UNIDAD 8: Interfaces - Calificables

Prof. Gotzon Valcarcel Jiménez

Reto 01:

Crea una interfaz llamada ColeccionInterfaz que contenga los siguientes métodos:

1. **estaVacia()**: Devuelve true si la colección está vacía y false en caso contrario.
2. **extraer()**: Devuelve y elimina el primer elemento de la colección.
3. **primero()**: Devuelve el primer elemento de la colección sin eliminarlo.
4. **añadir(Object obj)**: Añade un objeto por el extremo correspondiente y devuelve true si se ha añadido correctamente o false en caso contrario.

A continuación, implementa una clase llamada Pila que implemente esta interfaz. La clase debe funcionar como una pila (estructura LIFO) utilizando:

- Un array de tipo Object para almacenar los elementos.
- Un entero que sirva como contador de objetos.

Detalles de implementación

1. En la interfaz ColeccionInterfaz, los métodos deben declararse sin implementación.
2. En la clase Pila:
    - Los atributos deben incluir un array de Object y un entero que represente el número de elementos actuales en la pila.
    - El constructor debe recibir como parámetro el tamaño máximo de la pila.
    - El método estaVacia() debe comprobar si el contador es igual a 0.

- El método añadir(Object obj) debe comprobar si hay espacio disponible en el array. Si es así, añade el objeto en la posición correspondiente y aumenta el contador.
- El método primero() debe lanzar una excepción NoSuchElementException si la pila está vacía. De lo contrario, devuelve el último elemento añadido sin eliminarlo.
- El método extraer() debe lanzar una excepción NoSuchElementException si la pila está vacía. De lo contrario, reduce el contador y devuelve el elemento correspondiente.
- Sobrescribe el método toString() para mostrar los elementos de la pila.

Reto 02:

- Escribe una clase llamada **PruebaPila** que implemente los siguientes métodos:
- **rellenar(ColeccionInterfaz coleccion)**: Recibe como parámetro un objeto que implementa la interfaz ColeccionInterfaz y añade los números del 1 al 10 a la colección.
- **imprimirYVaciar(ColeccionInterfaz coleccion)**: Recibe como parámetro un objeto que implementa la interfaz ColeccionInterfaz, extrae cada elemento de la colección y lo imprime hasta que esta quede vacía.
- Finalmente, crea un objeto de tipo Pila, utiliza los métodos anteriores y verifica su funcionamiento.

Reto 03:

Escribe una clase llamada **Cola** que implemente la interfaz ColeccionInterfaz. La clase debe utilizar un objeto de la clase LinkedList como base para gestionar los elementos de la cola.

Detalles de implementación

1. **Relación de composición**:
    - La clase Cola debe tener un atributo privado de tipo LinkedList que se encargará de almacenar los elementos de la cola.
2. **Reglas de operación**:
    - Los elementos se añaden al final de la cola.
    - Los elementos se extraen del principio de la cola.
3. **Métodos a implementar (según ColeccionInterfaz)**:
    - **estaVacia()**: Devuelve true si la cola está vacía, de lo contrario, devuelve false.
    - **extraer()**: Elimina y devuelve el primer elemento de la cola. Si está vacía, lanza una excepción NoSuchElementException.
    - **primero()**: Devuelve el primer elemento de la cola sin eliminarlo. Si está vacía, lanza una excepción NoSuchElementException.
    - **añadir(Object obj)**: Añade un elemento al final de la cola y devuelve true.

Reto 04:

Desarrolla un programa para una biblioteca que maneje libros y revistas. Ambos comparten características comunes y tienen particularidades específicas:

1. **Características comunes (aplicables a libros y revistas)**:
    - **Código**, **título** y **año de publicación**.
    - Estas características se inicializan al momento de crear el objeto.
    - Deben contar con:
        - Un método toString() que devuelve los valores de sus atributos como una cadena de texto.
        - Métodos para obtener el **código** (getCodigo()) y el **año de publicación** (getAnio()).
2. **Características específicas**:
    - **Libros**:
        - Tienen un atributo prestado que indica si el libro está prestado (inicialmente false).
        - Implementan la interfaz **Prestable**, que incluye:
            - prestar(): Cambia el estado a prestado.
            - devolver(): Cambia el estado a no prestado.
            - prestado(): Devuelve true si está prestado, false en caso contrario.
    - **Revistas**:
        - Tienen un atributo adicional numero, que se inicializa al momento de crear la revista.

3. **Diseño del programa**:
    - Crear una superclase **Publicacion** con los atributos comunes.
    - Crear las clases **Libro** y **Revista**, que heredan de Publicacion y añaden sus atributos y comportamientos específicos.
    - La interfaz **Prestable** define los métodos relacionados con la gestión de préstamos.

Reto 05:

Desarrolla una aplicación que implemente los siguientes métodos:

1. **cuentaPrestados(Object[] objetos)**:
    - Recibe un array de objetos.
    - Devuelve cuántos de esos objetos están prestados.
    - Solo cuenta aquellos que implementen la interfaz Prestable.
2. **publicacionesAnterioresA(Publicacion[] publicaciones, int anio)**:
    - Recibe un array de publicaciones y un año.
    - Devuelve cuántas publicaciones tienen una fecha anterior al año proporcionado.

En el método **main()**:

- Crea un array de Publicacion con:
    - 2 libros y 2 revistas.
    - Presta uno de los libros.
- Muestra por pantalla:
    - Los datos almacenados en el array.
    - La cantidad de publicaciones prestadas.

- La cantidad de publicaciones con fecha anterior a 1990.