UNIVERSIDAD TECMILENIO

Avance de proyecto

Estructuras de datos

Adalberto Emmanuel Rojas Perea

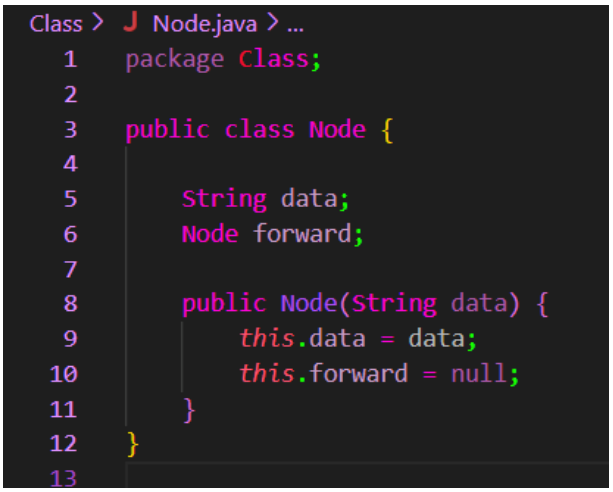
Leonardo Pereda Chánez
AL03087353

Documentación del código

Node.java

La clase **Node** toma como atributos el dato que va a contener **String**, y el puntero que apunta hacía adelante.

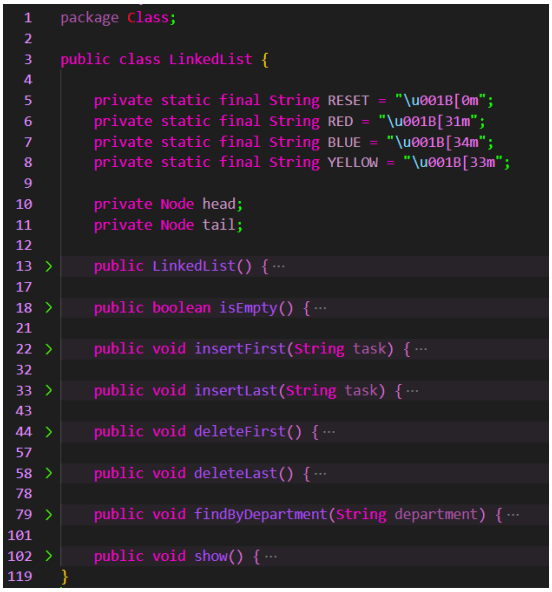
También cuenta con el constructor del Nodo, en el que se le asigna el dato que se le vaya a dar, y el puntero por defecto apunta hacia **null**.



LinkedList.java

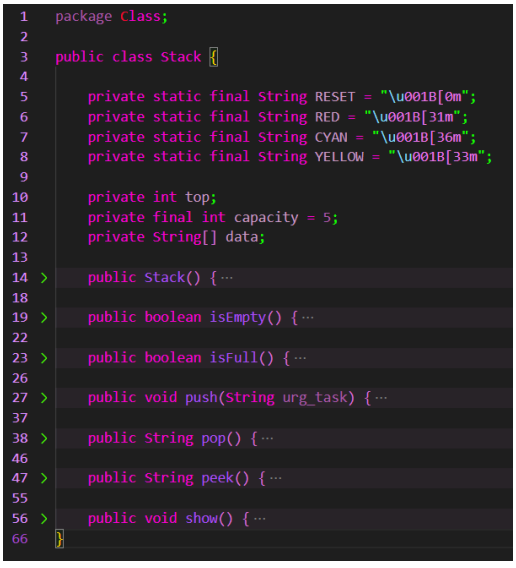
La clase **LinkedList** toma como atributos los Nodos cabeza y cola. Cuenta con el constructor en que el que asigna por defecto los Nodos cabeza y cola como **null.**

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



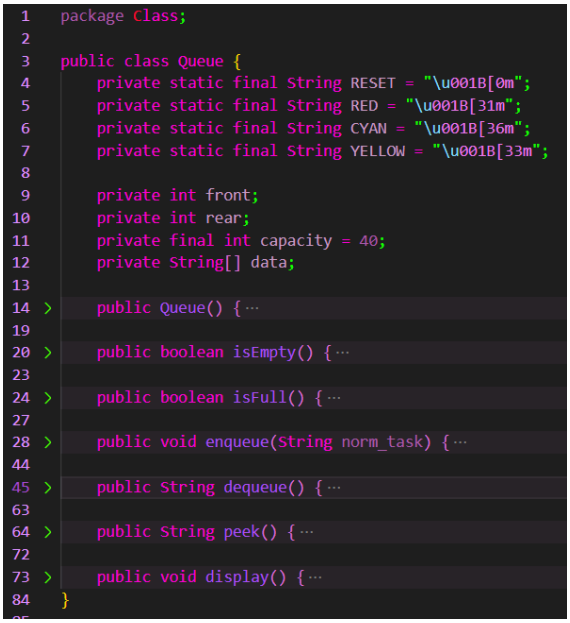
Stack.java

La clase **Stack** se inicializa con una capacidad máxima de 5 incidentes críticos, debido a que un negocio no debería de manejar con demasiadas a la vez, y esta cuenta con sus métodos de **push**, **pop**, **peek**, y **show**. Todo siguiendo con la lógica de LIFO



Queue.java

La clase **Queue**, similar a Stack, se inicializa con una capacidad máxima para programar tareas, y siguiendo la lógica FIFO con sus métodos **enqueue**, **dequeue**, **peek** y **display**.



Main.java

La clase Main está estructurada donde el usuario puede gestionar diferentes tareas para la empresa **IT SOLUTIONS CUU**, donde puede gestionar incidentes críticos de clientes, programar tareas y gestionar tareas por departamentos dentro de la empresa.

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 no termine la ejecución de este de manera abrupta; al igual que el programa muestre los menús de manera organizada.

```java
Run | Debug
public static void main(String[] args) {

    Queue queue = new Queue();
    Stack stack = new Stack();
    LinkedList linkedList = new LinkedList();
    Scanner scanner = new Scanner(System.in);

    int option = 0;

    do {
        System.out.println(YELLOW + "*** IT SOLUTIONS CUU ***" + RESET);
        System.out.println(x:"1- Gestionar incidentes críticos\n2- Programar tareas\n3- Gestionar tareas por departamento\n4- Salir del programa");

        try {
            System.out.print(BLUE + "\nIngresa una opción: " + RESET);
            option = scanner.nextInt();
            scanner.nextLine();
        } catch (InputMismatchException e) {
            System.out.println(RED + "Error. Opción inválida." + RESET);
            scanner.nextLine();
        }

        switch (option) {
            case 1: // Tareas urgentes (Stack)
                int urgOption = 0;

                do {
                    System.out.println(YELLOW + "\n*** Incidentes críticos ***" + RESET);
                    System.out.println(x:"1- Agregar incidente crítico\n2- Ver último incidente\n3- Marcar como resuelto el último incidente\n4- Ver todas los inciden

                    try {
                        System.out.print(BLUE + "\nIngresa una opción: " + RESET);
                        urgOption = scanner.nextInt();
```

```java
            switch (option) {
                case 1: // Tareas urgentes (Stack)
                    int urgOption = 0;

                    do {
                        System.out.println(YELLOW + "\n*** Incidentes críticos ***" + RESET);
                        System.out.println(x:"1- Agregar incidente crítico\n2- Ver último incidente\n3- Marcar como resuelto el último incidente\n4- Ver todas los incider

                        try {
                            System.out.print(BLUE + "\nIngresa una opción: " + RESET);
                            urgOption = scanner.nextInt();
                            scanner.nextLine();
                        } catch (InputMismatchException e) {
                            System.out.println(RED + "Error." + RESET);
                            scanner.nextLine();
                        }

                        switch (urgOption) {
                            case 1:
                                System.out.print(CYAN + "\nDetalles del incidente (Ej. Servidor caído, BDD corrompida): " + RESET);
                                String urgTask = scanner.nextLine();
                                stack.push(urgTask);
                                break;

                            case 2:
                                System.out.print(CYAN + "\nÚltima incidente agregado: " + RESET + YELLOW + stack.peek() + RESET + "\n");
                                break;

                            case 3:
                                String solvedStack = stack.pop();
                                System.out.println(CYAN + "\nIncidente " + RESET + YELLOW + solvedStack + RESET + CYAN + " marcado como resuelto." + RESET);
                                break;

                            case 4:
                                stack.show();
```

Evidencias de funcionamiento

```
PS C:\Users\leona\Desktop\avance-de-proyecto-edd>  & 'C:\Users\leona\AppData\L
\leona\AppData\Roaming\Code\User\workspaceStorage\96310efcdac087fb3189c5dd8f22
*** IT SOLUTIONS CUU ***
1- Gestionar incidentes críticos
2- Programar tareas
3- Gestionar tareas por departamento
4- Salir del programa

Ingresa una opción: 1

*** Incidentes críticos ***
1- Agregar incidente crítico
2- Ver último incidente
3- Marcar como resuelto el último incidente
4- Ver todas los incidentes críticos
5- Regresar al gestor de tareas

Ingresa una opción: 1

Detalles del incidente (Ej. Servidor caído, BDD corrompida): Servidor caído

*** Incidentes críticos ***
1- Agregar incidente crítico
2- Ver último incidente
3- Marcar como resuelto el último incidente
4- Ver todas los incidentes críticos
5- Regresar al gestor de tareas

Ingresa una opción: 2

Última incidente agregado: Servidor ca?do

*** Incidentes críticos ***
1- Agregar incidente crítico
2- Ver último incidente
3- Marcar como resuelto el último incidente
4- Ver todas los incidentes críticos
```