           System.out.println(imprimirConjunto(c4));
136           System.out.println("Conjunto 1 menos Conjunto 2");
137           c5=obtenerDiferencia(c1,c2);
138           System.out.println(imprimirConjunto(c5));
139           System.out.println("Pertenece el elemento 1 al Conjunto 1");
140           System.out.println(pertenece(c1,1));
141           System.out.println("Pertenece el elemento 20 al Conjunto 1");
142           System.out.println(pertenece(c1,20));
143        }
144     }
```

Ejecución de nuestro proyecto:

```
Conjunto 1
```

```
1 2 3 4 5 10 12
Conjunto 2
2 3 17
Cardinalidad del conjunto 1
7
Cardinalidad del conjunto 2
3
Uni n del Conjunto 1 y 2
1 2 3 4 5 10 12 17
Intersecci n del Conjunto 1 y 2
2 3
Conjunto 1 menos Conjunto 2
1 4 5 10 12
Pertenece el elemento 1 al Conjunto 1
true
Pertenece el elemento 20 al Conjunto 1
false
```

Clase *TestConjunto.java* para las pruebas unitarias:

```java
/**
 * Clase para pruebas unitarias de la clase {@link UsaConjunto.java}.
 * @author: Cinthia Rodriguez Maya
 * @version: Marzo de 2014
 */

package conjuntos.test;

import conjuntos.*;
import java.util.ArrayList;
import org.junit.Assert;
import org.junit.Test;

public class TestConjunto {

   //Conjuntos
   ArrayList c1 = new ArrayList();
   ArrayList c2 = new ArrayList();
   ArrayList c3 = new ArrayList();
   ArrayList c4 = new ArrayList();

   //Inicializamos nuestros conjuntos para las pruebas
   public TestConjunto() {
      c1.add(1);
      c1.add(3);
      c1.add(5);
      c1.add(7);

      c2.add(10);
      c2.add(20);
      c2.add(30);
      c2.add(40);
      c2.add(50);

      c3.add(2);
      c3.add(4);
      c3.add(6);
      c3.add(8);

      c4.add(1);
```

```
41          c4.add(2);
42          c4.add(3);
43      }
44
45      /**
46       * Prueba unitaria para {@link unir#UsaConjunto}.
47       */
48      @Test public void testUnir() {
49          ArrayList u1 = new ArrayList();
50          ArrayList u2 = new ArrayList();
51          ArrayList u3 = new ArrayList();
52
53          u1=UsaConjunto.unir(c1,c3);
54          u2=UsaConjunto.unir(c1,c2);
55          u3=UsaConjunto.unir(c2,c3);
56          Assert.assertTrue(u1.get(0).equals(1));
57          Assert.assertTrue(u1.get(1).equals(3));
58          Assert.assertTrue(u1.get(2).equals(5));
59          Assert.assertTrue(u1.get(3).equals(7));
60          Assert.assertTrue(u1.get(4).equals(2));
61          Assert.assertTrue(u1.get(5).equals(4));
62          Assert.assertTrue(u1.get(6).equals(6));
63          Assert.assertTrue(u1.get(7).equals(8));
64
65          Assert.assertTrue(u2.get(0).equals(1));
66          Assert.assertTrue(u2.get(1).equals(3));
67          Assert.assertTrue(u2.get(2).equals(5));
68          Assert.assertTrue(u2.get(3).equals(7));
69          Assert.assertTrue(u2.get(4).equals(10));
70          Assert.assertTrue(u2.get(5).equals(20));
71          Assert.assertTrue(u2.get(6).equals(30));
72          Assert.assertTrue(u2.get(7).equals(40));
73          Assert.assertTrue(u2.get(8).equals(50));
74
75          Assert.assertTrue(u3.get(0).equals(10));
76          Assert.assertTrue(u3.get(1).equals(20));
77          Assert.assertTrue(u3.get(2).equals(30));
78          Assert.assertTrue(u3.get(3).equals(40));
79          Assert.assertTrue(u3.get(4).equals(50));
80          Assert.assertTrue(u3.get(5).equals(2));
81          Assert.assertTrue(u3.get(6).equals(4));
82          Assert.assertTrue(u3.get(7).equals(6));
83          Assert.assertTrue(u3.get(8).equals(8));
84      }
85
86      /**
87       * Prueba unitaria para {@link intersectar#UsaConjunto}.
88       */
89      @Test public void testIntersectar() {
90          ArrayList i1 = new ArrayList();
91          ArrayList i2 = new ArrayList();
92          ArrayList i3 = new ArrayList();
93
94          i1=UsaConjunto.intersectar(c1,c4);
95          i2=UsaConjunto.intersectar(c2,c2);
96          i3=UsaConjunto.intersectar(c3,c2);
```

```
97
98          Assert.assertTrue(i1.get(0).equals(1));
99          Assert.assertTrue(i1.get(1).equals(3));
100
101         Assert.assertTrue(i2.get(0).equals(10));
102         Assert.assertTrue(i2.get(1).equals(20));
103         Assert.assertTrue(i2.get(2).equals(30));
104         Assert.assertTrue(i2.get(3).equals(40));
105         Assert.assertTrue(i2.get(4).equals(50));
106
107         Assert.assertTrue(i3.size()==0);
108     }
109
110     /**
111      * Prueba unitaria para {@link obtenerDiferencia#UsaConjunto}.
112      */
113     @Test public void testObtenerDiferencia() {
114         ArrayList d1 = new ArrayList();
115         ArrayList d2 = new ArrayList();
116         ArrayList d3 = new ArrayList();
117
118         d1=UsaConjunto.obtenerDiferencia(c3,c3);
119         d2=UsaConjunto.obtenerDiferencia(c1,c4);
120         d3=UsaConjunto.obtenerDiferencia(c2,c3);
121
122         Assert.assertTrue(d1.size()==0);
123         Assert.assertTrue(d2.get(0).equals(5));
124         Assert.assertTrue(d2.get(1).equals(7));
125         Assert.assertTrue(d3.get(0).equals(10));
126         Assert.assertTrue(d3.get(1).equals(20));
127         Assert.assertTrue(d3.get(2).equals(30));
128         Assert.assertTrue(d3.get(3).equals(40));
129         Assert.assertTrue(d3.get(4).equals(50));
130     }
131
132     /**
133      * Prueba unitaria para {@link obtenerCardinalidad#UsaConjunto}.
134      */
135     @Test public void testObtenerCardinalidad() {
136         Assert.assertTrue(UsaConjunto.obtenerCardinalidad(c1)==4);
137         Assert.assertTrue(UsaConjunto.obtenerCardinalidad(c2)==5);
138         Assert.assertTrue(UsaConjunto.obtenerCardinalidad(c3)==4);
139         Assert.assertTrue(UsaConjunto.obtenerCardinalidad(c4)==3);
140     }
141
142     /**
143      * Prueba unitaria para {@link pertenece#UsaConjunto}.
144      */
145     @Test public void testPertenece() {
146         Assert.assertTrue(UsaConjunto.pertenece(c1,1)==true);
147         Assert.assertTrue(UsaConjunto.pertenece(c1,3)==true);
148         Assert.assertTrue(UsaConjunto.pertenece(c1,5)==true);
149         Assert.assertTrue(UsaConjunto.pertenece(c1,7)==true);
150         Assert.assertTrue(UsaConjunto.pertenece(c1,10)==false);
151         Assert.assertTrue(UsaConjunto.pertenece(c1,12)==false);
152         Assert.assertTrue(UsaConjunto.pertenece(c1,-4)==false);
```

```
153         Assert.assertTrue(UsaConjunto.pertenece(c1,0)==false);
154 
155         Assert.assertTrue(UsaConjunto.pertenece(c2,10)==true);
156         Assert.assertTrue(UsaConjunto.pertenece(c2,20)==true);
157         Assert.assertTrue(UsaConjunto.pertenece(c2,30)==true);
158         Assert.assertTrue(UsaConjunto.pertenece(c2,40)==true);
159         Assert.assertTrue(UsaConjunto.pertenece(c2,50)==true);
160         Assert.assertTrue(UsaConjunto.pertenece(c2,1)==false);
161         Assert.assertTrue(UsaConjunto.pertenece(c2,22)==false);
162         Assert.assertTrue(UsaConjunto.pertenece(c2,-8)==false);
163         Assert.assertTrue(UsaConjunto.pertenece(c2,38)==false);
164         Assert.assertTrue(UsaConjunto.pertenece(c2,60)==false);
165 
166         Assert.assertTrue(UsaConjunto.pertenece(c3,2)==true);
167         Assert.assertTrue(UsaConjunto.pertenece(c3,4)==true);
168         Assert.assertTrue(UsaConjunto.pertenece(c3,6)==true);
169         Assert.assertTrue(UsaConjunto.pertenece(c3,8)==true);
170         Assert.assertTrue(UsaConjunto.pertenece(c3,11)==false);
171         Assert.assertTrue(UsaConjunto.pertenece(c3,-10)==false);
172         Assert.assertTrue(UsaConjunto.pertenece(c3,44)==false);
173         Assert.assertTrue(UsaConjunto.pertenece(c3,90)==false);
174 
175         Assert.assertTrue(UsaConjunto.pertenece(c4,1)==true);
176         Assert.assertTrue(UsaConjunto.pertenece(c4,2)==true);
177         Assert.assertTrue(UsaConjunto.pertenece(c4,3)==true);
178         Assert.assertTrue(UsaConjunto.pertenece(c4,-4)==false);
179         Assert.assertTrue(UsaConjunto.pertenece(c4,100)==false);
180         Assert.assertTrue(UsaConjunto.pertenece(c4,-33)==false);
181     }
182 
183     /**
184      * Prueba unitaria para {@link imprimirConjunto#UsaConjunto}.
185      */
186     @Test public void testImprimirConjunto() {
187         String con1="1 3 5 7 ";
188         String con2="10 20 30 40 50 ";
189         String con3="2 4 6 8 ";
190         String con4="1 2 3 ";
191         Assert.assertTrue(UsaConjunto.imprimirConjunto(c1).equals(con1));
192         Assert.assertTrue(UsaConjunto.imprimirConjunto(c2).equals(con2));
193         Assert.assertTrue(UsaConjunto.imprimirConjunto(c3).equals(con3));
194         Assert.assertTrue(UsaConjunto.imprimirConjunto(c4).equals(con4));
195     }
196 }
```

Ejecución de las pruebas unitarias

```
Buildfile: /home/cindy/Escritorio/proyecto5/build.xml

compile.proyecto5:
    [javac] Compiling 2 source files to /home/cindy/Escritorio/proyecto5/build
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.

proyecto5.jar:
      [jar] Building jar: /home/cindy/Escritorio/proyecto5/practica5.jar

test:
    [junit] Running conjuntos.test.TestConjunto
```

```
    [junit] Testsuite: conjuntos.test.TestConjunto
    [junit] Tests run: 6, Failures: 0, Errors: 0, Time elapsed: 0.082 sec
    [junit] Tests run: 6, Failures: 0, Errors: 0, Time elapsed: 0.082 sec
    [junit]
    [junit] Testcase: testUnir took 0.014 sec
    [junit] Testcase: testIntersectar took 0.001 sec
    [junit] Testcase: testObtenerDiferencia took 0.001 sec
    [junit] Testcase: testObtenerCardinalidad took 0.001 sec
    [junit] Testcase: testPertenece took 0.001 sec
    [junit] Testcase: testImprimirConjunto took 0.001 sec

BUILD SUCCESSFUL
Total time: 4 seconds
```

Apéndice G

Implementación del proyecto 7

Clase *Buscaminas.java*

```
/**
 * Programa que genera un campo con minas y muestra la cantidad de minas
    adyacentes
 * @author: Cinthia Rodriguez Maya
 * @Fecha: Abril de 2014
 */
import java.util.Random;

public class Buscaminas {

   public static void main(String[] args) {

      Random aleatorios = new Random();
      int m=aleatorios.nextInt(5)+4;
      int n=aleatorios.nextInt(5)+4;
      //La entrada es un arreglo de tipo char
      //A lo menos la entrada es de 4x4 y a lo m s de 8x8
      char[][] entrada = new char[m][n];
      //Recorremos la matriz
      for(int i=0; i<entrada.length; i++) {
         for(int j=0; j<entrada[i].length; j++) {
            //La probalidad de generar una bomba es de 1 entre 7
            int bomba=aleatorios.nextInt(7);
            if(bomba==0) {
               entrada[i][j]='*';
            } else {
               entrada[i][j]='.';
            }
         }
      }

      //Imprimimos la matriz
      for(int i=0; i<entrada.length; i++) {
         for(int j=0; j<entrada[i].length; j++) {
            System.out.print(entrada[i][j]);
         }
         System.out.println();
      }

```

```
39          //Ahora vamos a etiquetar con los n meros
40          int[][] salida = new int[m][n];
41
42          //Recorremos la matriz de salida y comparamos con que hay en la de
                entrada
43          for(int i=0; i<entrada.length; i++) {
44             for(int j=0; j<entrada[i].length; j++) {
45                if(entrada[i][j]=='*') {
46                   salida[i][j]=9;
47                   //vemos si es v lido marcar los alrededores
48                   if(i-1>=0 && j-1>=0) {
49                      salida[i-1][j-1]++;
50                   }
51                   if(i-1>=0) {
52                      salida[i-1][j]++;
53                   }
54                   if(i-1>=0 && j+1<n) {
55                      salida[i-1][j+1]++;
56                   }
57                   if(j-1>=0) {
58                      salida[i][j-1]++;
59                   }
60                   if(j+1<n) {
61                      salida[i][j+1]++;
62                   }
63                   if(i+1<m && j-1>=0) {
64                      salida[i+1][j-1]++;
65                   }
66                   if(i+1<m) {
67                      salida[i+1][j]++;
68                   }
69                   if(i+1<m && j+1<n) {
70                      salida[i+1][j+1]++;
71                   }
72                }
73             }
74          }
75
76          System.out.println();
77          for(int i=0; i<salida.length; i++) {
78             for(int j=0; j<salida[i].length; j++) {
79                if(salida[i][j]>=9) {
80                   System.out.print("*");
81                } else {
82                   System.out.print(salida[i][j]);
83                }
84             }
85             System.out.println();
86          }
87       }
88    }
```

Ejecución de nuestro proyecto

```
.......
.***.*.
..*....
......*
...**..

1232211
1***2*1
13*3222
012322*
001**21
```

Apéndice H

Implementación del proyecto 8

Clase *Empleado.java*

```
1  /**
2   * Clase Empleado
3   * Programa que modela el empleado de una empresa
4   *
5   * @author Cinthia Rodriguez Maya
6   * @version: Abril de 2014
7   */
8
9  public class Empleado {
10
11    protected String nombre;
12    protected String cedula;
13    protected int edad;
14    protected double salario;
15
16    //Constructor
17    public Empleado(String nombre, String cedula, int edad, double salario)
          {
18       if(edad >= 18 && edad <= 45 && salario >= 3000 && salario <= 15000){
19          this.nombre = nombre;
20          this.cedula = cedula;
21          this.edad = edad;
22          this.salario = salario;
23       } else {
24          if(edad < 18 || edad > 45) {
25             System.out.println("Edad fuera de rango");
26          }
27          if(salario<3000 || salario>15000) {
28             System.out.println("Salario fuera de rango");
29          }
30       }
31    }
32
33    /**
34     * Metodo obtenerTipo
35     * return el tipo de empleado segun salario
36     */
37    public String obtenerTipo() {
38       String cad="";
```

```
39          if(salario < 5000) {
40             cad="Becario";
41          } else if(salario > 5000 && salario < 10000) {
42             cad="Empleado A";
43          } else if (salario>10000 && salario < 150000) {
44             cad="Empleado B";
45          } else if(salario==15000) {
46             cad="Empleado Base";
47          }
48          return cad;
49       }
50
51       //Metodo toString
52       public String toString() {
53          return "Nombre.-" + "\n" + "\t" + this.nombre + "\n" + "Cedula.-" +
                "\n" + "\t" + this.cedula + "\n" + "Salario.-" + "\n" + "\t" + "$
                " + this.salario;
54       }
55    }
```

Clase *Programador.java*

```
1   /**
2    * Clase Empleado
3    * Programa que modela un programador, extiende a la clase Empleado
4    *
5    * @author Cinthia Rodriguez Maya
6    * @version: Abril de 2014
7    */
8
9   public class Programador extends Empleado {
10
11      private int lineasDeCodigoPorHora;
12      private int lenguajesDomina;
13
14      //Constructor
15      public Programador(String nombre, String cedula, int edad, double
           salario, int lineasDeCodigoPorHora, int lenguajesDomina) {
16          super(nombre, cedula, edad, salario);
17          if(lenguajesDomina >= 3) {
18             this.lineasDeCodigoPorHora = lineasDeCodigoPorHora;
19             this.lenguajesDomina = lenguajesDomina;
20          }
21      }
22
23      /**
24       * Metodo obtenerTipoProgramador
25       * return el tipo de programador si este es Intermedio o Avanzado
26       */
27      public String obtenerTipoProgramador() {
28          if(this.lenguajesDomina < 5) {
29             return "Intermedio";
30          } else {
31             return "Avanzado";
32          }
33      }
```

```
34
35      //Metodo toString
36      public String toString() {
37          return "Nombre.-" + "\n" + "\t" + this.nombre + "\n" + "Cedula.-" +
                "\n" + "\t" + this.cedula + "\n" + "Edad.-" + "\n" + "\t" + this.
                edad + "\n" + "Salario.-" + "\n" + "\t" + "$" + this.salario + "\
                n" + "Lineas De Codigo Por Hora.-" + "\n" + "\t" + this.
                lineasDeCodigoPorHora + "\n" + "Cantidad De Lenguajes Que Domina
                .-" + "\n" + "\t" + this.lenguajesDomina;
38      }
39  }
```

Clase *UsoEmpresa.java*

```
1   /**
2    * Clase UsoEmpresa
3    * Clase que contiene el metodo main para ejecutar el programa de los
         empleados
4    * de una empresa de computacion
5    *
6    * @author Cinthia Rodriguez Maya
7    * @version: Abril de 2014
8    */
9
10  public class UsoEmpresa {
11
12      public static void main(String[] args) {
13
14          Empleado e1 = new Empleado("Pedro Perez Pereira", "18237374", 27,
                12500);
15          Programador p1 = new Programador("Anita Lopez Marquez", "38475957",
                21, 9000, 120, 3);
16          System.out.println(e1);
17          System.out.println(p1);
18          System.out.println(e1.obtenerTipo());
19          System.out.println(p1.obtenerTipoProgramador());
20          System.out.println(p1.obtenerTipo());
21      }
22  }
```

Ejecución de nuestro proyecto

```
Nombre.-
       Pedro Perez Pereira
Cedula.-
       18237374
Salario.-
      \$12500.0
Nombre.-
       Anita Lopez Marquez
Cedula.-
       38475957
Edad.-
       21
Salario.-
      \$9000.0
Lineas De Codigo Por Hora.-
       120
Cantidad De Lenguajes Que Domina.-
       3
Empleado B
Intermedio
Empleado A
```

Apéndice I

Implementación del proyecto 9

Clase *Mensaje.java*

```
/**
 * Clase Mensaje
 * Clase que modela un mensaje compuesto por remitente y asunto
 *
 * @author Cinthia Rodriguez Maya
 * @version: Abril de 2014
 */

public class Mensaje {

   String remitente;
   String asunto;

   //Constructor
   public Mensaje(String r, String a) {
      remitente=r;
      asunto=a;
   }

   //Metodo toString
   public String toString() {
      return "Mensaje de  " + remitente + " -> " + asunto;
   }
}
```

Clase *Contestadora.java*

```
/**
 * Clase Contestadora
 * Clase que simula una contestadora telefonica
 * la cual puede almacenar hasta 10 objetos de tipo Mensaje
 *
 * @author Cinthia Rodriguez Maya
 * @version: Abril de 2014
 */

import java.util.Scanner;

public class Contestadora {
```

```
13
14      Mensaje[] c = new Mensaje[10];
15
16      /**
17       * Metodo menu
18       * muestra las posibles actividades que se pueden hacer
19       * con la contestadora
20       */
21      public void menu() {
22         int opcion=0;
23         Scanner entrada = new Scanner(System.in);
24         do {
25            System.out.println("1. Guarda Mensaje");
26            System.out.println("2. Borra mensaje segun remitente");
27            System.out.println("3. Borra todos los mensajes");
28            System.out.println("4. Muestra todos los mensajes");
29            System.out.println("5. Salir");
30            opcion=entrada.nextInt();
31            switch(opcion) {
32               case 1:
33                  Scanner entrada2 = new Scanner(System.in);
34                  String r;
35                  String s;
36                  System.out.println("Escribe el nombre del remitente");
37                  r=entrada2.nextLine();
38                  System.out.println("Escribe el mensaje que quieres dejar");
39                  s=entrada2.nextLine();
40                  try {
41                     guardaMensaje(r,s);
42                  } catch(AgendaLlenaException a_llena) {
43                     System.out.println("No es posible guardar mas mensajes "
                           +  a_llena);
44                  }
45                  break;
46               case 2:
47                  Scanner entrada3 = new Scanner(System.in);
48                  String remi;
49                  System.out.println("Escribe el nombre del remitente");
50                  remi=entrada3.nextLine();
51                  borraRemitente(remi);
52                  break;
53               case 3:
54                  borraTodo();
55                  break;
56               case 4:
57                  try {
58                     muestraTodos();
59                  } catch(NoHayDatosException n_datos) {
60                     System.out.println("No hay mensajes que mostrar " +
                        n_datos);
61                  }
62                  break;
63               case 5:
64                  System.out.println("Hasta pronto");
65                  break;
66               default:
```

```
67                  System.out.println("Introduce una opcion valida");
68                  break;
69              }
70          }while(opcion!=5);
71      }
72
73      /**
74       * Metodo guardaMensaje
75       * almacena un mensaje en la contestadora
76       * @param quien, remitente
77       * @param me, el texto del mensaje
78       */
79      public void guardaMensaje(String quien, String me) throws
            AgendaLlenaException {
80          int sePuede=0;
81          for(int i=0; i<c.length; i++) {
82              if(c[i]==null) {
83                  Mensaje m = new Mensaje(quien,me);
84                  c[i]=m;
85                  break;
86              } else {
87                  sePuede++;
88              }
89          }
90          if(sePuede==10) {
91              throw new AgendaLlenaException();
92          } else {
93              System.out.println("El mensaje se ha guardado exitosamente");
94          }
95      }
96
97      /**
98       * Metodo borraRemitente
99       * elimina un mensaje de la contestadora dado el remitente
100      * @param rem, remitente
101      */
102     public void borraRemitente(String rem) {
103         boolean sePudo=false;
104         for(int i=0; i<c.length; i++) {
105             if(c[i]!=null) {
106                 String quien=c[i].remitente;
107                 if(quien.equals(rem)) {
108                     c[i]=null;
109                     sePudo=true;
110                 }
111             }
112         }
113         if(sePudo) {
114             System.out.println("Borrado exitoso");
115         } else {
116             System.out.println("No existe el remitente");
117         }
118     }
119
120     /**
121      * Metodo borraTodo
```

```
122         * elimina todos los mensajes almacenados en la contestadora
123         */
124        public void borraTodo() {
125           for(int i=0; i<c.length; i++) {
126              c[i]=null;
127           }
128           System.out.println("Se borraron todos los mensajes");
129        }
130
131        /**
132         * Metodo muestraTodos
133         * muestra todos los mensajes almacenados en la contestadora
134         */
135        public void muestraTodos() throws NoHayDatosException {
136           int contador=0;
137           for(int i=0; i<c.length; i++) {
138              if(c[i]!=null) {
139                 System.out.println(c[i]);
140                 contador++;
141              }
142           }
143           if(contador==0) {
144              throw new NoHayDatosException();
145           }
146        }
147     }
```

Clase *AgendaLlenaException.java*

```
1   /**
2    * Clase AgendaLlenaException
3    * Clase que define nuestra propia excepcion para indicar que
4    * nuestra agenda del proyecto de la contestadora esta llena
5    *
6    * @author Cinthia Rodriguez Maya
7    * @version: Abril de 2014
8    */
9
10  public class AgendaLlenaException extends Exception {
11
12     public AgendaLlenaException() {
13        super();
14     }
15
16     public AgendaLlenaException(String s) {
17        super(s);
18     }
19  }
```

Clase *NoHayDatosException.java*

```
1   /**
2    * Clase NoHayDatosException
3    * Clase que define nuestra propia excepcion para indicar que
4    * no hay informaci n almacenda en la contestadora
5    *
```

```
 * @author Cinthia Rodriguez Maya
 * @version: Abril de 2014
 */

public class NoHayDatosException extends Exception {

   public NoHayDatosException() {
      super();
   }

   public NoHayDatosException(String s) {
      super(s);
   }
}
```

Clase *PruebaC.java*

```
/**
 * Clase que contiene el metodo main para ejecutar el programa
 * de la contestadora
 * @author Cinthia Rodriguez Maya
 * @version: Abril de 2014
 */

public class PruebaC {

   public static void main(String[] args) {
      Contestadora c = new Contestadora();
      c.menu();
   }
}
```

Ejecución de nuestro proyecto

```
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
4
No hay mensajes que mostrar NoHayDatosException
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
cinthia
Escribe el mensaje que quieres dejar
hola
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
amparo
Escribe el mensaje que quieres dejar
pasa en la tarde a mi oficina
```

```
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje seg n remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
4
Mensaje de  cinthia -> hola
Mensaje de  amparo -> pasa en la tarde a mi oficina
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
cinthia
Escribe el mensaje que quieres dejar
programar es divertido
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
gerardo
Escribe el mensaje que quieres dejar
nos vemos con canek afuera de la biblioteca a las 3 pm
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
4
Mensaje de  cinthia -> hola
Mensaje de  amparo -> pasa en la tarde a mi oficina
Mensaje de  cinthia -> programar es divertido
Mensaje de  gerardo -> nos vemos con canek afuera de la biblioteca a las 3 pm
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
2
Escribe el nombre del remitente
cinthia
Borrado exitoso
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
4
Mensaje de  amparo -> pasa en la tarde a mi oficina
Mensaje de  gerardo -> nos vemos con canek afuera de la biblioteca a las 3 pm
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
salvador
Escribe el mensaje que quieres dejar
espero todo marche bien
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
```

```
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
pepe pecas
Escribe el mensaje que quieres dejar
necesitamos tiempo para entregar la tarea 7
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
4
Mensaje de  salvador -> espero todo marche bien
Mensaje de  amparo -> pasa en la tarde a mi oficina
Mensaje de  pepe pecas -> necesitamos tiempo para entregar la tarea 7
Mensaje de  gerardo -> nos vemos con canek afuera de la biblioteca a las 3 pm
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
mickey mouse
Escribe el mensaje que quieres dejar
ya esta en produccion una nueva pelicula ja-ja
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
cinthia
Escribe el mensaje que quieres dejar
el cafe del jarocho es el mejor
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
cinthia
Escribe el mensaje que quieres dejar
ya termine de ver la serie breaking bad
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
fulano
Escribe el mensaje que quieres dejar
anita lava la tina
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
alumnito
```

```
Escribe el mensaje que quieres dejar
no podre ir al examen me enferme de varicela
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
caperucita roja
Escribe el mensaje que quieres dejar
ya le llevo los panes a mi abuelita
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
piolin
Escribe el mensaje que quieres dejar
parece que he visto a un lindo gatito
No es posible guardar mas mensajes AgendaLlenaException
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
4
Mensaje de  salvador -> espero todo marche bien
Mensaje de  amparo -> pasa en la tarde a mi oficina
Mensaje de  pepe pecas -> necesitamos tiempo para entregar la tarea 7
Mensaje de  gerardo -> nos vemos con canek afuera de la biblioteca a las 3 pm
Mensaje de  mickey mouse -> ya esta en produccion una nueva pelicula ja-ja
Mensaje de  cinthia -> el cafe del jarocho es el mejor
Mensaje de  cinthia -> ya termine de ver la serie breaking bad
Mensaje de  fulano -> anita lava la tina
Mensaje de  alumnito -> no podre ir al examen me enferme de varicela
Mensaje de  caperucita roja -> ya le llevo los panes a mi abuelita
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
2
Escribe el nombre del remitente
alumnito
Borrado exitoso
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
1
Escribe el nombre del remitente
piolin
Escribe el mensaje que quieres dejar
parece que he visto a un lindo gatito
El mensaje se ha guardado exitosamente
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
4
Mensaje de  salvador -> espero todo marche bien
Mensaje de  amparo -> pasa en la tarde a mi oficina
Mensaje de  pepe pecas -> necesitamos tiempo para entregar la tarea 7
Mensaje de  gerardo -> nos vemos con canek afuera de la biblioteca a las 3 pm
```

```
Mensaje de  mickey mouse -> ya esta en produccion una nueva pelicula ja-ja
Mensaje de  cinthia -> el cafe del jarocho es el mejor
Mensaje de  cinthia -> ya termine de ver la serie breaking bad
Mensaje de  fulano -> anita lava la tina
Mensaje de  piolin -> parece que he visto a un lindo gatito
Mensaje de  caperucita roja -> ya le llevo los panes a mi abuelita
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
3
Se borraron todos los mensajes
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
4
No hay mensajes que mostrar NoHayDatosException
1. Guarda Mensaje
2. Borra mensaje segun remitente
3. Borra todos los mensajes
4. Muestra todos los mensajes
5. Salir
5
Hasta Pronto
```

Apéndice J

Implementación del proyecto 10

Clase *Contacto.java*

```
/**
 * Clase Contacto
 * Esta clase modela un objeto de tipo contacto para almacenarse
 * en nuestra agenda
 *
 * @author Cinthia Rodriguez Maya
 * @version: Abril de 2014
 */

import java.io.Serializable;

public class Contacto implements Serializable {

   protected String nombre;
   protected String apellido;
   protected String celular;
   protected String email;
   protected Fecha cumpleanios;

   //Constructor
   public Contacto(String n, String ape, String c, String em, Fecha cumple
      ) {
      nombre=n;
      apellido=ape;
      celular=c;
      email=em;
      cumpleanios=cumple;
   }

   //Metodo toString
   public String toString() {
      return "Nombre " + nombre  + " Apellido " + apellido + "\n" + "
         Celular " + celular + "\n" + "e-mail " + email + "\n" + "Cumple "
          + cumpleanios;
   }
}
```

Clase *Fecha.java*

```
/**
 * Clase Fecha
 * modela un objeto de clase Fecha
 *
 * @author Cinthia Rodriguez Maya
 * @version: Abril de 2014
 */


import java.io.Serializable;

public class Fecha implements Serializable {

   private int dia;
   private int mes;
   private String nombre_mes="";

   //Constructor
   public Fecha(int d, int m) {
      dia=d;
      mes=m;
      daNombreMes(mes);
   }

   /**
    * Metodo daNombre
    * Dependiendo del numero de mes regresa el nombre de este
    * @param m, el numero de mes
    */
   public void daNombreMes(int m) {
      switch(m) {
         case 1:
            nombre_mes="Enero";
            break;
         case 2:
            nombre_mes="Febrero";
            break;
         case 3:
            nombre_mes="Marzo";
            break;
         case 4:
            nombre_mes="Abril";
            break;
         case 5:
            nombre_mes="Mayo";
            break;
         case 6:
            nombre_mes="Junio";
            break;
         case 7:
            nombre_mes="Julio";
            break;
         case 8:
            nombre_mes="Agosto";
```

```
            break;
         case 9:
            nombre_mes="Septiembre";
            break;
         case 10:
            nombre_mes="Octubre";
            break;
         case 11:
            nombre_mes="Noviembre";
            break;
         case 12:
            nombre_mes="Diciembre";
            break;
      }
   }

   //Metodo toString
   public String toString() {
      return dia + " de " + nombre_mes;
   }
}
```

Clase *Agenda.java*

```
/**
 * Clase Agenda
 * esta clase permite obtener informacion de la entrada estandar
 * para poder simular el comportamiento de una agenda
 *
 * @author Cinthia Rodriguez Maya
 * @version: Abril de 2014
 */


import java.io.*;
import java.util.Scanner;

public class Agenda {

   private String nombreArchivo="";
   Scanner entrada = new Scanner(System.in);
   private Contacto[] contactos;

   //Constructor
   public Agenda() {

   }

   //Constructor
   public Agenda(String nombreArchivo) throws RuntimeException{
      this.nombreArchivo=nombreArchivo;
      File archivoContactos= new File(nombreArchivo);
      if(archivoContactos.exists()) {
         if(!archivoContactos.canRead()) {
            throw new RuntimeException("No es posible leer el archivo "+
               nombreArchivo);
```

```
32            }
33            if(!archivoContactos.canWrite()){
34               throw new RuntimeException("No es posible escribir en el
                     archivo "+nombreArchivo);
35            }
36            contactos=leerContactos();
37         }else{
38            contactos=new Contacto[100];
39         }
40      }
41
42      /**
43       * Metodo menu
44       * despliega las posibles acciones a realizar con la agenda
45       */
46      public void menu() {
47         Agenda miAgenda=new Agenda("Contactos");
48         System.out.println("Bienvenido a tu agenda");
49         System.out.println("Selecciona una opcion");
50         int opcion=0;
51         do {
52            System.out.println("1. Agregar un contacto");
53            System.out.println("2. Consultar toda la agenda");
54            System.out.println("3. Buscar contacto");
55            System.out.println("4. Modificar contacto");
56            System.out.println("5. Borrar un contacto");
57            System.out.println("6. Borrar todos los contactos");
58            System.out.println("7. Salir");
59            opcion=entrada.nextInt();
60            switch(opcion) {
61               case 1:
62                  miAgenda.agregarContacto();
63                  miAgenda.grabar();
64                  break;
65               case 2:
66                  miAgenda.muestraTodo();
67                  break;
68               case 3:
69                  miAgenda.buscaContacto();
70                  break;
71               case 4:
72                  miAgenda.modificaContacto();
73                  break;
74               case 5:
75                  miAgenda.borraContacto();
76                  break;
77               case 6:
78                  miAgenda.borraTodo();
79                  break;
80               case 7:
81                  System.out.println("Hasta Pronto");
82                  break;
83               default:
84                  System.out.println("Selecciona una opcion valida");
85                  break;
86            }
```

```
87          }while(opcion!=7);
88          miAgenda.grabar();
89      }
90
91      /**
92       * Metodo agregarContacto
93       * agrega contactos a la agenda
94       */
95      public void agregarContacto() {
96          int contador=0;
97          for(int i=0; i<contactos.length; i++) {
98              if(contactos[i]==null) {
99                  System.out.println("Proporciona la siguiente informacion");
100                 System.out.println("Nombre");
101                 String n=entrada.next();
102                 System.out.println("Apellido");
103                 String a=entrada.next();
104                 System.out.println("Telefono");
105                 String t=entrada.next();
106                 System.out.println("Escribe el e-mail");
107                 String em=entrada.next();
108                 System.out.println("Fecha de cumplea os");
109                 System.out.println("Dia(1-31)");
110                 int d=entrada.nextInt();
111                 System.out.println("Mes(1-12)");
112                 int m=entrada.nextInt();
113                 Fecha f = new Fecha(d,m);
114                 Contacto c = new Contacto(n,a,t,em,f);
115                 contactos[i]=c;
116                 break;
117             } else {
118                 contador++;
119             }
120         }
121         if(contador==contactos.length) {
122             System.out.println("La agenda esta llena");
123         }
124     }
125
126     /**
127      * Metodo muestraTodo
128      * despliega la informacion de todos los contactos almacenados en la
           agenda
129      */
130     public void muestraTodo() {
131         int contador=0;
132         for(int i=0; i<contactos.length;i++) {
133             if(contactos[i]!=null) {
134                 System.out.println(contactos[i]);
135             } else {
136                 contador++;
137             }
138         }
139         if(contador==contactos.length) {
140             System.out.println("La agenda esta vacia");
141         }
```

```
142     }
143
144     /**
145      * Metodo buscaContacto
146      * busca un contacto dado solo su nombre
147      * muestra todos los que coincidan con el nombre
148      */
149     public void buscaContacto() {
150        System.out.println("Nombre de la persona que buscas: ");
151        String nombre_buscado=entrada.next();
152        int contador=0;
153        for(int i=0; i<contactos.length; i++) {
154           if(contactos[i]!=null) {
155              String nombre_actual=contactos[i].nombre;
156              if(nombre_actual.equals(nombre_buscado)) {
157                 System.out.println(contactos[i]);
158              } else {
159                 contador++;
160              }
161           } else {
162              contador++;
163           }
164        }
165        if(contador==contactos.length) {
166           System.out.println("Contacto no encontrado");
167        }
168     }
169
170     /**
171      * Metodo modificaContacto
172      * permite cambiar alguno de los datos de algun contacto
173      * almacenado en la agenda
174      */
175      public void modificaContacto() {
176         System.out.println("Proporciona los siguientes datos de la persona
               que quieres modificar");
177         System.out.println("Nombre: ");
178         String nombre_m=entrada.next();
179         System.out.println("Apellido");
180         String apellido_m=entrada.next();
181         int modificar=0;
182         int contador=0;
183         for(int i=0; i<contactos.length; i++){
184            if(contactos[i]!=null) {
185               if(contactos[i].nombre.equals(nombre_m) && contactos[i].
                     apellido.equals(apellido_m))  {
186                  System.out.println("  Qu   deseas modificar de este
                        contacto?");
187                  System.out.println("1. Nombre");
188                  System.out.println("2. Apellido");
189                  System.out.println("3. Tel fono");
190                  System.out.println("4. e-mail");
191                  System.out.println("5. Cumpleanios");
192                  modificar=entrada.nextInt();
193                  switch(modificar) {
194                     case 1:
```

```
195                         System.out.println("Escribe el nuevo nombre");
196                         String nombre_new = entrada.next();
197                         contactos[i].nombre=nombre_new;
198                         break;
199                      case 2:
200                         System.out.println("Escribe el nuevo apellido");
201                         String apellido_new = entrada.next();
202                         contactos[i].apellido=apellido_new;
203                         break;
204                      case 3:
205                         System.out.println("Escribe el nuevo telefono");
206                         String tel_new = entrada.next();
207                         contactos[i].celular=tel_new;
208                         break;
209                      case 4:
210                         System.out.println("Escribe el nuevo e-mail");
211                         String email_new = entrada.next();
212                         contactos[i].email=email_new;
213                         break;
214                      case 5:
215                         System.out.println("Modifica el cumpleanios");
216                         System.out.println("Dia");
217                         int d_new = entrada.nextInt();
218                         System.out.println("Mes");
219                         int m_new = entrada.nextInt();
220                         Fecha f_new = new Fecha(d_new, m_new);
221                         contactos[i].cumpleanios = f_new;
222                         break;
223                      default:
224                         System.out.println("Opcion incorrecta");
225                         break;
226                   }
227                }
228             } else {
229                contador++;
230             }
231          }
232          if(contador==contactos.length) {
233             System.out.println("Contacto no encontrado");
234          } else {
235             System.out.println("Modificacion exitosa");
236          }
237       }
238
239       /**
240        * Metodo borraContacto
241        * elimina de la agenda a un contacto
242        * dados el nombre y apellido
243        */
244       public void borraContacto() {
245          int contador=0;
246          System.out.println("Escribe el nombre del contacto que quieres
              borrar");
247          String n = entrada.next();
248          System.out.println("Escribe el apellido del contacto que quieres
              eliminar");
```

```
249         String ape= entrada.next();
250         for(int i=0; i<contactos.length; i++){
251            if(contactos[i]!=null) {
252               if(contactos[i].nombre.equals(n)&&contactos[i].apellido.equals
                     (ape)) {
253                  contactos[i]=null;
254               } else {
255                  contador++;
256               }
257            } else {
258               contador++;
259            }
260         }
261         if(contador==contactos.length) {
262            System.out.println("El contacto no existe");
263         } else {
264            System.out.println("Contacto borrado");
265         }
266      }
267
268      /**
269       * Metodo borraTodo
270       * elimina todos los contactos contenidos en la agenda
271       */
272      public void borraTodo() {
273         for(int i=0; i<contactos.length; i++) {
274            contactos[i]=null;
275         }
276         System.out.println("Se han borrado todos los contactos");
277      }
278
279      /**
280       * Metodo grabar
281       * envia a un archivo la informacion contenida en la agenda
282       */
283      public void grabar(){
284         try{
285            ObjectOutputStream agenda= new ObjectOutputStream(new
                  FileOutputStream(nombreArchivo));
286            agenda.writeObject(contactos);
287            agenda.close();
288         } catch(NotSerializableException e){
289            System.out.println("Error en la grabacion: "+e+"Objeto no
                  serializable");
290         } catch(IOException e){
291            System.out.println("Error en la grabacion: "+e);
292         }
293      }
294
295      /**
296       * Metodo leerContactos
297       * abre el archivo si este existe y carga la informacion
298       * a la agenda
299       */
300      public Contacto[] leerContactos(){
301         try{
```

```
302             ObjectInputStream lector=new ObjectInputStream(new
                   FileInputStream(nombreArchivo));
303             Contacto [] contactos=(Contacto []) lector.readObject();
304             lector.close();
305             this.contactos=contactos;
306          } catch(IOException e){
307             System.out.println("Lectura fallida: "+e);
308          } catch(ClassNotFoundException e){
309             System.out.println("Lectura fallida: "+e);
310          }
311          return contactos;
312       }
313
314    }
```

Clase *UsaAgenda.java*

```
1  /**
2   * Clase UsaAgenda
3   * clase que contiene el metodo main del proyecto de la agenda
4   *
5   * @author Cinthia Rodriguez Maya
6   * @version: Abril de 2014
7   */
8
9  public class UsaAgenda {
10
11    public static void main(String[] args) {
12       Agenda a = new Agenda();
13       a.menu();
14    }
15 }
```

Ejecución de nuestro proyecto

```
cinthia@cinthia-Inspiron-1011:~/Escritorio/Tesis/codigo/proyecto10\$ java UsaAgenda
Bienvenido a tu agenda
Selecciona una opcion
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
2
La agenda esta vacia
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
1
Proporciona la siguiente informacion
Nombre
cinthia
Apellido
rodriguez
Telefono
```

```
57555728
Escribe el e-mail
cinthia.rguez@gmail.com
Fecha de cumplea os
Dia(1-31)
13
Mes(1-12)
2
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
7
Hasta Pronto
cinthia@cinthia-Inspiron-1011:~/Escritorio/Tesis/codigo/proyecto10\$ java UsaAgenda
Bienvenido a tu agenda
Selecciona una opcion
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
2
Nombre cinthia Apellido rodriguez
Celular 57555728
e-mail cinthia.rguez@gmail.com
Cumple 13 de Febrero
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
1
Proporciona la siguiente informacion
Nombre
juan
Apellido
rodriguez
Telefono
57769592
Escribe el e-mail
juan@hotmail.com
Fecha de cumplea os
Dia(1-31)
27
Mes(1-12)
6
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
2
Nombre cinthia Apellido rodriguez
Celular 57555728
e-mail cinthia.rguez@gmail.com
Cumple 13 de Febrero
Nombre juan Apellido rodriguez
Celular 57769592
e-mail juan@hotmail.com
Cumple 27 de Junio
1. Agregar un contacto
```

```
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
3
Nombre de la persona que buscas:
cinthia
Nombre cinthia Apellido rodriguez
Celular 57555728
e-mail cinthia.rguez@gmail.com
Cumple 13 de Febrero
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
7
Hasta Pronto
cinthia@cinthia-Inspiron-1011:~/Escritorio/Tesis/codigo/proyecto10\$ java UsaAgenda
Bienvenido a tu agenda
Selecciona una opcion
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
4
Proporciona los siguientes datos de la persona que quieres modificar
Nombre:
juan
Apellido
rodriguez
  Qu   deseas modificar de este contacto?
1. Nombre
2. Apellido
3. Tel fono
4. e-mail
5. Cumpleanios
3
Escribe el nuevo telefono
57555728
Modificacion exitosa
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
7
Hasta Pronto
cinthia@cinthia-Inspiron-1011:~/Escritorio/Tesis/codigo/proyecto10\$ java UsaAgenda
Bienvenido a tu agenda
Selecciona una opcion
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
2
Nombre cinthia Apellido rodriguez
Celular 57555728
e-mail cinthia.rguez@gmail.com
```

```
Cumple 13 de Febrero
Nombre juan Apellido rodriguez
Celular 57555728
e-mail juan@hotmail.com
Cumple 27 de Junio
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
5
Escribe el nombre del contacto que quieres borrar
juan
Escribe el apellido del contacto que quieres eliminar
rodriguez
Contacto borrado
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
3
Nombre de la persona que buscas:
anita
Contacto no encontrado
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
7
Hasta Pronto
cinthia@cinthia-Inspiron-1011:~/Escritorio/Tesis/codigo/proyecto10\$ java UsaAgenda
Bienvenido a tu agenda
Selecciona una opcion
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
2
Nombre cinthia Apellido rodriguez
Celular 57555728
e-mail cinthia.rguez@gmail.com
Cumple 13 de Febrero
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
6
Se han borrado todos los contactos
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
2
La agenda esta vacia
```

```
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
7
Hasta Pronto
cinthia@cinthia-Inspiron-1011:~/Escritorio/Tesis/codigo/proyecto10\$ java UsaAgenda
Bienvenido a tu agenda
Selecciona una opcion
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
2
La agenda esta vacia
1. Agregar un contacto
2. Consultar toda la agenda
3. Buscar contacto
4. Modificar contacto
5. Borrar un contacto
6. Borrar todos los contactos
7. Salir
7
Hasta Pronto
```

Bibliografía

[ACM2011] "Operational definition of computational thinking for K-12 education", 2011.

[Alonso1997] Alonso, Catalina M., Domingo J. Gallego, and Peter Honey. "Los estilos de aprendizaje: procedimientos de diagnósticos y mejora", Bilbao: Mensajero, 1997.

[Barnes2007] Barnes, David J., and Michael Kölling. "Programación orientada a objetos con Java", MANEJO EFICIENTE DEL TIEMPO (2007).

[Barr2011] Barr Valerie, Chris Stephenson, "Bringing Computational Thinking to K12: What is Involved and Whats is the Role of the Computer Science Education Community?" ACM Inroads Vol. 2 No. 1, 2011.

[Bohm1966]Bohm, C. y G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules", Communications of the ACM, Vol. 9, No. 5, pag: 336-371, 1966.

[Cabral2012] Cabral Perdomo Ignacio, "Enseñando niños a programar", Software Gurú, No. 37, 2012.

[Carrascosa1985] Carrascosa, J. y Gil, D. (1985) "La metodología de la superficialidad y el aprendizaje de las ciencias", Enseñanza de las ciencias, 3, pag: 113-120, 1985.

[Cormen2009] Cormen, Thomas; Leiserson, Charles; Rivest, Ronald; Stein, Clifford (2009). Introduction to algorithms (3 edición). Cambridge, Massachusetts: The MIT Press.

[Deitel2004] Deitel, Harvey M., and Paul J. Deitel. "Cómo programar en Java", Pearson Educación, 2004.

[Espinosa2000] Espinosa, Sergio Dávila. "El aprendizaje significativo: Esa extraña expresión", Contexto educativo: revista digital de investigación y nuevas tecnologías No. 9, pag: 6, 2000.

[Feldman2005] Feldman, R.S. "Psicología: con aplicaciones en países de habla hispana", (Sexta Edición) México, McGrawHill, (2005).

[Fenstermacher2006] Fenstermacher, Gary. "Tres aspectos de la filosofía de la investigación sobre la

enseñanza", Wittrock, M.: La investigación de la enseñanza I. Barcelona: Paidós (1989).

[Gamma2006] Gamma, Erich, and Kent Beck. "JUnit" (2006).

[Gimeno1992] Gimeno Sacristán, José, and Antonio Pérez Gómez. "Comprender y transformar la enseñanza", Madrid: Morata (1992).

[Hetzel1988]Hetzel, William C., and Bill Hetzel, "The complete guide to software testing", Wellesley, MA: QED Information Sciences, 1988.

[Kernighan1999] Kernighan, Brian W., and Rob Pike. "The practice of programming", Addison-Wesley Professional, 1999.

[Linder1993] Linder, C. (1993) A challenge to conceptual change. Science Education, No. 77, pag: 293-300, 1993.

[Löic2013] Loïc Martínez Normand, Fernando Alonso Amo, "Paradigmas de Programación", Administración Digital (2013).

[López2007] López Gaona, Amparo. "Introduccion Al Desarrollo de Programas Con Java", segunda edición, Las prensas de Ciencias UNAM, 2011.

[Martínez2004] Martínez Valcárcel Nicolás, "Los modelos de enseñanza y la práctica de aula", Universidad de Murcia, España 2004

[Morales2004] Morales Bueno Patricia, Victoria Landa Fitzgerald, "APRENDIZAJE BASADO EN PROBLEMAS", vol. 13, Universidad de Bío-Bío, Chile, 2004.

[Myers2011]Myers, Glenford J., Corey Sandler, and Tom Badgett. "The art of software testing", John Wiley and Sons, 2011.

[Otero1990] Otero, J.C y Campanario J.M (1990) Variables cognitivas y metacognitivas en la comprensión de textos científicos: el papel de los esquemas y el control de la propia comprensión, Enseñanza de las ciencias, No. 8 , pag: 17-22, 1990.

[Peter1997] Peter Abel. IBM PC Assembly Language and Programming. Fourth Edition. Prentice Hall. 1997.

[Polya1973] Polya G. "How to solve it", segunda edición, Princeton University Press, New Jersey 1973

[Pozo1987] Pozo, Juan Ignacio, and Mario Carretero. "Del pensamiento formal a las concepciones espontáneas: ¿Qué cambia en la enseñanza de la ciencia?", Infancia y aprendizaje Vol 10, No. 38, pag: 35-52, 1987.

[Pozo1992] Pozo, Juan Ignacio, et al. "Las ideas de los alumnos sobre la ciencia como teorías implíci-

tas", Infancia y aprendizaje Vol. 15, No. 57, pag: 3-21, 1992.

[Pozo1996] Pozo Municio, Ignacio. "Aprendices y maestros", Madrid: Alianza (1996).

[Rojas2001] Velásquez, Freddy Rojas. "Enfoques sobre el aprendizaje humano" PDF), 2001
$http://ares.unimet.edu.ve/programacion/psfase3/modII/biblio$

[Skiena2006] Skiena S. Steven, Miguel A. Revilla, "Concursos Internacionales de Informática y Programación: manual de entrenamiento por Internet", Universidad de Valladolid, Secretariado de Publicaciones e Intercambio de Editorial, 2006.

[Sprinthall1996] Sprinthall, N. Sprinthall. R. y Oja, S. "Psicología de la Educación", Madrid, (1996)

[UNESCO2005] UNESCO, Oficina de información pública, Enseñanza de las ciencias y la tecnología, México; 2005

[Valle1993] Valle Arias, A., et al. "Apprendizaje significativo y enfoques de aprendizaje: el papel del alumno en el proceso de construcción de conocimientos", Revista de Ciencias de la Educación 156, pag: 481-502, 1993.

[Villalobos2009] Villalobos, J. Proyecto CUPI2–Una solución integral al problema de enseñar y aprender a programar, 10º Premio Colombiano en Investigación Educativa (Departamento de Ingeniería de Sistemas y Computación–Facultad de Ingeniería–Universidad de Los Andes), 1–39, 2009.