TAD Lista

Práctica 4
Listas Ordenadas

Objetivo

El objetivo de esta práctica es que el alumno programe una especialización de la clase Lista para crear listas ordenadas. A su vez continuará reforzando su conocimiento sobre Tipos Abstractos de Datos, Herencia de clases y uso de comparadores.

Descripción general

Se programará la clase especializada `ListaOrdenada` que será una subclase de `Lista`. Se busca que los elementos de la lista se encuentren ordenados, para ello se requiere que sus elementos implementen la interfaz Comparator. La clase especializada deberá contar con un constructor que reciba un comparador. Finalmente, se incluirá el método `fundir(Lista lis)` para intercalar los elementos de la lista ordenada que llama al método, con los elementos de la lista `lis`.

Material

El material de esta práctica consta de los siguientes archivos:

- **InterfazLista.class** interfaz para el tipo abstracto de datos Lista.
- **Nodo.class** clase que implementa los nodos que sirven para crear una lista.
- **Lista.class** clase con una posible implementación de la interfaz `InterfazLista`.
- **Lista$Milterador.class** clase interna de la clase Lista que implementa un iterador de la lista.
- **Tarea.class** clase para trabajar con tareas. Se usará para crear una lista de tareas.
- **ComparaPrioridades.class** clase que crea un comparador de tareas de acuerdo a sus prioridades. Se usará para crear una lista ordenada de tareas.
- **PruebaListaOrdenada.class** programa para probar la clase `ListaOrdenada`.
- Documentación:
    - InterfazLista.html documentación de la interfaz `InterfazLista`.
    - Nodo.html documentación de la clase `Nodo`.
    - Lista.html documentación de la clase `Lista`.
    - Tarea.html documentación de la clase `Tarea`.
    - ComparaPrioridades.html documentación de la clase `ComparaPrioridades`.

Desarrollo

1. Descargar los archivos `InterfazLista.class, Nodo.class, Lista.class, Lista$MiIterador.class, Tarea.class, ComparaPrioridades.class y PruebaListaOrdenada.class` en el directorio donde se va a desarrollar la práctica.

2. Leer la documentación y revisar el material de la práctica. Recordar que la firma de los métodos proporcionados y de los requeridos en el desarrollo de la práctica no podrán ser modificados.

3. Se debe crear la clase `ListaOrdenada` que implementará la especialización de la clase `Lista` para crear una lista cuyos elementos estén ordenados. Entre los métodos por programar están:

    a. **Constructor** de la clase que reciba como parámetro un Comparador.
    b. Sobrescribir el método **agregar** para insertar de manera ordenada un objeto dentro de la lista.
    c. Método **fundir** que reciba como parámetro un objeto de tipo Lista. Este método integrará los elementos de la lista pasada como parámetro, a la lista que llama al método, asegurándose de mantener el orden de la lista.
    d. Sobrescribir los métodos **agregarAlInicio** y **sustituir** para imprimir un mensaje indicando que dichos métodos no aplican para la lista ordenada.

4. Ejecutar el programa `PruebaListaOrdenada` para probar la implementación realizada. Si el programa funciona adecuadamente se verán los siguientes mensajes:

```
Tareas pendientes:
- Lavar ropa, prioridad: 5
- Hacer practica de ED, prioridad: 1
- Cita con el dentista, prioridad: 4
- Hacer ejercicio, prioridad: 2

Tareas ordenadas por prioridad:
- Hacer practica de ED, prioridad: 1
- Hacer ejercicio, prioridad: 2
- Cita con el dentista, prioridad: 4
- Lavar ropa, prioridad: 5

Otra lista de tareas pendientes:
- Alimentar a Fido, prioridad: 1
- Ver serie de TV, prioridad: 6
- Comprar despensa, prioridad: 3
```

```
Fusionar el resto de tareas pendientes a mi lista ordenada de
tareas...
- Hacer practica de ED, prioridad: 1
- Alimentar a Fido, prioridad: 1
- Hacer ejercicio, prioridad: 2
- Comprar despensa, prioridad: 3
- Cita con el dentista, prioridad: 4
- Lavar ropa, prioridad: 5
- Ver serie de TV, prioridad: 6

La lista incluye la tarea "Limpiar mi cuarto" ?... false

La lista incluye la tarea "lavar ropa" ?... true

Eliminar la tarea "lavar ropa" ...

Lista final de tareas ordenadas por prioridad:
- Hacer practica de ED, prioridad: 1
- Alimentar a Fido, prioridad: 1
- Hacer ejercicio, prioridad: 2
- Comprar despensa, prioridad: 3
- Cita con el dentista, prioridad: 4
- Ver serie de TV, prioridad: 6

Llamada al metodo agregarAlInicio...
El metodo agregarAlInicio no aplica para la Lista Ordenada.

Llamada al metodo sustituir...
El metodo sustituir no aplica para la Lista Ordenada.
```

5. Escribir una tabla con la complejidad de los métodos programados en la clase `ListaOrdenada`.

Operación	**Tiempo de ejecución**
constructor	
agregar	
fundir	