UNIVERSIDAD TECMILENIO

Actividad 2

Estructuras de datos

Adalberto Emmanuel
Rojas Perea

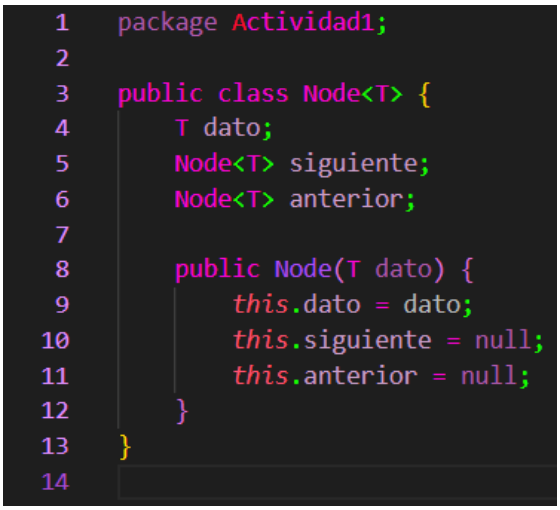
Leonardo Pereda Chánez
AL03087353

Documentación del código

Node.java

La clase **Node** toma como atributos el dato que va a contener **<T>**, en el que el tipo de dato es variable, y los punteros que apuntan hacia atrás o delante del nodo.

También cuenta con el constructor del Nodo, en el que se le asigna el dato que se le vaya a dar, y los punteros por defecto apuntan hacia **null**.

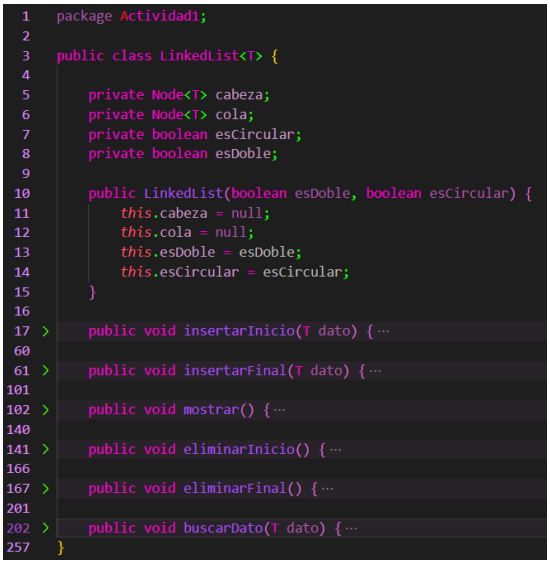


```java
package Actividad1;

public class Node<T> {
    T dato;
    Node<T> siguiente;
    Node<T> anterior;

    public Node(T dato) {
        this.dato = dato;
        this.siguiente = null;
        this.anterior = null;
    }
}

```

LinkedList.java

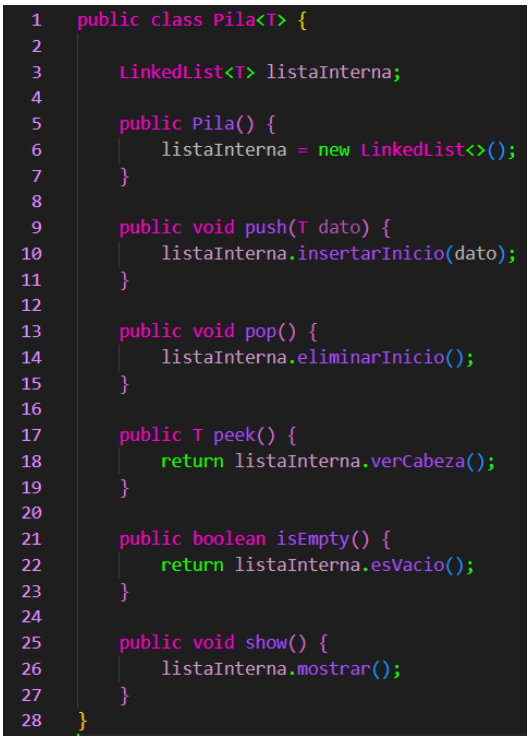
La clase **LinkedList** toma como atributos los Nodos cabeza y cola. Cuenta con el constructor en que el que asigna por defecto los Nodos cabeza y cola como **null.**

La clase cuenta con todos los métodos que manejan las operaciones de las listas, desde insertar, eliminar y mostrar.



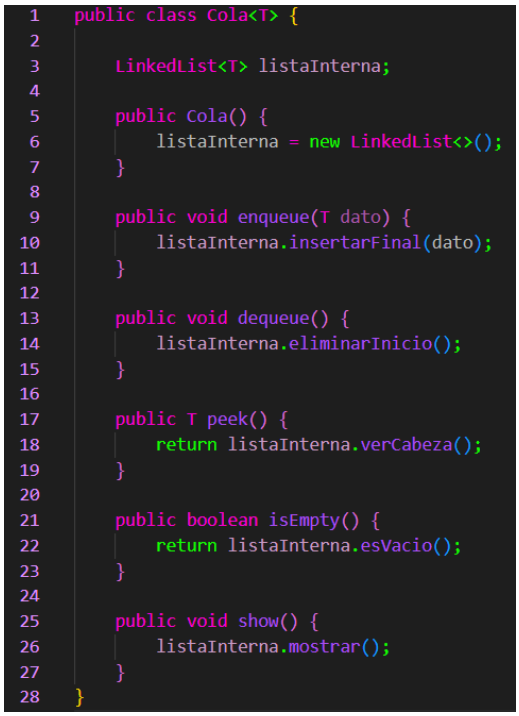
Pila.java

La clase **Pila** llama a varios métodos establecidos en **LinkedList**, y se les nombre según las operaciones relacionadas a las estructura de datos **Pila**, como lo es **push**, **pop**, **peek** y **show**.



Cola.java

La clase **Cola** llama a varios métodos establecidos en **LinkedList**, justo como en la clase **Pila**, con métodos como **enqueue**, **dequeue**, **peek** y **show**.



```java
public class Cola<T> {

    LinkedList<T> listaInterna;

    public Cola() {
        listaInterna = new LinkedList<>();
    }

    public void enqueue(T dato) {
        listaInterna.insertarFinal(dato);
    }

    public void dequeue() {
        listaInterna.eliminarInicio();
    }

    public T peek() {
        return listaInterna.verCabeza();
    }

    public boolean isEmpty() {
        return listaInterna.esVacio();
    }

    public void show() {
        listaInterna.mostrar();
    }
}
```

Main.java

La clase Main está realizada con el fin de que el usuario pueda interactuar de manera simulada con operaciones de un sistema operativo, donde puede ejecutar comandos o abrir programas.

La estructura del código está compuesta principalmente por **do while**, **switch**, **while** y **try_catch**, con el fin de que el programa sea resiliente ante diferentes errores de usuario y que este no se corrompa y termine el proceso de manera abrupta; al igual que el programa muestre los menús de manera organizada.

```java
import java.util.Scanner;
import java.util.InputMismatchException;

public class Main {
    Run | Debug
    public static void main(String[] args) {

            Pila<String> pila = new Pila<>();
            Cola<String> cola = new Cola<>();

            Scanner scanner = new Scanner(System.in);
            int opcion = 0;
        do {
            System.out.println(x:"*** ¡¡¡Bienvenido a tu sistema operativo!!! ***");
            System.out.println(x:"\n¿Qué deseas realizar hoy?");
            System.out.println(x:"\n1- Gestionar comandos\n2- Gestionar programas\n3- Salir del programa");

            try {
                System.out.print(s:"\nIngresa una opción: ");
                opcion = scanner.nextInt();
            } catch (InputMismatchException e) {
                System.out.println(x:"Error: Elige una opción correcta.");
                scanner.nextLine();
            }

            switch (opcion) {
                case 1: // Pila
                    int tareaPila = 0;

                    do {
                        System.out.println(x:"\n*** ¡¡¡Bienvenido al gestor de comandos!!! ***");
                        System.out.println(x:"\n¿Qué tarea deseas realizar?");
                        System.out.println(x:"\n1- Ejecutar un comando\n2- Eliminar el último comando\n3- Mostrar último comando\n4- Mostrar historial de comandos\n5-Regr

                        try {
                            System.out.print(s:"\nIngresa una opción: ");
                            tareaPila = scanner.nextInt();
```

```java
switch (tareaPila) {
    case 1:
        try {
            System.out.println(x:"\nEscribe el nombre de un comando. (Ej. lsblk, cd, mkdir)");
            System.out.print(s:"\n[java@datastructures ~]$ ");
            String comando = scanner.next();
            pila.push(comando);
            System.out.println(x:"\nEjecutando el comando...");
        } catch (InputMismatchException e) {
            System.out.println(x:"Error. Escriba un comando válido.");
            scanner.nextLine();
        }

        break;

    case 2:
        if (verificarListaPila(pila)) {
            String comando = pila.peek();
            System.out.println("\nEliminando el comando " + comando + "...");
            pila.pop();
        }

        break;

    case 3:
        if (verificarListaPila(pila)) {
            System.out.println("\n" + pila.peek());
        }

        break;

    case 4:
        if (verificarListaPila(pila)) {
            pila.show();
```

Evidencias de funcionamiento

```
=== PRUEBA DE FUNCIONES PILA ===
- Comando 1 insertado PUSH
- Comando 2 insertado PUSH
- Comando 3 insertado PUSH
- Comando 4 insertado PUSH

Pila de comandos:
Comando 4 -> Comando 3 -> Comando 2 -> Comando 1 -> null

Eliminación con POP
Comando 3 -> Comando 2 -> Comando 1 -> null

Ver último comando PEEK
Comando 3

=== PRUEBA DE FUNCIONES COLA ===
Programa 1 puesto en cola ENQUEUE
Programa 2 puesto en cola ENQUEUE
Programa 3 puesto en cola ENQUEUE
Programa 4 puesto en cola ENQUEUE

Cola de programas:
Programa 1 -> Programa 2 -> Programa 3 -> Programa 4 -> null

Eliminación con DEQUEUE
Programa 2 -> Programa 3 -> Programa 4 -> null

Ver primer programa PEEK
Programa 2
PS C:\Users\leona\Desktop\Actividad 2>
```

```
*** ¡¡¡Bienvenido a tu sistema operativo!!! ***

¿Qué deseas realizar hoy?

1- Gestionar comandos
2- Gestionar programas
3- Salir del programa

Ingresa una opción: 2

*** ¡¡¡Bienvenido al gestor de aplicaciones!!! ***

¿Qué tarea deseas realizar?

1- Abrir una aplicación
2- Finalizar programa en ejecución
3- Ver programa en ejecución
4- Mostrar los programas en espera
5- Regresar al SO

Ingresa una opción: 1

Escribe el nombre de una aplicación. (Ej. Firefox, Spotify)

[java@datastructures ~]$ Firefox

Abriendo la aplicación...
```