Práctica 6
Pila en arreglo

Estructuras de datos

1. Meta

Que el alumno domine el manejo de información almacenada en una **Pila**.

2. Objetivos

Al finalizar la práctica el alumno será capaz de:

- Implementar el tipo de dato abstracto **Pila** utilizando un arreglo.

3. Antecedentes

Una `Pila` es una estructura de datos caracterizada por:

1. El último elemento que entra a la `Pila` es el primer elemento que sale.
2. Tiene un tamaño dinámico.

A continuación se define el tipo de dato abstracto `Pila`.

Definición 1: Pila

Una **Pila** es una estructura de datos tal que:

1. Tiene un número variable de elementos de tipo T.
2. Sólo se tiene un apuntador al tope de la `Pila`.
3. Cuando se agrega un elemento, este se coloca en el tope de la `Pila`.

Nombre: $Vector$.
Valores: $\mathbb{N}, T$, con $null \in T$.
Constructores :
Pila(): $\emptyset \rightarrow Pila$
Precondiciones: $\emptyset$
Postcondiciones :

- Una $Pila$ vacía.

Métodos de acceso :

mira(this) $\rightarrow$ **e:** $Pila \times \mathbb{N} \rightarrow T$

Precondiciones: $\emptyset$

Postcondiciones :

- $e \in T$, e es el elemento almacenado en el tope de la $Pila$.

Métodos de manipulación :

expulsa(this) $\rightarrow$ **e:** $Pila \rightarrow T$

Precondiciones :$\emptyset$

Postcondiciones :

- Elimina y devuelve el elemento e que se encuentra en el tope de la $Pila$.

empuja(this, e): $Pila \times T \xrightarrow{?} \emptyset$

Precondiciones: $\emptyset$

Postcondiciones :

- El elemento e es asignado al tope de la $Pila$.

A continuación presentamos la interfaz `Pila`

```java
/*
 * Codigo utilizado para el curso de Estructuras de Datos.
 * Se permite consultarlo para fines didacticos en forma personal,
 * pero no esta permitido transferirlo tal cual a estudiantes actuales o potenciales.
 */
package estructuras.lineales;

import java.util.Collection;
import java.util.NoSuchElementException;

public interface Pila<E> extends Collection<E> {

    /**
     * Muestra el elemento al tope de la pila.
     * @return Una referencia al elemento siguiente.
     * @throws NoSuchElementException si la pila esta vacia
     */
    public E mira() throws NoSuchElementException;

    /**
     * Devuelve el elemento al tope de la pila y lo elimina.
     * @return Una referencia al elemento siguiente.
     * @throws NoSuchElementException si la pila esta vacia.
     */
    public E expulsa() throws NoSuchElementException;

    /**
     * Agrega un elemento al tope de la pila.
     * @param e Referencia al elemento a agregar.
     */
    public void empuja(E e);

}
```

Actividad 1

Revisa la documentación de la clase `Stack` de `Java`. ¿Cuáles serían los métodos equivalentes a los definidos aquí? ¿En qué difieren?

---

4. Desarrollo

Para implementar el TDA $Pila$ se harán dos implementaciones, con apuntadores y con arreglos dinámicos. En esta práctica se realizará la implementación con arreglos. Para esto se utilizará la clase abstracta `ColeccionAbstracta` de la práctica anterior, que implementa algunos métodos de la interfaz `Collection`.

Además de extender la clase abstracta, se debe implementar la interfaz Pila. Esto se deberá hacer en una clase `PilaArreglo`. Por razones didácticas, no se permite el uso de ninguna clase que se encuentre en el API de Java, por ejemplo clases como `Vector`, `ArrayList` o cualquiera que haga el manejo de arreglos dinámicos.

5. Preguntas

1. Explica, de acuerdo a cada una de tus implementaciones, cómo funciona el método `empuja`.
2. ¿Cuál es la complejidad, en el peor caso, de los métodos `mira`, `expulsa` y `empuja`?

6. Forma de entrega

- Asegúrense de borrar todos los archivos generados, incluyendo archivos de respaldo usando `ant clean`.
- Copien el directorio `Practica6` dentro de un directorio llamado `<apellido1_apellido2>` donde `apellido1` es el primer apellido de un miembro del equipo y `apellido2` es el primer apellido del otro miembro. Por ejemplo: `marquez_vazquez`.
- Borren el directorio libs en la copia.
- Agreguen un archivo `reporte.pdf` dentro del directorio `Practica6` de la copia, con el nombre completo de los integrantes del equipo, las repuestas a las preguntas de la sección anterior y cualquier comentario que quieran hacer sobre la práctica.
- Compriman el directorio `<apellido1_apellido2>` creando `<apellido1_apellido2>.tar.gz`. Por ejemplo: `marquez_vazquez.tar.gz`.
- Suban este archivo en la sección correspondiente del aula virtual. Basta una entrega por equipo.

7. Forma de evaluación

Para su calificación final se tomarán en cuenta los aspectos siguientes:

70% Calificación generada por las pruebas automáticas. Usaremos nuestros archivos, por lo que si realizan modificaciones a sus copias, éstas no tendrán efecto al momento de calificar.

15% Documentación completa y adecuada. Entrega en el formato requerido, sin archivos `.class`, respaldos, bibliotecas de `JUnit` u otros no requeridos.

15% Revisión manual del código, para verificar que se cumpla con las especificaciones, que no se haya copiado, etcétera.