ENUNCIADO DEL EXAMEN – PRÁCTICAS LTP – 2/11/2021

Resuelve los 4 ejercicios de este enunciado de examen en el proyecto BlueJ facilitado.

IMPORTANTE. Se valorará, cuando sean aplicables, el uso adecuado de los mecanismos de **herencia**, **sobrecarga**, **polimorfismo** o **genericidad**.

Contenido del proyecto BlueJ.

Se facilita el siguiente conjunto de clases e interfaces:

- **Coleccion<T>**: Interfaz que modela las operaciones sobre una colección de elementos.
- **Vector<T>**: Interfaz que extiende a **Coleccion<T>**, añadiendo funcionalidades para acceder arbitrariamente a los elementos de una colección.
- **VectorList<T>**: Clase que implementa la interfaz **Vector**.
- **Figure**. Clase que guarda información sobre figuras geométricas (concretamente, su posición).
- **ColeccionUse**: Clase con un método **main** para probar el resto de clases. Esta clase NO se debe modificar. Esta clase compilará cuando se resuelvan los ejercicios.

EJERCICIO 1 (1,75 puntos).

Crea una nueva interfaz, llamada **VectorOrd<T>**, que extienda a **Coleccion<T>** definiendo dos nuevas operaciones:

T maximo()

T minimo()

La funcionalidad de estos métodos:

- **maximo()** es un método consultor que obtiene el elemento mayor del vector.
- **minimo()** es un método consultor que obtiene el elemento menor del vector.

EJERCICIO 2 (2,75 puntos).

Crea una nueva clase, llamada **VectorOrdList<T>**, que implemente la interfaz **VectorOrd<T> c**on las siguientes características:

- Extenderá la clase **VectorList<T>**.
- No dispondrá de atributos (solamente los recibidos por herencia).
- Los elementos de este vector estarán siempre ordenados. El menor será el primer elemento de **laLista** (atributo heredado de **VectorList**) y el mayor el último elemento del atributo **laLista**.

EJERCICIO 3 (2,75 puntos).

Sobreescribe en la clase **VectorOrdList<T>** el método:

void add(T e)

De modo que se añada el elemento **e** en la posición adecuada para que los elementos del vector ordenado mantengan su orden.

***Nota*:** Se debe usar **compareTo** para comparar los elementos, por lo que debes asegurarte de que los elementos dispongan de dicho método. Puedes restringir la parametricidad.

EJERCICIO 4 (2,75 puntos).

Crea una nueva clase, llamada **FigurasOrdenadas<T>**, que extienda la clase **VectorOrdList<T>**, pero con restricción de la genericidad a la clase **Figure**. **FigurasOrdenadas** no dispondrá de atributos (solamente los recibidos por herencia).

Añade, e implementa, en **FigurasOrdenadas** el siguiente método:

double getXmax()

Este método devolverá el valor del atributo **x** de la figura que esté más alejada del origen del eje bidimensional (punto (0,0)).

***Nota*:** La clase **Figure** implementa la interfaz **Comparable** estableciendo como orden natural la distancia de la figura al origen.

TEST DE LOS EJERCICIOS.

Si se resuelven correctamente los ejercicios, la clase **ColeccionUse** compilará y la ejecución de su método **main** generará la siguiente salida:

```
VectorList vL : [8, 7, 6, 5, 4, 3, 2, 1]
VectorOrdList vOL : [13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 99.0]
máximo : 99.0
mínimo : 13.0
FiguresOrdenadas fO : [Position: (1.0, 1.0), Position: (2.0, 2.0), Position:
(3.0, 3.0), Position: (4.0, 4.0), Position: (5.0, 5.0), Position: (6.0, 6.0),
Position: (7.0, 7.0), Position: (8.0, 8.0)]
x del elemento más alejado del eje de fO: 8.0
```

Obtener esta salida es indicio, pero no garantía, de la resolución correcta de los ejercicios.
