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 aleatoriamente 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 **Pila<T>**, que extienda a **Coleccion<T>** definiendo dos nuevas operaciones:

T cima()

T desapilar()

La funcionalidad de estos métodos:

- **cima()** es un método consultor que devuelve el valor de la cima de la pila. La cima es el último elemento insertado en la pila.
- **desapilar()** es un método modificador que borra la cima de la pila y devuelve su valor.

EJERCICIO 2 (2,75 puntos).

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

- Extenderá la clase **VectorList<T>**.
- No dispondrá de atributos (solamente los recibidos por herencia).
- La **cima** de la pila será último elemento de **laLista** (atributo heredado de **VectorList**).

EJERCICIO 3 (2,75 puntos).

Añade a la interfaz **Coleccion<T>** un nuevo método**:**

boolean pertenece(T e)

Este método determina si un elemento pertenece a una colección.

Implementa este método una sola vez, en el lugar adecuado para que esté disponible para las clases **VectorList** y **PilaList**.

EJERCICIO 4 (2,75 puntos).

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

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

double getXcima()

Este método devolverá el valor del atributo **x** de la figura que esté en la cima de la pila de figuras.

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]
Pertenecen al Vector? (8 : true) (99 : false)
PilaList pL : [20.0, 19.0, 18.0, 17.0, 16.0, 15.0, 99.0]
Pertenecen a la Pila? (16.0 : true) (99.0 : true)
cima : 99.0
desapilar : 99.0
cima : 15.0
PilaList pL : [20.0, 19.0, 18.0, 17.0, 16.0, 15.0]
Pertenecen a la Pila? (16.0 : true) (99.0 : false)
FiguresPila pF : [Position: (4.0, 6.5), Position: (3.5, 8.2)]
x de la cima de pF: 3.5
```

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