MANUAL TÉCNICO

PRÁCTICA 1, IPC1

PRIMER SEMESTRE 2022

BRANDON ANDY JEFFERSON TEJAXÚN PICHIYÁ

202112030

Clase Acciones

Método “jugar”

Se crearon objetos de las clases Mapa, Fruta y Pacman.

```
Mapa nGame = new Mapa();
Fruta fruta = new Fruta();
Pacman pacman = new Pacman();
```

Se asignan dimensiones que tendrá el mapa, para luego almacenar el mapa en la matriz Mapa y posteriormente se añaden los demás elementos.

```
nGame.setColumnas(41);
nGame.setFilas(20);
int[][] mapa = nGame.getMatrizMapa();
mapa = fruta.nuevaFruta(5, mapa);
mapa = pacman.ubicarPac(mapa);
```

Se limpia consola por cada iteración del bucle (por cada movimiento).

```
do {
        new ProcessBuilder("cmd","/c","cls").inheritIO().start().waitFor();
        Barra.pacmanS();
        System.out.println();
        System.out.print(Tablero.getMapa(mapa, pacman));
        mov = sc.next();
        if(mov.equals("W") || mov.equals("w")) {
                mapa = pacman.up(pacman.getPosX(), pacman.getPosY(), mapa);
        }else if(mov.equals("S") || mov.equals("s")) {
                mapa = pacman.down(pacman.getPosX(), pacman.getPosY(), mapa);
        }else if(mov.equals("A") || mov.equals("a")) {
                mapa = pacman.left(pacman.getPosX(), pacman.getPosY(), mapa);
        }else if(mov.equals("D") || mov.equals("d")) {
                mapa = pacman.right(pacman.getPosX(), pacman.getPosY(), mapa);
        }else if(mov.equals("m") || mov.equals("M")) break;
}while(pacman.getPuntos() < 30);
```

Al finalizar la partida se limpiará la consola y se mostrará el estado final del mapa.

```
new ProcessBuilder("cmd","/c","cls").inheritIO().start().waitFor();
String horaFin = hora();

Barra.pacmanS();
System.out.println();
System.out.println(Tablero.getMapa(mapa,pacman).replace("Movimiento: ",""));
System.out.println();
System.out.println("  Puntaje total: " + pacman.getPuntos());
System.out.println("  Movimientos totales: " + pacman.getMovimientos());
System.out.print("  Ingrese su Nombre: ");
String nombre = sc.next();
insertar(nombre,pacman.getPuntos(),pacman.getMovimientos(),horaInicio + " - " + horaFin);
```

Clase Mapa

Función “getMatrizMapa”

Se encarga de asignar dimensiones a la matriz matrizM para posteriormente colocar los distintos elementos que conforman el mapa. Los primeros dos ciclos recorren los extremos de la matriz (primera columna y última columna, primera fila y última fila) asignándole el valor de 1. El tercer ciclo recorre el interior de la matriz omitiendo los extremos y asignándole el valor de 0 ó 1 en posiciones aleatorias.

Al finalizar los procesos se retorna la matriz mapa con sus muros externos e interiores.

```
public int [][] getMatrizMapa(){
	int [][] matrizM = new int [getFilas() + 2][getColumnas() + 2];
	for(int i = 0; i < matrizM.length; i++) {
		matrizM[i][0] = 1;
		matrizM[i][matrizM[i].length - 1] = 1;
	}
	for(int j = 0; j < matrizM[0].length; j++) {
		matrizM[0][j] = 1;
		matrizM[matrizM.length - 1][j] = 1;
	}
	for(int i = 2; i < matrizM.length - 2;) {
		int paso;
		for(int j = 2; j < matrizM[i].length - 2;) {
			matrizM[i][j] = aleatorio.nextInt(100);
			if(matrizM[i][j] < 60) matrizM[i][j] = 0;
			else matrizM[i][j] = 1;
			paso = aleatorio.nextInt(2) + 1;
			j += paso;
		}
		paso = aleatorio.nextInt(2) + 1;
		i += paso;
	}
	return matrizM;
}
```

Clase Fruta

Función “ocupado”

Se encarga de verificar si en la posición en la que se ubicará la fruta no esté ocupada por un muro, una fruta o Pac-Man. Si la posición está ocupada retorna verdadero.

```
public boolean ocupado(int x, int y, int[][] matriz) {
	if(matriz[x][y] == 1 || matriz[x][y] == 2 || matriz[x][y] == 3) return true;
	return false;
}
```

Función “nuevaFruta”

Se encarga de posicionar las frutas dentro de la matriz en una posición aleatoria, recibiendo como parámetros la cantidad y la matriz mapa. No fijará la fruta en una posición ocupada por otro elemento, es decir que buscará otra posición hasta encontrar un sitio vacío. Al encontrar una posición vacía se asignará un valor de 2 a esa posición.

Al finalizar se retorna la matriz mapa con las frutas ubicadas en posiciones aleatorias.

```
public int[][] nuevaFruta(int cantidad,int[][] matriz){
	for(int i = 1; i <= cantidad; i++) {
		int x,y;
		do {
			x = aleatorio.nextInt(matriz.length - 2) + 1;
			y = aleatorio.nextInt(matriz[0].length - 2) + 1;
		}while(ocupado(x,y,matriz));
		matriz[x][y] = 2;
	}
	return matriz;
}
```

Clase Pacman

Se construyeron setters y getters para sus variables.

Función “ocupado”

Se encarga de verificar si en la posición en la que se ubicará Pac-Man no esté ocupada por un muro o una fruta. Si la posición está ocupada retorna verdadero.

```
public boolean ocupado(int x, int y, int[][] matriz) {
	if(matriz[y][x] == 1 || matriz[y][x] == 2) return true;
	return false;
}
```

Función “ubicarPac”

Se encarga de posicionar a Pac-Man dentro de la matriz en una posición aleatoria, recibiendo como parámetro la matriz mapa. No fijará a Pac-Man en una posición ocupada por otro elemento, es decir que buscará otra posición hasta encontrar un sitio vacío. Al encontrar una posición en la que se pueda posicionar a Pac-Man se asignará un valor de 3 a esa posición.

Al finalizar se retornará la matriz con Pac-Man ubicado en una posición aleatoria.

```
public int[][] ubicarPac(int[][] matriz){
	int x,y;
	do {
		x = aleatorio.nextInt(matriz[0].length - 2) + 1;
		y = aleatorio.nextInt(matriz.length - 2) + 1;
	}while(ocupado(x,y,matriz));
	matriz[y][x] = 3;
	setPosX(x);
	setPosY(y);
	return matriz;
}
```

Función “up”

Se encarga de mover a Pac-Man una posición arriba respecto a la posición actual. Si en la posición a la que se moverá Pac-Man hay una fruta (2) se le sumará un valor aleatorio entre 1 y 5 a su punteo. Si en la posición a la que se moverá hay un muro (1) no se efectuará el cambio de posición, a menos que se encuentre en la primera fila después del muro superior externo y se reubicará en la ultima fila antes del muro inferior externo. Si en la posición a la que se moverá no hay nada (0) o hay una fruta (2) se efectuará el cambio de posición y se sumará 1 a la cantidad de movimientos.

Al finalizar se retornará la matriz mapa con Pac-Man en su nueva posición.

```
public int[][] up(int x,int y,int[][] matriz){
	if(matriz[y - 1][x] == 2) {
		int p = getPuntos();
		int nAl = aleatorio.nextInt(5) + 1;
		p += nAl;
		setPuntos(p);
		setUltFr(nAl);
		matriz = new Fruta().nuevaFruta(1, matriz);
	}
	int m = getMovimientos();
	if(matriz[y - 1][x] != 1) {
		m ++ ;
		setMovimientos(m);
		matriz[y][x] = 0;
		matriz[y - 1][x] = 3;
		setPosY(y - 1);
	}
	else if(y == 1) {
		m ++ ;
		setMovimientos(m);
		matriz[y][x] = 0;
		matriz[matriz.length - 2][x] = 3;
		setPosY(matriz.length - 2);
	}
	return matriz;
}
```

Función “down”

Se encarga de mover a Pac-Man una posición abajo respecto a la posición actual. Si en la posición a la que se moverá Pac-Man hay una fruta (2) se le sumará un valor aleatorio entre 1 y 5 a su punteo. Si en la posición a la que se moverá hay un muro (1) no se efectuará el cambio de posición, a menos que se encuentre en la última fila antes del muro inferior externo y se reubicará en la primera fila después del muro superior externo. Si en la posición a la que se moverá no hay nada (0) o hay una fruta (2) se efectuará el cambio de posición y se sumará 1 a la cantidad de movimientos.

Al finalizar se retornará la matriz mapa con Pac-Man en su nueva posición.

```
public int[][] down(int x,int y,int[][] matriz){
	if(matriz[y + 1][x] == 2) {
		int p = getPuntos();
		int nAl = aleatorio.nextInt(5) + 1;
		p += nAl;
		setPuntos(p);
		setUltFr(nAl);
		matriz = new Fruta().nuevaFruta(1, matriz);
	}
	int m = getMovimientos();
	if(matriz[y + 1][x] != 1) {
		m ++ ;
		setMovimientos(m);
		matriz[y][x] = 0;
		matriz[y + 1][x] = 3;
		setPosY(y + 1);
	}
		else if(y == matriz.length - 2) {
		m ++ ;
		setMovimientos(m);
		matriz[y][x] = 0;
		matriz[1][x] = 3;
		setPosY(1);
	}
	return matriz;
}
```

Función “left”

Se encarga de mover a Pac-Man una posición a la izquierda respecto a la posición actual. Si en la posición a la que se moverá Pac-Man hay una fruta (2) se le sumará un valor aleatorio entre 1 y 5 a su punteo. Si en la posición a la que se moverá hay un muro (1) no se efectuará el cambio de posición, a menos que se encuentre en la primera columna después del muro lateral izquierdo y se reubicará en la última columna antes del muro lateral derecho. Si en la posición a la que se moverá no hay nada (0) o hay una fruta (2) se efectuará el cambio de posición y se sumará 1 a la cantidad de movimientos.

Al finalizar se retornará la matriz mapa con Pac-Man en su nueva posición.

```
public int[][] left(int x,int y,int[][] matriz){
	if(matriz[y][x - 1] == 2) {
		int p = getPuntos();
		int nAl = aleatorio.nextInt(5) + 1;
		p += nAl;
		setPuntos(p);
		setUltFr(nAl);
		matriz = new Fruta().nuevaFruta(1, matriz);
	}
	int m = getMovimientos();
	if(matriz[y][x - 1] != 1) {
		m ++ ;
		setMovimientos(m);
		matriz[y][x] = 0;
		matriz[y][x - 1] = 3;
		setPosX(x - 1);
	}
	else if(x == 1) {
		m ++ ;
		setMovimientos(m);
		matriz[y][x] = 0;
		matriz[y][matriz[0].length - 2] = 3;
		setPosX(matriz[0].length - 2);
	}
	return matriz;
}
```

Función “right”

Se encarga de mover a Pac-Man una posición a la derecha respecto a la posición actual. Si en la posición a la que se moverá Pac-Man hay una fruta (2) se le sumará un valor aleatorio entre 1 y 5 a su punteo. Si en la posición a la que se moverá hay un muro (1) no se efectuará el cambio de posición, a menos que se encuentre en la última columna antes del muro lateral derecho y se reubicará en la primera columna después del muro lateral izquierdo. Si en la posición a la que se moverá no hay nada (0) o hay una fruta (2) se efectuará el cambio de posición y se sumará 1 a la cantidad de movimientos.

Al finalizar se retornará la matriz mapa con Pac-Man en su nueva posición.

```
public int[][] right(int x,int y,int[][] matriz){
	if(matriz[y][x + 1] == 2) {
		int p = getPuntos();
		int nAl = aleatorio.nextInt(5) + 1;
		p += nAl;
		setPuntos(p);
		setUltFr(nAl);
		matriz = new Fruta().nuevaFruta(1, matriz);
	}
	int m = getMovimientos();
	if(matriz[y][x + 1] != 1) {
		m ++ ;
		setMovimientos(m);
		matriz[y][x] = 0;
		matriz[y][x + 1] = 3;
		setPosX(x + 1);
	}
	else if(x == matriz[0].length - 2) {
		m ++ ;
		setMovimientos(m);
		matriz[y][x] = 0;
		matriz[y][1] = 3;
		setPosX(1);
	}
	return matriz;
}
```