# MANUAL TÉCNICO
# JUEGO: ESCAPE DEL LABERINTO

## SOFTWARE UTILIZADO:

- Lenguaje: **Java 17 con openJDK 17**
- IDE (Integrated Development Environment): **JetBrains IntelliJ IDEA (Community Edition)**
- Sistema operativo: **Fedora Linux 35**
- Herramienta de gestión y construcción de proyecto: **Maven, version: 4.0.0**
- Version: **1.0 – SNAPSHOT**
- Paquetería: **Jar**

- **ARCHIVO: Main.java**

```
package practica1.ipc1;
import java.util.Scanner;

public class Main{
```

En esta parte se declaran las variables globales donde se guardaran, los mapas que se vayan creando, el oro que requieren, y su nombre.

```
	// VARS GLOBALES
	static Scanner input = new Scanner(System.in);
	static String[][][] maps;
	static int[] reqGoldList = new int[10];
	static String[] mapNamesList = new String[10];
	static int mapIndex = 1;


	public static void main(String[] args) {

		// definimos el limite de mapas a 10
		maps = new String[10][][];

		// inicia: asignación de mapa por defecto como primer mapa en la
lista
```

Se define el mapa por defecto de manera manual en un array de 2 dimensiones

```
	String[][] mapDefault
={{"#","#","#","#","#","#","#","#","S","#","#","#","#","#","#","#","#","#","#","#"
,"#","#","#","#","#","#","#","#","#","#"},
{"S","O","O","O","O","O","O","#","O","O","O","O","O","O","O","O","O","#","O","O","
O","O","O","O","O","O","#","O","O","S"},
```

```
{"#","#","#","#","#","#","O","O","#","#","#","O","#","#","#","O","#","O","O","#","
#","#","#","#","#","O","O","#","O","#"},
{"#","O","O","G","O","#","O","#","G","G","#","O","#","G","#","O","#","G","#","O","
#","O","O","O","#","#","O","O","O","#"},
{"#","O","#","#","O","#","O","O","O","O","#","G","#","O","#","O","#","#","O","O","
#","O","#","O","O","#","O","#","O","#"},
{"#","G","#","O","O","#","O","#","#","#","#","#","#","O","#","O","O","O","#","O","
#","O","#","#","#","#","O","#","O","#"},
{"#","O","#","#","O","O","O","O","O","O","#","O","#","O","O","O","O","O","O","O","
O","O","#","O","O","O","O","#","#","#"},
{"#","#","O","O","O","#","O","#","O","#","O","O","#","#","O","#","#","O","#","#","
#","#","#","O","#","O","#","#","G","#"},
{"#","O","O","#","O","#","O","O","O","#","O","#","O","#","O","O","#","O","#","O","
O","O","O","O","#","O","O","#","G","#"},
{"#","O","#","#","O","#","#","#","O","#","O","#","O","#","#","O","#","O","#","#","
#","O","#","#","#","#","G","#","O","#"},
{"#","O","O","O","O","#","O","#","O","#","O","#","O","#","O","O","#","O","O","G","
#","O","O","#","O","#","O","#","O","#"},
{"#","#","#","G","#","O","O","O","O","#","O","#","O","#","#","O","#","#","#","#","
#","O","#","o","#","#","O","#","O","#"},
{"#","O","O","#","O","O","#","#","#","O","O","#","O","#","O","O","O","O","O","O","
O","O","O","O","O","#","O","#","O","#"},
{"#","#","O","#","#","O","#","O","O","#","O","#","O","O","O","#","O","#","#","O","
#","O","O","#","O","#","O","#","O","#"},
{"#","O","O","#","O","O","#","O","#","#","O","#","G","#","O","#","#","O","O","#","
O","O","#","O","O","#","O","#","O","#"},
{"#","#","O","#","#","O","O","O","G","#","O","#","#","#","O","#","O","O","O","O","
O","#","O","O","#","#","O","#","O","#"},
{"#","O","O","#","O","O","#","O","#","#","O","O","#","O","O","#","O","#","#","#","
#","#","O","#","O","#","G","#","O","#"},
{"#","#","O","O","O","#","#","#","O","#","O","#","O","#","O","O","O","#","G","O","
O","O","O","O","O","#","#","O","#"},
{"#","G","#","O","#","O","O","O","O","#","O","O","#","#","#","#","#","#","#","#","
#","#","#","#","O","O","O","O","#"},
{"#","G","#","O","O","O","#","#","O","O","O","O","O","G","O","O","O","G","O","O","
O","G","O","O","#","#","O","#","O","#"},
```

```
{"#","O","#","#","#","#","#","O","O","#","#","#","#","#","#","#","#","#","#","#","#","#","#","O","O","#","O","#","#","#"},
{"#","O","#","O","O","O","#","O","#","O","O","O","O","O","O","O","O","O","O","O","O","O","O","O","O","#","O","#","O","O","O","S"},
{"#","O","O","O","#","O","#","O","O","#","O","#","#","#","#","#","#","O","#","#","#","O","O","O","O","#","O","#","#","#"},
{"#","#","#","#","#","O","#","O","#","O","O","#","G","O","O","O","#","O","O","O","#","O","#","#","#","#","O","#","O","#"},
{"#","O","#","O","O","O","#","O","G","#","O","#","#","#","#","O","#","#","#","O","#","O","O","O","O","#","O","O","O","#"},
{"#","O","#","#","#","O","#","#","#","#","O","#","O","O","O","O","#","G","#","O","#","#","#","O","#","G","#","#","O","#"},
{"#","O","#","O","O","O","O","O","O","O","O","O","O","#","O","O","#","O","#","O","O","O","O","#","O","#","O","O","#"},
{"#","O","#","#","#","#","#","#","#","#","#","O","#","O","#","#","#","O","#","O","#","#","#","#","#","O","O","O","#","#"},
{"#","O","O","O","O","O","G","#","O","O","O","O","#","O","O","O","O","O","O","O","O","O","O","O","O","#","O","G","#"},
{"#","#","#","#","#","#","#","#","#","#","#","#","#","#","S","#","#","#","#","#","#","#","#","#","#","#","#","#","#","#"}};

            maps[0] = new String[mapDefault.length][mapDefault.length];

            for (int i = 0; i < mapDefault.length; i++) {
                  for (int j = 0; j < mapDefault.length; j++) {
                        maps[0][i][j] = mapDefault[i][j];
                  }
            }

            reqGoldList[0] = 15;
            mapNamesList[0] = "Predeterminado";

            // termina: asignación de mapa por defecto como primer mapa en la lista, oro requerido para mapa por defecto y nombre
```

Se asigna como primer mapa en la lista de los mapas

```
            mainMenu();

      }
```

Función menú principal, la cual muestra el menú principal y ejecuta las demás partes a partir de la elección del usuario *NOTA: No todas las partes del juego son funcionales*

```
	// MENU PRINCIPAL
	public static void mainMenu() {

		int selection = 0;

		String mainMenuOptions = """
				================ Escape del laberinto =================
				==================== Menú principal ===================

				1): Jugar
				2): Crear mapa
				3): Reportes / Estadísticas de juego
				4): Visualizar mapa
				5): Salir del juego
				>\040""";

		selection = askForNumberL(mainMenuOptions, 1, 5);


		switch (selection) {
			case 1 :
				System.out.println("====== Jugar ======");
				System.out.println("!! Parte sin funcionamiento !!");
				break;
			case 2: {
				if (mapIndex == 10) {
					System.out.println("Aviso: Ya se ha alcanzado el
maximo de mapas creados, no se pueden crear mas");
					mainMenu();
				}
				prepareMapCreation();
				break;
			}
			case 3:
				System.out.println("====== Mostrando reportes generales
======");
				System.out.println("!! Parte con reducido
funcionamiento !!");
				showStats();
				break;
			case 4:
				System.out.println("====== Entrando a previsualizador
de mapas ======");
				previewMap(maps);
				break;
			case 5:
				System.out.println("====== Saliendo del juego ...
======");
				System.exit(0);
				break;
		}

	}
	// TEMRINA MENU PRINCIPAL
```

```
	// MOSTRAR REPORTES GENERALES
```

Función para mostrar los reportes generales
(Solo muestra el total de mapas creados)

```
	public static void showStats(){

		System.out.println("** Total de mapas creados **");

		for (int i = 0; i < maps.length; i++) {
			if (maps[i] != null) {
				System.out.println(i+1);
			} else {
				break;
			}
		}

		mainMenu();

	}

	// TERMINA MOSTRAR REPORTES GENERALES
```

Función para visualizar un mapa, seleccionado de la lista de mapas que están disponibles

```
	// PREVISUALIZADOR DE MAPAS
	public static void previewMap(String[][][] maps){
		int selection;
		int avaliableMaps = 0;

		// obtenemos la cantidad de mapas dispobnibles

		for (int i = 0; i < maps.length; i++) {
			if (maps[i] != null) {
				avaliableMaps = i+1;
			} else {
				break;
			}
		}

		do {

			System.out.println("Mapas disponibles:");
			// listamos los mapas que estan disponibles
			for (int i = 0; i < maps.length; i++) {
				if (maps[i] != null) {
					System.out.println("Mapa #"+(i+1)+" Nombre:
"+(mapNamesList[i]));
				} else {
					break;
				}
			}
			// pedimos el mapa a mostrar
```

```
                    selection = askForNumberL("Selecciona un mapa de los "
+avaliableMaps+ " disponibles, o escribe 0 para regresar a menu principal\n> ", 0,
avaliableMaps);

                    // finalmente, mostramos el mapa seleccionado o salimos del
previsualizador

                    if (selection > 0){
                          for (int i = (selection-1); i < maps.length;) {
                                System.out.println("Mostrando mapa #"+(i+1));
                                for (int j = 0; j < maps[i].length; j++) {
                                      for (int k = 0; k < maps[i][j].length;
k++) {
                                            System.out.print(maps[i][j][k]+" ");
                                      }
                                      System.out.println("");
                                }
                                System.out.println("\nNombre de este mapa:
"+mapNamesList[i]);
                                System.out.println("Oro requerido para salir en
este mapa: "+reqGoldList[i]+"\n");
                                break;
                          }

                    }

             } while (selection != 0);

             mainMenu();

       }
       // TERMINA PREVISUALIZADOR DE MAPAS
```

Función para la creación de maps

Se prepara la creación, preguntando por los parámetros que tendrá el mapa, y luego llamando al constructor de la clase *CustomMap* para crear un nuevo mapa, que será llenado por el usuario, con los parámetros que se han pedido.

```
       // PREPARAR CREACION DE MAPAS
       public static void prepareMapCreation(){

             System.out.println("====== Entrando a creador de mapas ======");
             System.out.println("Aviso: Puede crear hasta 9 mapas");
             System.out.println("Comenzando creación del nuevo mapa:");
             System.out.println("De que tamaño desea crearlo?");

             // pedimos los parametros del mapa
             int rows = askForNumberGZ("Filas? > ");
             int columns = askForNumberGZ("Columnas? > ");
             int reqGold = askForNumberGZ("Cantidad de oro necesario para salir? >
");

             System.out.print("Nombre? > ");
             String name = input.next();
```

```
            // guardamos el oro requerido y el nombre
            reqGoldList[mapIndex] = reqGold;
            mapNamesList[mapIndex] = name;

            // construimos un nuevo mapa con la clase CustomMap
            CustomMap newMap = new CustomMap(rows, columns, reqGold, name);

            // Lo llenamos por el usuario
            newMap.FillMap();

            // Lo guardamos en la lista de mapas
            maps[mapIndex] = newMap.generatedMap;

            mapIndex += 1;

            // una vez terminada la creacion de un mapa, regresar al menu
            mainMenu();

      }
      // TERMINA PREPARAR CREACION DE MAPAS
```

Función auxiliares que sirven para pedir un numero dependiendo del caso que se necesite, las cuales también se aseguran que el valor introducido sea el correcto

```
      // FUNCIONES AUXILIARES

      // funcion para pedir numeros dentro de un rango
      public static int askForNumberL(String message, int lMin, int lMax){
            int enteredNumber = 0;
            boolean inputError = false;

            do {
                  try {
                        System.out.print(message);
                        enteredNumber = input.nextInt();
                        if ((enteredNumber >= lMin) && (enteredNumber <= lMax))
{
                              inputError = false;
                        } else {
                              inputError = true;
                              System.out.println("El valor ingresado debe ser
entre "+ lMin +" y " + lMax);
                              System.out.println("Intenta de nuevo");
                        }

                  } catch (Exception ex) {
                        inputError = true;
                        System.out.println("El valor ingresado no es valido,
intenta de nuevo");
                        input.next();

                  }
```

```
            } while (inputError);
            return enteredNumber;
      }

      // funcion para pedir numeros mayor que cero
      public static int askForNumberGZ(String message){
            int enteredNumber = 0;
            boolean inputError = false;

            do {
                  try {
                        System.out.print(message);
                        enteredNumber = input.nextInt();

                        if ((enteredNumber > 0)) {
                              inputError = false;
                        } else {
                              inputError = true;
                              System.out.println("El valor ingresado debe ser
mayor que cero");
                              System.out.println("Intenta de nuevo");
                        }

                  } catch (Exception ex) {
                        inputError = true;
                        System.out.println("El valor ingresado no es valido,
intenta de nuevo");
                        input.next();

                  }

            } while (inputError);
            return enteredNumber;
      }

      // funcion para pedir numeros sin limite
      public static int askForNumberNL(String message){
            int enteredNumber = 0;
            boolean inputError = false;

            do {
                  try {
                        System.out.print(message);
                        enteredNumber = input.nextInt();
                        inputError = false;

                  } catch (Exception ex) {
                        inputError = true;
                        System.out.println("El valor ingresado no es valido,
intenta de nuevo");
                        input.next();

                  }

            } while (inputError);
```

```
            return enteredNumber;
      }
}
```

- **ARCHIVO: CustomMap.java**

Esta clase nos servirá como plano para crear nuevos mapas, con sus respectivas características, como el tamaño, y que luego serán llenado por el usuario

```
package practica1.ipc1;
import java.util.Scanner;


public class CustomMap {

      // VARS GLOBALES
      public static Scanner input = new Scanner(System.in);
      static String name;
      static int columns;
      static int rows;
      static int reqGold;
      static String[][] generatedMap;

      // CONSTRUCTOR
      CustomMap(int rows, int columns, int reqGold, String name) {
            this.rows = rows;
            this.columns = columns;
            this.reqGold = reqGold;
            this.name = name;

            generatedMap = new String[rows][columns];
      }

      // FUNCION PARA PEDIR NUMERO MAYOR QUE CERO
      public static int askForNumberGZ(String message){
            int enteredNumber = 0;
            boolean inputError = false;

            do {
                  try {
                        System.out.print(message);
                        enteredNumber = input.nextInt();

                        if ((enteredNumber > 0)) {
                              inputError = false;
                        } else {
                              inputError = true;
                              System.out.println("El valor ingresado debe ser
mayor que cero");
                              System.out.println("Intenta de nuevo");
                        }

                  } catch (Exception ex) {
```

```
                        inputError = true;
                        System.out.println("El valor ingresado no es valido,
intenta de nuevo");
                        input.next();

                  }

            } while (inputError);
            return enteredNumber;
      }

      // FUNCION PARA PEDIR NUMEROS POSITIVOS
      public static int askForNumber(String message){
            int enteredNumber = 0;
            boolean inputError = false;

            do {
                  try {
                        System.out.print(message);
                        enteredNumber = input.nextInt();

                        if ((enteredNumber >= 0)) {
                              inputError = false;
                        } else {
                              inputError = true;
                              System.out.println("El valor ingresado debe ser
positivo");
                              System.out.println("Intenta de nuevo");
                        }

                  } catch (Exception ex) {
                        inputError = true;
                        System.out.println("El valor ingresado no es valido,
intenta de nuevo");
                        input.next();

                  }

            } while (inputError);
            return enteredNumber;
      }

      // funcion para pedir numeros dentro de un rango
      public static int askForNumberL(String message, int lMin, int lMax){
            int enteredNumber = 0;
            boolean inputError = false;

            do {
                  try {
                        System.out.print(message);
                        enteredNumber = input.nextInt();
                        if ((enteredNumber >= lMin) && (enteredNumber <= lMax))
{
                              inputError = false;
                        } else {
```

```
                                inputError = true;
                                System.out.println("El valor ingresado debe ser
entre "+ lMin +" y " + lMax);
                                System.out.println("Intenta de nuevo");
                          }

                    } catch (Exception ex) {
                          inputError = true;
                          System.out.println("El valor ingresado no es valido,
intenta de nuevo");
                          input.next();

                    }

             } while (inputError);
             return enteredNumber;
       }

       // FUNCION PARA LLENAR EL MAPA POR EL USUARIO
       public void FillMap() {

             System.out.println("Comenzando llenado de mapa...");

             // CASILLAS DE SALIDA
             System.out.println("Colocando salidas del mapa:");
             int exitsCount = askForNumberGZ("Cuantas salidas tendra este mapa? >
");

             for (int i = 0; i < exitsCount; i++) {

                    // mostramos el mapa en su estado actual
                    System.out.println("Mapa actual:");
                    for (int k = 0; k < generatedMap.length; k++) {
                          for (int l = 0; l < generatedMap[k].length; l++) {
                                if (generatedMap[k][l] == null) {
                                      generatedMap[k][l] = "-";
                                }
                                System.out.print(generatedMap[k][l]+" ");
                          }
                          System.out.println("");
                    }


                    System.out.println("Tomando coordenadas de las salidas: (debes
ingresarlas solo en las paredes)");
                    System.out.println("Donde se colocara la salida #"+(i+1)+"?");
                    int rowExit = 0;
                    int columnExit = 0;
                    boolean inputCorrect = false;

                    do {
                          rowExit = askForNumberL("Fila? > ", 1, rows);
                          rowExit -= 1;
                          columnExit = askForNumberL("Columna? > ", 1, columns);
                          columnExit -= 1;
```

```
                        if(alreadyFilled(generatedMap, rowExit, columnExit)){
                              System.out.println("Error: esa casilla ya esta
llena");
                        } else { inputCorrect = true; }

                        generatedMap[rowExit][columnExit] = "S";
                  } while (!inputCorrect);

            }
            // TERMINA CASILLAS DE SALIDA

            // mostramos el mapa en su estado actual
            System.out.println("Mapa actual:");
            for (int k = 0; k < generatedMap.length; k++) {
                  for (int l = 0; l < generatedMap[k].length; l++) {
                        if (generatedMap[k][l] == null) {
                              generatedMap[k][l] = "-";
                        }
                        System.out.print(generatedMap[k][l]+" ");
                  }
                  System.out.println("");
            }

            // CASILLAS DE ORO
            System.out.println("Colocando casillas de oro:");

            boolean goldCountCorrect = false;
            int goldCount = 0;
            do {
                  goldCount = askForNumberGZ("Cuantas casillas con oro tendra
este mapa? Debe ser mayor de " + reqGold + " > ");
                  if (goldCount < reqGold) {
                        System.out.println("Error: El mapa debe tener la misma
cantidad o mas casillas de oro que el oro requerido para salir");
                  } else { goldCountCorrect = true; }

            } while (!goldCountCorrect);



            for (int i = 0; i < goldCount; i++) {

                  // mostramos el mapa en su estado actual
                  System.out.println("Mapa actual:");
                  for (int k = 0; k < generatedMap.length; k++) {
                        for (int l = 0; l < generatedMap[k].length; l++) {
                              if (generatedMap[k][l] == null) {
                                    generatedMap[k][l] = "-";
                              }
                              System.out.print(generatedMap[k][l]+" ");
                        }
                        System.out.println("");
                  }

                  System.out.println("Tomando coordenadas de casillas de oro:");
```

```
                    System.out.println("Donde se colocara la casilla de oro #" +
(i + 1) + "?");
                    int rowGold = 0;
                    int columnGold = 0;
                    boolean inputCorrect = false;

                    do {
                          rowGold = askForNumberL("Fila? > ", 1, rows);
                          rowGold -= 1;
                          columnGold = askForNumberL("Columna? > ", 1, columns);
                          columnGold -= 1;
                          if(alreadyFilled(generatedMap, rowGold, columnGold)){
                                System.out.println("Error: esa casilla ya esta
llena");
                          } else { inputCorrect = true; }

                          generatedMap[rowGold][columnGold] = "G";
                    } while (!inputCorrect);

             }
             // TERMINA CASILLAS DE ORO


             // mostramos el mapa en su estado actual
             System.out.println("Mapa actual:");
             for (int k = 0; k < generatedMap.length; k++) {
                    for (int l = 0; l < generatedMap[k].length; l++) {
                          if (generatedMap[k][l] == null) {
                                generatedMap[k][l] = "-";
                          }
                          System.out.print(generatedMap[k][l]+" ");
                    }
                    System.out.println("");
             }

             // CASILLAS DE PAREDES
             System.out.println("Colocando paredes:");

             // Calculamos cuantos espacios restan donde podemos colocar paredes y
espacios vacios
             int leftSpaces = ((rows*columns) - exitsCount - goldCount);
             for (int i = 0; i < leftSpaces; i++) {
                    System.out.println("Elige que llenar:");
                    System.out.println("1) Pared");
                    System.out.println("2) Casilla libre");

                    int selection = askForNumberL("> ", 1, 2);
                    boolean inputCorrect = false;

                    if (selection == 1){
                          do {
                                int rowWallOrSpace = askForNumber("Fila? > ");
                                rowWallOrSpace -= 1;
                                int columnWallOrSpace = askForNumber("Columna? >
");
```

```
                                    columnWallOrSpace -= 1;
                                    if(alreadyFilled(generatedMap, rowWallOrSpace,
columnWallOrSpace)){
                                          System.out.println("Error: esa casilla ya
esta llena");
                                    } else { inputCorrect = true; }

                                    generatedMap[rowWallOrSpace][columnWallOrSpace] =
"#";
                              } while (!inputCorrect);
                        } else if (selection == 2) {
                              do {
                                    int rowWallOrSpace = askForNumber("Fila? > ");
                                    rowWallOrSpace -= 1;
                                    int columnWallOrSpace = askForNumber("Columna? >
");
                                    columnWallOrSpace -= 1;
                                    if(alreadyFilled(generatedMap, rowWallOrSpace,
columnWallOrSpace)){
                                          System.out.println("Error: esa casilla ya
esta llena");
                                    } else { inputCorrect = true; }

                                    generatedMap[rowWallOrSpace][columnWallOrSpace] =
"O";
                              } while (!inputCorrect);
                        }

                        // mostramos el mapa en su estado actual
                        System.out.println("Mapa actual:");
                        for (int k = 0; k < generatedMap.length; k++) {
                              for (int l = 0; l < generatedMap[k].length; l++) {
                                    if (generatedMap[k][l] == null) {
                                          generatedMap[k][l] = "-";
                                    }
                                    System.out.print(generatedMap[k][l]+" ");
                              }
                              System.out.println("");
                        }
                  }
                  System.out.println("Mapa creado y guardado exitosamente");

                  // TERMINA CASILLAS DE PAREDES

            }

            // FUNCION PARA COMPROBAR SI YA SE LLENO LA CASILLA QUE SE ESTÁ INTENTANDO
LLENAR
            private boolean alreadyFilled(String[][] mapToVerify, int row, int column){
                  if ("S".equals(mapToVerify[row][column]) ||
"G".equals(mapToVerify[row][column]) || "#".equals(mapToVerify[row][column]) ||
"O".equals(mapToVerify[row][column])){
                        return true;
                  } else {
                        return false;
```

```
                }
            }
}
```