Facultad de Ciencias, UNAM
Práctica 3

Alejandro Hernández Mora Pablo Camacho González
Luis Manuel Martínez Dámaso José Ricardo Desales Santos Emiliano Galeana Araujo

Fecha de entrega: 20 de agosto

Introducción.

Para modelar la EDD **Pila** y la EDD **Cola** usaremos a la clase `Lista` que implementa una **Lista Doblemente Ligada**.

Interfaz Apilable.

El comportamiento de la EDD **Pila** debe ser el siguiente:

- Agregar un elemento al tope de la **Pila**.
- Eliminar el tope de la **Pila**.
- Ver el elemento que está en el tope de la **Pila**.

Por lo tanto, la interfaz `Apilable` debe tener los siguientes métodos:

- push(T elemento). Método para agregar al tope de la pila. El tope de la pila será la cabeza de la lista que estamos extendiendo.
- pop(). Método para eliminar el elemento que está en el tope de la pila, además de devolver dicho elemento.
- top(). Método para ver el elemento que se encuentra en el tope de la Pila

Entonces la interfaz debe quedar así:

```
public interface Apilable<T> {

    public void push(T elemento) throws IllegalArgumentException;

    public T pop() throws NoSuchElementException;

    public T top() throws NoSuchElementException;

}
```

Interfaz Encolable.

El comportamiento de la EDD **Cola** debe ser el siguiente:

- Agregar un elemento al final de la **Cola**.
- Eliminar el primer elemento de la **Cola**.
- Ver el primer elemento de la **Cola**.

Para simular el comportamiento con la lsita que heredamos, agregaremos siempre al final de la lista y eliminaremos siempre al inicio. Considerando lo anterior la interfaz `Encolable` debe tener los siguientes métodos:

- queue(T elemento). Método para agregar al final de la cola.
- dequeue(). Método para eliminar el elemento que está en al principio de la cola, además de devolver dicho elemento.
- peek(). Método para ver el elemento que se encuentra al principio de la cola

Entonces la interfaz debe quedar así:

```
public interface Encolable<T> {

    public void queue(T elemento) throws IllegalArgumentException;

    public T dequeue() throws NoSuchElementException;

    public T peek() throws NoSuchElementException;

}
```

Clase Pila y Cola.

Para la implementación de la EDD **Pila** y **Pila**, debemos tener en cuenta que hay métodos de la clase `Lista` que chocan con el comportamiento que debería tener las clases anteriores.
Los métodos que vamos a prohibir el acceso al usuario son los siguientes:

- Métodos para agregar elementos.
- Métodos para eliminar elementos.

Prohibiremos los métodos anteriores para que el usuario use los métodos que definimos en la interfaz `Apilable` o `Encolable`. Para prohibir dichos métodos, lo que haremos será sobreescribirlos y lanzar una excepción de tipo `UnsupportedOperationException`.

Considerando lo anterior, la clase `Pila` queda de la siguiente forma:

```
public class Pila<T> extends Lista<T> implements Apilable<T> {
    public Pila() {
        /*Aqui no hay que hacer nada.*/
    }

    public Pila(Iterable<T> iterable) {
        /*Aqui va tu codigo*/
    }

    public Pila(Pila<T> p) {
        /*Aqui va tu codigo*/
    }

    @Override
    public void push(T elemento) throws IllegalArgumentException{
        /*Aqui va tu codigo*/
    }

    @Override
    public T pop() throws NoSuchElementException{
        /*Aqui va tu codigo*/
    }
```

```
    @Override
    public T top() throws NoSuchElementException{
        /*Aqui va tu codigo*/
    }

    @Override
    public void agregar(T elemento){
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para agregar elementos a una pila usa el metodo push(elemento)");
    }

    @Override
    public void agregarAlFinal(T elemento) {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para agregar elementos a una pila usa el metodo push(elemento)");
    }

    @Override
    public T getPrimero() {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }

    @Override
    public T getUltimo() {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }

    @Override
    public void eliminar(T elemento) {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }

    @Override
    public void eliminarPrimero() {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }

    @Override
    public void eliminarUltimo() {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }
}
```

La clase `Cola` queda así:

```
public class Cola<T> extends Lista<T> implements Encolable<T> {

    public Cola() {
        /*Aqui no hay que hacer nada.*/
    }

    public Cola(Iterable<T> iterable) {
        /*Aqui va tu codigo*/
    }

    public Cola(Cola<T> c) {
        /*Aqui va tu codigo*/
    }

    @Override
    public void queue(T elemento) throws IllegalArgumentException{
        /*Aqui va tu codigo*/
    }

    @Override
    public T dequeue() throws NoSuchElementException{
        /*Aqui va tu codigo*/
    }

    @Override
    public T peek() throws NoSuchElementException{
        /*Aqui va tu codigo*/
    }

    @Override
    public void agregar(T elemento){
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para agregar elementos a una pila usa el metodo push(elemento)");
    }

    @Override
    public void agregarAlFinal(T elemento) {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para agregar elementos a una pila usa el metodo push(elemento)");
    }

    @Override
    public T getPrimero() {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }

    @Override
    public T getUltimo() {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }

    @Override
    public void eliminar(T elemento) {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }

    @Override
```

```
    public void eliminarPrimero() {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }

    @Override
    public void eliminarUltimo() {
        throw new UnsupportedOperationException("No se puede hacer esta operacion.
        Para ver el tope de la pila usa el metodo top()");
    }
}
```