MANUAL TECNICO

Practica 1

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

LABORATORIO DE INTRODUCCION A LA PROGRAMACION Y COMPUTACION 1, SECCION D

Juan Josue Zuleta Beb, carné: 20206353

Requisitos del Sistema

Sistema operativo 64 bits

- Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit)
- macOS 10.5 o superior
- Linux GNOME o KDE desktop

RAM: 1gb como mínimo.
Disco duro: 300mb como mínimo + 1gb para cache.
IDE: Intellij Idea
JDK: Versión 16
Resolución de pantalla: 1024×768 resolución mínima de pantalla

Paradigma de programación

Arreglos de una y dos dimensiones

Inicio del programa:

Para iniciar el menú del programa se utilizo un ciclo Do-While dentro del cual se ejecutará siempre que se cumpla la condición un método Escáner acompañado de un método switch case.

La estructura del switch case será la siguiente:

Caso 1:

Leer y escribir el tablero principal y el tablero de penalizaciones, la posición del jugador, recorrer el tablero mientras que la posición del jugador sea menor que 64 y si se cumple la condición se llamará al método Dado para que se obtenga un numero al azar, de esta forma se recorrerá el tablero.

Caso 2:

Leer el tablero de posiciones y penalizaciones, así como también la última posición del jugador e imprimirlas en pantalla, en este caso no se volverán a ejecutar las líneas de código que impliquen llenar el tablero de posición de penalizaciones o que la posición inicial del jugador sea cero.

Caso 3:

Se llamara al método generarReportes();

Del cual podremos escribir un archivo con extensión HTML por me dio del cual recabaremos la información almacenada en memoria en la última ejecución en curso del programa, si se ejecuta el programa sin ninguna otra acción de por medio, el reporte estará vacío, ya que aun no hay datos para almacenarlos en este.

Caso 4:

Se llamará un caso por Default el cual terminara con el proceso del ciclo Do-while y terminará con el proceso del switch-case. El programa saldrá y no guardará ningún estado.

Llenado del tablero principal:

```
public static int [][] tablero1 = {
        {64,63,62,61,60,59,58,57},
        {49,50,51,52,53,54,55,56},
        {48,47,46,45,44,43,42,41},
        {33,34,35,36,37,38,39,40},
        {32,31,30,29,28,27,26,25},
        {17,18,19,20,21,22,23,24},
        {16,15,14,13,12,11,10,9},
        {1,2,3,4,5,6,7,8}
};
```

El tablero principal se ha inicializado con los valores predeterminados para el tablero. Su acceso es publico para que los diferentes métodos puedan leerlo sin problema.

Llenado del tablero de penalizaciones:

```java
//LLENADO DE LA MATRIZ SECUNDARIA

for (int i = 0; i < tablero2.length; i++) {
    int countpen = 0;
    for (int j = 0; j < tablero2[i].length; j++) {
        if (countpen < 4) {
            tablero2[i][j] = tool.nextInt( bound: 2);
            if (tablero2[i][j] == 1 ) {
                countpen++;
            }
        }
    }
}

//LLENADO DE LAS UBIACIONES DEPENALIZACIONES

for (int i = 0; i < tablero2.length; i++) {
    for (int j = 0; j < tablero2[i].length; j++) {
        if (tablero2[i][j] == 1){
            posPena[i][j] = true;
        }else{
            posPena[i][j] = false;
        }
    }
}
```

Para esta parte del algoritmo se realizo un llenado de una matriz en la cual se lanzan al azar un valor entre cero y uno, estos valores son leídos por la segunda matriz, la cual convertirá cada valor de uno en un valor verdadero y cada valor de cero en un valor falso, para una matriz booleana.

Impresión del tablero, posición del jugador y posición de las penalizaciones:

```java
for (int i = 0; i < tablero1.length; i++) {
    System.out.println("-----------------------------------------------------------------------------------------------------------");
    for (int j = 0; j < tablero1[i].length; j++) {
        if (posJugador != tablero1[i][j] && !posPena[i][j]) {
            System.out.print("|    " + tablero1[i][j] + "\t |");
        }
        if (posJugador == tablero1[i][j] && posPena[i][j]) {
            System.out.print("|    " + tablero1[i][j] + "#@" + "\t |");
            count = 1;
        } else if (posJugador == tablero1[i][j]) {
            System.out.print("|    " + tablero1[i][j] + "@" + "\t |");
        } else if (posPena[i][j]) {
            System.out.print("|    " + tablero1[i][j] + "#" + "\t |");
        }
    }
    System.out.println();
}
System.out.println("-----------------------------------------------------------------------------------------------------------");
```

Este ciclo for nos ayudara a leer nuestra matriz principal del tablero se recorrerá cada fila y columna y se imprimirá en pantalla, al mismo tiempo se validarán por medio se ciclos “if”, si se cumple que la posición del jugador y la posición de una penalización están en la misma casilla, entonces se imprimirá un “@#”, si la casilla únicamente corresponde a la posición del jugador entonces se imprimirá un “@”.

Verificación de penalizaciones:

```
if (count == 1) {
    if (posJugador >= tablero1[2][7] && posJugador <= tablero1[0][0]) {

        if (count2 != 2) {
            aux = tool.nextInt( bound: 4);
        }else{
            aux = 5;
        }
        if (aux == 1) {
            penalizacionD1(); count2++; countImpresionPD = 1;
        } else if (aux == 2) {
            penalizacionD3(); count2++; countImpresionPD = 2;
        } else if(aux == 3) {
            penalizacionD2(); count2++; countImpresionPD = 3;
        }
        if (aux == 5){
            //System.out.println("No habra mas penalizaciones Dificiles");
        }
```

En este método se verifica la posición del jugador y los rangos del tablero, en este ejemplo vemos únicamente los rangos de penalizaciones de nivel difícil las cuales comprenden las 3 ultimas filas superiores, y funciona como sigue:

Si el jugador está en la penalización y esta corresponde al rango difícil, entonces se saca un numero al azar entre 1 y 3 para decidir cual de las 3 opciones de penalización se ejecutará, de esta forma se elige siempre una penalización al azar cada vez que el jugador cae en una casilla penalizada. Este mismo método se implementa para las penalizaciones de nivel medio y las de nivel fácil.

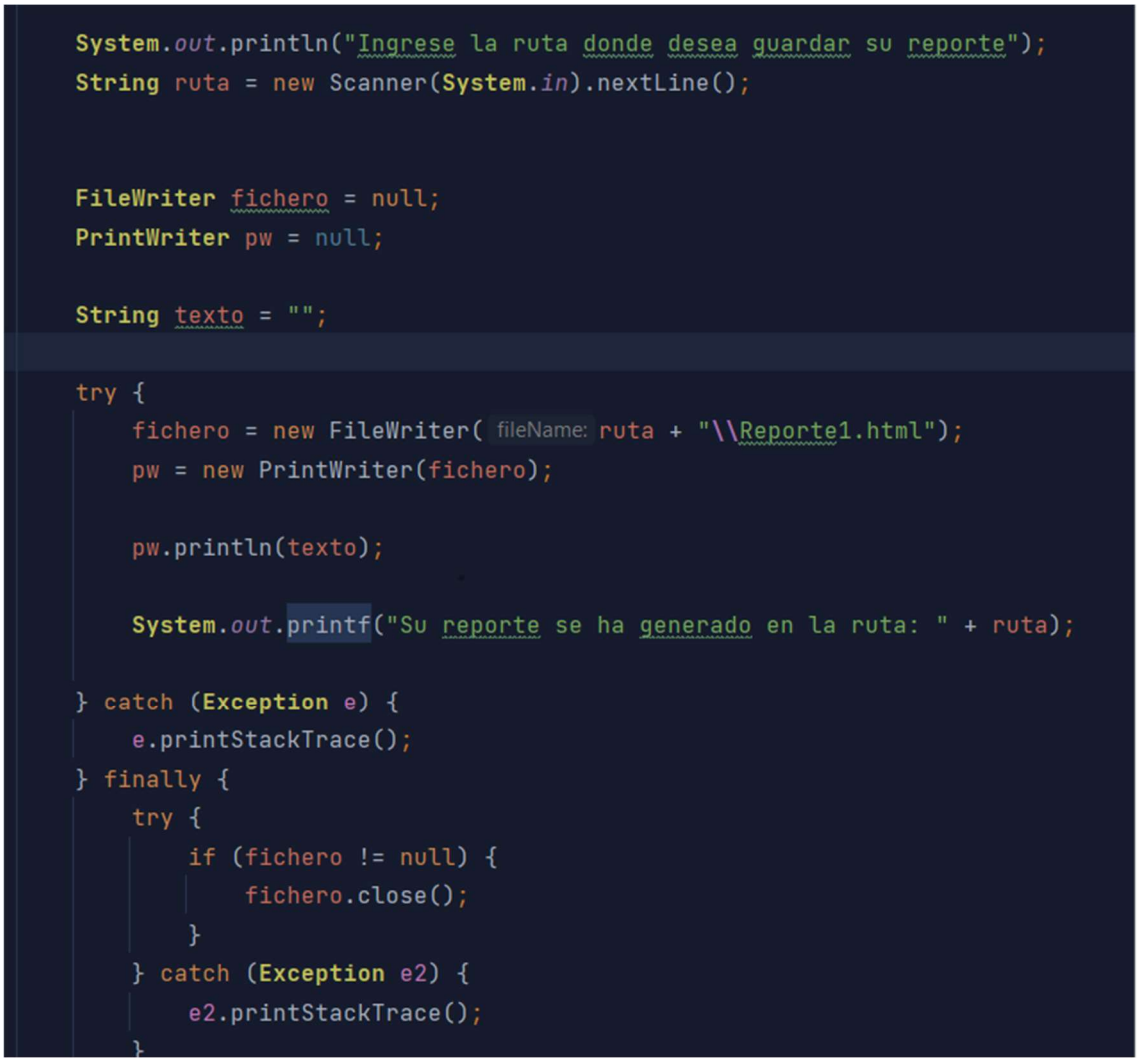
Método del dado y movimiento del jugador:

```
// =========> Dado y movimiento del jugador <=========
public static void dado() {

    try {
        leer2 = escribe.next().charAt(0);
    } catch (Exception e) {}
    if (leer2 == '1') {
        mover = (int) (Math.random() * 4) + 2;
        System.out.println(" =====>| has sacado un: " + mover + " |<===== ");
        if (mover != 0) {
            posJugador = posJugador + mover;
        }
        System.out.println(" =====>| Tu posicion actual es: " + posJugador + " |<===== ");
    }
}
```

Este método al ejecutarse elige un numero al azar entre 2 y 6 el cual se sumará a la posición del jugador actual, este método se ejecutará cada vez que el jugador elija la opción de jugar.

Método para generar reportes:



```
System.out.println("Ingrese la ruta donde desea guardar su reporte");
String ruta = new Scanner(System.in).nextLine();


FileWriter fichero = null;
PrintWriter pw = null;

String texto = "";

try {
    fichero = new FileWriter( fileName: ruta + "\\Reporte1.html");
    pw = new PrintWriter(fichero);

    pw.println(texto);

    System.out.printf("Su reporte se ha generado en la ruta: " + ruta);

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (fichero != null) {
            fichero.close();
        }
    } catch (Exception e2) {
        e2.printStackTrace();
    }
```

Para este método se utiliza un código especial que lee una ruta de asignación de impresión, mediante la cual el usuario determinará la ruta en la cual desea imprimir el archivo con extensión HTML, este método funciona mediante la creación de una pagina web por medio de HTML y CSS estos se crean como un machote dentro del cual el lenguaje Java escribirá los datos que se desean imprimir.

Todas las variables que se desean imprimir junto con el machote en formato de escritura HTML se deben sustituir en la variable texto por medio de la cual se decidirá cuales son los textos que ingresaran por medio de Java.