Paradigmas Avanzados de Programación

Candy Crush Scala

Grado en Ingeniería Informática
Universidad de Alcalá

Alexander Brun Guajardo
Pablo García García

12 de mayo de 2023

Índice general

Capítulo 1

Implementación básica

Durante este primer capítulo trataremos de explicar el funcionamiento de nuestro juego. El objetivo será imitar al famoso juego *Candy Crush*, pero implementando el paradigma funcional con las herramientas que *Scala* nos ofrece.

1.1. Funcionamiento del juego

Primero veamos como es el juego. Para lanzarlo abriremos con el IDE *IntelliJ* el proyecto que se encuentra en la carpeta `Parte 1`. Dentro de este, podemos editar la configuración (esquina superior derecha) y cambiar los parámetros del juego. Por defecto está en `-m 1 10 10`, pero podemos poner `-m` o `-a` dependiendo de si queremos jugar de manera manual o automática, `1` o `2` para la dificultad del juego, y después las dimensiones del tablero. Si ejecutamos el juego, veremos algo similar a lo que se muestra en la Figura 1.1.

En esta situación podemos escribir primero la fila y después la columna, que harían referencia a la posición del caramelo que queremos "tocar". Hagamos antes un resumen del funcionamiento del juego.

1. Al comenzar una partida, disponemos de **cinco vidas**.
2. Introduciremos la fila y columna del caramelo a eliminar, pero este solo se eliminará con sus contiguos en caso de que los haya.
3. Si no hay contiguos del mismo tipo, no se borrará nada y perderemos una vida.
4. Romper cinco caramelos nos otorgará un objeto especial, **bomba**. Al romper esta, se romperá toda la fila o columna (de forma aleatoria).
5. Romper seis caramelos nos otorgará un objeto especial, **TNT**. Al romper este, se romperán todos los caramelos a distancia 4 de este.
6. Romper más de seis caramelos nos otorgará un objeto especial, **rompecabezas**. Al romper este, se romperán todos los caramelos de un cierto tipo (de forma aleatoria).

```
Modo de ejecucion: manual
Dificultad: 1
Numero de filas: 10
Numero de columnas: 10


VIDAS:

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 4 | 2 | 3 | 2 | 2 | 1 | 1 | 3 | 1 | 3 |
1  | 1 | 3 | 4 | 3 | 2 | 2 | 4 | 4 | 1 | 3 |
2  | 3 | 2 | 4 | 2 | 2 | 1 | 2 | 4 | 2 | 1 |
3  | 1 | 4 | 2 | 4 | 1 | 4 | 4 | 2 | 3 | 3 |
4  | 1 | 4 | 4 | 2 | 3 | 2 | 4 | 4 | 1 | 2 |
5  | 3 | 3 | 4 | 3 | 3 | 3 | 1 | 2 | 3 | 2 |
6  | 1 | 1 | 3 | 4 | 3 | 3 | 1 | 4 | 1 | 2 |
7  | 1 | 4 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8  | 4 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9  | 1 | 1 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |

Introduzca fila:
```

Figura 1.1: Ejecución de ejemplo

Siguiendo con el ejemplo anterior, veamos qué sucede al romper el caramelo en la posición $(1, 7)$. Como tiene más caramelos del mismo tipo alrededor, se romperán este y sus contiguos. Deberán "caer" todos los caramelos, y en las posiciones donde quedarían vacías, se meten nuevos caramelos aleatorios. De la misma manera, si seleccionamos una casilla sin similares contiguos, perderemos una vida.

```
   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 4 | 2 | 3 | 2 | 2 | 1 | 1 | 3 | 1 | 3 |
1  | 1 | 3 | 4 | 3 | 2 | 2 | X | X | 1 | 3 |
2  | 3 | 2 | 4 | 2 | 2 | 1 | 2 | X | 2 | 1 |
3  | 1 | 4 | 2 | 4 | 1 | 4 | 4 | 2 | 3 | 3 |
4  | 1 | 4 | 4 | 2 | 3 | 2 | 4 | 4 | 1 | 2 |
5  | 3 | 3 | 4 | 3 | 3 | 3 | 1 | 2 | 3 | 2 |
6  | 1 | 1 | 3 | 4 | 3 | 3 | 1 | 4 | 1 | 2 |
7  | 1 | 4 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8  | 4 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9  | 1 | 1 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |


   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 4 | 2 | 3 | 2 | 2 | 1 | 4 | 3 | 1 | 3 |
1  | 1 | 3 | 4 | 3 | 2 | 2 | 1 | 1 | 1 | 3 |
2  | 3 | 2 | 4 | 2 | 2 | 1 | 2 | 3 | 2 | 1 |
3  | 1 | 4 | 2 | 4 | 1 | 4 | 4 | 2 | 3 | 3 |
4  | 1 | 4 | 4 | 2 | 3 | 2 | 4 | 4 | 1 | 2 |
5  | 3 | 3 | 4 | 3 | 3 | 3 | 1 | 2 | 3 | 2 |
6  | 1 | 1 | 3 | 4 | 3 | 3 | 1 | 4 | 1 | 2 |
7  | 1 | 4 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8  | 4 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9  | 1 | 1 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |
```

Figura 1.2: Romper caramelos

Veamos ahora ejemplos de ejecución de los diferentes objetos especiales. Comenzamos por la bomba con el ejemplo de la Figura 1.3. En la posición (5, 1) de este ejemplo vemos 5 cuatros juntos, por lo que podremos borrarlos y generar una bomba. Al explotarla, esta ha elegido borrar toda la columna en la que se encuentra.

Siguiendo con el TNT, vemos en el ejemplo de la Figura 1.4, que en la posición (8, 2) vemos seis 3 contiguos, por lo que borramos todos y ponemos en esta posición el TNT, y dejamos caer los caramelos. Al seleccionar el TNT se borran todos los elementos en un radio de 4 elementos a este.

Como último ejemplo de objeto especial, veamos el rompecabezas. En el ejemplo de la Figura 1.5, vemos que en la posición (1, 5) tenemos más de seis 3, por lo que se genera un rompecabezas. Al romper este, elige de forma aleatoria un tipo de caramelo y borra todos estos del tablero, en este caso ha sido el tipo 3.

```
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
 ---------------------------------------------
0 | 2 | 3 | 1 | 2 | 2 | 1 | 1 | 3 | 1 | 3 |
1 | 1 | 4 | 3 | 3 | 2 | 2 | 3 | 1 | 1 | 3 |
2 | 4 | 2 | 4 | 2 | 2 | 1 | 4 | 3 | 2 | 1 |
3 | 1 | 3 | 4 | 4 | 1 | 4 | 1 | 2 | 3 | 3 |
4 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 4 | 1 | 2 |
5 | 1 | 4 | 4 | 3 | 3 | 3 | 4 | 2 | 3 | 2 |
6 | 1 | 4 | 4 | 4 | 3 | 3 | 4 | 4 | 1 | 2 |
7 | 1 | 1 | 1 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8 | 1 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9 | 4 | 4 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |

Introduzca fila: 5
Introduzca columna: 1
Se ha seleccionado la fila 5 y la columna 1
Se han borrado 5 caramelos

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
 ---------------------------------------------
0 | 2 | 3 | 1 | 2 | 2 | 1 | 1 | 3 | 1 | 3 |
1 | 1 | 4 | 3 | 3 | 2 | 2 | 3 | 1 | 1 | 3 |
2 | 4 | 2 | 4 | 2 | 2 | 1 | 4 | 3 | 2 | 1 |
3 | 1 | 3 | 4 | 4 | 1 | 4 | 1 | 2 | 3 | 3 |
4 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 4 | 1 | 2 |
5 | 1 | X | X | 3 | 3 | 3 | 4 | 2 | 3 | 2 |
6 | 1 | X | X | X | 3 | 3 | 4 | 4 | 1 | 2 |
7 | 1 | 1 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8 | 1 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9 | 4 | 4 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |


  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
 ---------------------------------------------
0 | 2 | 3 | 1 | 2 | 2 | 1 | 1 | 3 | 1 | 3 |
1 | 1 | 4 | 3 | 3 | 2 | 2 | 3 | 1 | 1 | 3 |
2 | 4 | 2 | 4 | 2 | 2 | 1 | 4 | 3 | 2 | 1 |
3 | 1 | 3 | 4 | 4 | 1 | 4 | 1 | 2 | 3 | 3 |
4 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 4 | 1 | 2 |
5 | 1 | B | X | 3 | 3 | 3 | 4 | 2 | 3 | 2 |
6 | 1 | X | X | X | 3 | 3 | 4 | 4 | 1 | 2 |
7 | 1 | 1 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8 | 1 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9 | 4 | 4 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |


  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
 ---------------------------------------------
0 | 2 | 2 | 1 | 3 | 2 | 1 | 1 | 3 | 1 | 3 |
1 | 1 | 3 | 1 | 2 | 2 | 2 | 3 | 1 | 1 | 3 |
2 | 4 | 4 | 1 | 3 | 2 | 1 | 4 | 3 | 2 | 1 |
3 | 1 | 2 | 3 | 2 | 1 | 4 | 1 | 2 | 3 | 3 |
4 | 3 | 3 | 4 | 4 | 3 | 2 | 2 | 4 | 1 | 2 |
5 | 1 | 2 | 4 | 2 | 3 | 3 | 4 | 2 | 3 | 2 |
6 | 1 | B | 2 | 3 | 3 | 3 | 4 | 4 | 1 | 2 |
7 | 1 | 1 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8 | 1 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9 | 4 | 4 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |

Introduzca fila: 6
Introduzca columna: 1

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
 ---------------------------------------------
0 | 2 | X | 1 | 3 | 2 | 1 | 1 | 3 | 1 | 3 |
1 | 1 | X | 1 | 2 | 2 | 2 | 3 | 1 | 1 | 3 |
2 | 4 | X | 1 | 3 | 2 | 1 | 4 | 3 | 2 | 1 |
3 | 1 | X | 3 | 2 | 1 | 4 | 1 | 2 | 3 | 3 |
4 | 3 | X | 4 | 4 | 3 | 2 | 2 | 4 | 1 | 2 |
5 | 1 | X | 4 | 2 | 3 | 3 | 4 | 2 | 3 | 2 |
6 | 1 | X | 2 | 3 | 3 | 3 | 4 | 4 | 1 | 2 |
7 | 1 | X | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8 | 1 | X | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9 | 4 | X | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |


  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
 ---------------------------------------------
0 | 2 | 1 | 1 | 3 | 2 | 1 | 1 | 3 | 1 | 3 |
1 | 1 | 2 | 1 | 2 | 2 | 2 | 3 | 1 | 1 | 3 |
2 | 4 | 3 | 1 | 3 | 2 | 1 | 4 | 3 | 2 | 1 |
3 | 1 | 4 | 3 | 2 | 1 | 4 | 1 | 2 | 3 | 3 |
4 | 3 | 2 | 4 | 4 | 3 | 2 | 2 | 4 | 1 | 2 |
5 | 1 | 1 | 4 | 2 | 3 | 3 | 4 | 2 | 3 | 2 |
6 | 1 | 3 | 2 | 3 | 3 | 3 | 4 | 4 | 1 | 2 |
7 | 1 | 1 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8 | 1 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9 | 4 | 1 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |
```

Figura 1.3: Ejemplo explosión bomba

```
   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 2 | 1 | 1 | 3 | 2 | 1 | 1 | 3 | 1 | 3 |
1  | 1 | 2 | 1 | 2 | 2 | 2 | 3 | 1 | 1 | 3 |
2  | 4 | 3 | 1 | 3 | 2 | 1 | 4 | 3 | 2 | 1 |
3  | 1 | 4 | 3 | 2 | 1 | 4 | 1 | 2 | 3 | 3 |
4  | 3 | 2 | 4 | 4 | 3 | 2 | 2 | 4 | 1 | 2 |
5  | 1 | 1 | 4 | 2 | 3 | 3 | 4 | 2 | 3 | 2 |
6  | 1 | 3 | 2 | 3 | 3 | 3 | 4 | 4 | 1 | 2 |
7  | 1 | 1 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8  | 1 | 4 | 3 | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9  | 4 | 1 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |

Introduzca fila: 8
Introduzca columna: 2
Se ha seleccionado la fila 8 y la columna 2
Se han borrado 6 caramelos

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 2 | 1 | 1 | 3 | 2 | 1 | 1 | 3 | 1 | 3 |
1  | 1 | 2 | 1 | 2 | 2 | 2 | 3 | 1 | 1 | 3 |
2  | 4 | 3 | 1 | 3 | 2 | 1 | 4 | 3 | 2 | 1 |
3  | 1 | 4 | 3 | 2 | 1 | 4 | 1 | 2 | 3 | 3 |
4  | 3 | 2 | 4 | 4 | 3 | 2 | 2 | 4 | 1 | 2 |
5  | 1 | 1 | 4 | 2 | 3 | 3 | 4 | 2 | 3 | 2 |
6  | 1 | 3 | 2 | 3 | 3 | 3 | 4 | 4 | 1 | 2 |
7  | 1 | 1 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8  | 1 | 4 | X | X | X | 1 | 3 | 1 | 2 | 4 |
9  | 4 | 1 | 4 | X | X | X | 2 | 1 | 2 | 3 |

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 2 | 1 | 1 | 3 | 2 | 1 | 1 | 3 | 1 | 3 |
1  | 1 | 2 | 1 | 2 | 2 | 2 | 3 | 1 | 1 | 3 |
2  | 4 | 3 | 1 | 3 | 2 | 1 | 4 | 3 | 2 | 1 |
3  | 1 | 4 | 3 | 2 | 1 | 4 | 1 | 2 | 3 | 3 |
4  | 3 | 2 | 4 | 4 | 3 | 2 | 2 | 4 | 1 | 2 |
5  | 1 | 1 | 4 | 2 | 3 | 3 | 4 | 2 | 3 | 2 |
6  | 1 | 3 | 2 | 3 | 3 | 3 | 4 | 4 | 1 | 2 |
7  | 1 | 1 | 1 | 2 | 4 | 1 | 4 | 2 | 3 | 3 |
8  | 1 | 4 | T | X | X | 1 | 3 | 1 | 2 | 4 |
9  | 4 | 1 | 4 | X | X | X | 2 | 1 | 2 | 3 |

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 2 | 1 | 1 | 3 | 4 | 1 | 1 | 3 | 1 | 3 |
1  | 1 | 2 | 1 | 2 | 4 | 1 | 3 | 1 | 1 | 3 |
2  | 4 | 3 | 1 | 3 | 2 | 2 | 4 | 3 | 2 | 1 |
3  | 1 | 4 | 3 | 2 | 2 | 1 | 1 | 2 | 3 | 3 |
4  | 3 | 2 | 4 | 3 | 2 | 4 | 2 | 4 | 1 | 2 |
5  | 1 | 1 | 4 | 2 | 1 | 2 | 4 | 2 | 3 | 2 |
6  | 1 | 3 | 2 | 4 | 3 | 3 | 4 | 4 | 1 | 2 |
7  | 1 | 1 | 1 | 2 | 3 | 3 | 4 | 2 | 3 | 3 |
8  | 1 | 4 | T | 3 | 3 | 1 | 3 | 1 | 2 | 4 |
9  | 4 | 1 | 4 | 2 | 4 | 1 | 2 | 1 | 2 | 3 |

Introduzca fila: 8
Introduzca columna: 2

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 2 | 1 | 1 | 3 | 4 | 1 | 1 | 3 | 1 | 3 |
1  | 1 | 2 | 1 | 2 | 4 | 1 | 3 | 1 | 1 | 3 |
2  | 4 | 3 | 1 | 3 | 2 | 2 | 4 | 3 | 2 | 1 |
3  | 1 | 4 | 3 | 2 | 2 | 1 | 1 | 2 | 3 | 3 |
4  | X | X | X | X | X | X | X | 4 | 1 | 2 |
5  | X | X | X | X | X | X | X | 2 | 3 | 2 |
6  | X | X | X | X | X | X | X | 4 | 1 | 2 |
7  | X | X | X | X | X | X | X | 2 | 3 | 3 |
8  | X | X | X | X | X | X | X | 1 | 2 | 4 |
9  | X | X | X | X | X | X | X | 1 | 2 | 3 |

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
   -----------------------------------------
0  | 1 | 3 | 2 | 3 | 3 | 1 | 2 | 3 | 1 | 3 |
1  | 2 | 2 | 4 | 4 | 2 | 3 | 3 | 1 | 1 | 3 |
2  | 2 | 2 | 3 | 3 | 3 | 3 | 1 | 3 | 2 | 1 |
3  | 2 | 3 | 3 | 3 | 2 | 1 | 3 | 2 | 3 | 3 |
4  | 2 | 3 | 1 | 1 | 3 | 2 | 4 | 4 | 1 | 2 |
5  | 4 | 4 | 3 | 1 | 3 | 4 | 2 | 2 | 3 | 2 |
6  | 2 | 1 | 1 | 3 | 4 | 1 | 1 | 4 | 1 | 2 |
7  | 1 | 2 | 1 | 2 | 4 | 1 | 3 | 2 | 3 | 3 |
8  | 4 | 3 | 1 | 3 | 2 | 2 | 4 | 1 | 2 | 4 |
9  | 1 | 4 | 3 | 2 | 2 | 1 | 1 | 1 | 2 | 3 |
```

Figura 1.4: Ejemplo explosión TNT

```
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
-----------------------------------------
0 | 1 | 3 | 2 | 3 | 3 | 1 | 2 | 3 | 1 | 3 |
1 | 2 | 2 | 4 | 4 | 2 | 3 | 3 | 1 | 1 | 3 |
2 | 2 | 2 | 3 | 3 | 3 | 3 | 1 | 3 | 2 | 1 |
3 | 2 | 3 | 3 | 3 | 2 | 1 | 3 | 2 | 3 | 3 |
4 | 2 | 3 | 1 | 1 | 3 | 2 | 4 | 4 | 1 | 2 |
5 | 4 | 4 | 3 | 1 | 3 | 4 | 2 | 2 | 3 | 2 |
6 | 2 | 1 | 1 | 3 | 4 | 1 | 1 | 4 | 1 | 2 |
7 | 1 | 2 | 1 | 2 | 4 | 1 | 3 | 2 | 3 | 3 |
8 | 4 | 3 | 1 | 3 | 2 | 2 | 4 | 1 | 2 | 4 |
9 | 1 | 4 | 3 | 2 | 2 | 1 | 1 | 1 | 2 | 3 |

Introduzca fila: 1
Introduzca columna: 5
Se ha seleccionado la fila 1 y la columna 5
Se han borrado 10 caramelos

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
-----------------------------------------
0 | 1 | 3 | 2 | 3 | 3 | 1 | 2 | 3 | 1 | 3 |
1 | 2 | 2 | 4 | 4 | 2 | X | X | 1 | 1 | 3 |
2 | 2 | 2 | X | X | X | X | 1 | 3 | 2 | 1 |
3 | 2 | X | X | X | 2 | 1 | 3 | 2 | 3 | 3 |
4 | 2 | X | 1 | 1 | 3 | 2 | 4 | 4 | 1 | 2 |
5 | 4 | 4 | 3 | 1 | 3 | 4 | 2 | 2 | 3 | 2 |
6 | 2 | 1 | 1 | 3 | 4 | 1 | 1 | 4 | 1 | 2 |
7 | 1 | 2 | 1 | 2 | 4 | 1 | 3 | 2 | 3 | 3 |
8 | 4 | 3 | 1 | 3 | 2 | 2 | 4 | 1 | 2 | 4 |
9 | 1 | 4 | 3 | 2 | 2 | 1 | 1 | 1 | 2 | 3 |

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
-----------------------------------------
0 | 1 | 3 | 2 | 3 | 3 | 1 | 2 | 3 | 1 | 3 |
1 | 2 | 2 | 4 | 4 | 2 | R | X | 1 | 1 | 3 |
2 | 2 | 2 | X | X | X | X | 1 | 3 | 2 | 1 |
3 | 2 | X | X | X | 2 | 1 | 3 | 2 | 3 | 3 |
4 | 2 | X | 1 | 1 | 3 | 2 | 4 | 4 | 1 | 2 |
5 | 4 | 4 | 3 | 1 | 3 | 4 | 2 | 2 | 3 | 2 |
6 | 2 | 1 | 1 | 3 | 4 | 1 | 1 | 4 | 1 | 2 |
7 | 1 | 2 | 1 | 2 | 4 | 1 | 3 | 2 | 3 | 3 |
8 | 4 | 3 | 1 | 3 | 2 | 2 | 4 | 1 | 2 | 4 |
9 | 1 | 4 | 3 | 2 | 2 | 1 | 1 | 1 | 2 | 3 |

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
-----------------------------------------
0 | 1 | 3 | 3 | 4 | 4 | 3 | 1 | 3 | 1 | 3 |
1 | 2 | 3 | 2 | 3 | 3 | 1 | 2 | 1 | 1 | 3 |
2 | 2 | 3 | 2 | 3 | 2 | R | 1 | 3 | 2 | 1 |
3 | 2 | 2 | 4 | 4 | 2 | 1 | 3 | 2 | 3 | 3 |
4 | 2 | 2 | 1 | 1 | 3 | 2 | 4 | 4 | 1 | 2 |
5 | 4 | 4 | 3 | 1 | 3 | 4 | 2 | 2 | 3 | 2 |
6 | 2 | 1 | 1 | 3 | 4 | 1 | 1 | 4 | 1 | 2 |
7 | 1 | 2 | 1 | 2 | 4 | 1 | 3 | 2 | 3 | 3 |
8 | 4 | 3 | 1 | 3 | 2 | 2 | 4 | 1 | 2 | 4 |
9 | 1 | 4 | 3 | 2 | 2 | 1 | 1 | 1 | 2 | 3 |

Introduzca fila: 2
Introduzca columna: 5

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
-----------------------------------------
0 | 1 | X | X | 4 | 4 | X | 1 | X | 1 | X |
1 | 2 | X | 2 | X | X | 1 | 2 | 1 | 1 | X |
2 | 2 | X | 2 | X | 2 | X | 1 | X | 2 | 1 |
3 | 2 | 2 | 4 | 4 | 2 | 1 | X | 2 | X | X |
4 | 2 | 2 | 1 | 1 | X | 2 | 4 | 4 | 1 | 2 |
5 | 4 | 4 | X | 1 | X | 4 | 2 | 2 | X | 2 |
6 | 2 | 1 | 1 | X | 4 | 1 | 1 | 4 | 1 | 2 |
7 | 1 | 2 | 1 | 2 | 4 | 1 | X | 2 | X | X |
8 | 4 | X | 1 | X | 2 | 2 | 4 | 1 | 2 | 4 |
9 | 1 | 4 | X | 2 | 2 | 1 | 1 | 1 | 2 | X |

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
-----------------------------------------
0 | 1 | 1 | 2 | 2 | 4 | 3 | 3 | 3 | 2 | 2 |
1 | 2 | 3 | 2 | 2 | 4 | 1 | 1 | 3 | 2 | 3 |
2 | 2 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 4 | 2 |
3 | 2 | 3 | 2 | 1 | 4 | 1 | 2 | 2 | 1 | 2 |
4 | 2 | 2 | 2 | 4 | 2 | 2 | 1 | 4 | 1 | 1 |
5 | 4 | 2 | 4 | 4 | 2 | 4 | 4 | 2 | 2 | 1 |
6 | 2 | 4 | 1 | 1 | 4 | 1 | 2 | 4 | 1 | 2 |
7 | 1 | 1 | 1 | 1 | 4 | 1 | 1 | 2 | 1 | 2 |
8 | 4 | 2 | 1 | 2 | 2 | 2 | 4 | 1 | 2 | 2 |
9 | 1 | 4 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 4 |
```

Figura 1.5: Ejemplo explosión rompecabezas

Como último ejemplo de ejecución, veamos el final de cómo la máquina juega sola, que para cuando pierde, que suele ser rápido, pues al elegir las posiciones de forma aleatoria, es fácil elegir aquellas que no tienen contiguos iguales.

```
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
  -----------------------------------------
0 | 1 | 3 | 1 | 2 | 2 | 3 | 1 | 3 | 1 | 1 |
1 | 4 | 2 | 3 | 3 | 4 | 2 | 4 | 3 | 3 | 1 |
2 | 4 | 1 | 2 | 1 | 4 | 2 | 3 | 4 | 3 | 3 |
3 | 1 | 2 | 2 | 1 | 4 | 1 | 1 | 4 | 2 | 1 |
4 | 2 | 3 | 4 | 4 | 1 | 3 | 4 | 1 | 1 | 4 |
5 | 4 | 1 | 4 | 1 | 4 | 1 | 4 | 4 | 1 | 2 |
6 | 3 | 4 | 1 | 3 | 2 | 2 | 4 | T | 2 | 1 |
7 | 2 | 2 | 3 | 2 | 2 | 1 | 3 | 2 | 3 | 2 |
8 | 3 | 1 | 1 | 4 | 3 | 2 | 4 | 2 | 1 | 1 |
9 | 2 | 4 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 1 |

Se ha seleccionado la fila 9 y la columna 4
Se han borrado 0 caramelos

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
  -----------------------------------------
0 | 1 | 3 | 1 | 2 | 2 | 3 | 1 | 3 | 1 | 1 |
1 | 4 | 2 | 3 | 3 | 4 | 2 | 4 | 3 | 3 | 1 |
2 | 4 | 1 | 2 | 1 | 4 | 2 | 3 | 4 | 3 | 3 |
3 | 1 | 2 | 2 | 1 | 4 | 1 | 1 | 4 | 2 | 1 |
4 | 2 | 3 | 4 | 4 | 1 | 3 | 4 | 1 | 1 | 4 |
5 | 4 | 1 | 4 | 1 | 4 | 1 | 4 | 4 | 1 | 2 |
6 | 3 | 4 | 1 | 3 | 2 | 2 | 4 | T | 2 | 1 |
7 | 2 | 2 | 3 | 2 | 2 | 1 | 3 | 2 | 3 | 2 |
8 | 3 | 1 | 1 | 4 | 3 | 2 | 4 | 2 | 1 | 1 |
9 | 2 | 4 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 1 |

Has perdido una vida!

  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
  -----------------------------------------
0 | 1 | 3 | 1 | 2 | 2 | 3 | 1 | 3 | 1 | 1 |
1 | 4 | 2 | 3 | 3 | 4 | 2 | 4 | 3 | 3 | 1 |
2 | 4 | 1 | 2 | 1 | 4 | 2 | 3 | 4 | 3 | 3 |
3 | 1 | 2 | 2 | 1 | 4 | 1 | 1 | 4 | 2 | 1 |
4 | 2 | 3 | 4 | 4 | 1 | 3 | 4 | 1 | 1 | 4 |
5 | 4 | 1 | 4 | 1 | 4 | 1 | 4 | 4 | 1 | 2 |
6 | 3 | 4 | 1 | 3 | 2 | 2 | 4 | T | 2 | 1 |
7 | 2 | 2 | 3 | 2 | 2 | 1 | 3 | 2 | 3 | 2 |
8 | 3 | 1 | 1 | 4 | 3 | 2 | 4 | 2 | 1 | 1 |
9 | 2 | 4 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 1 |

Fin del juego, te has quedado sin vidas!
```

Figura 1.6: Ejemplo fin de partida

1.2. Implementación

1.2.1. Función principal

Una vez demostrado el correcto funcionamiento del juego, abordaremos cómo funciona el código de la implementación realizada. En primer lugar comentaremos cómo funciona el "bucle" (pues este es recursivo) principal del juego.

Para esto nos vamos al fichero `Juego.scala` donde desde la función `iniciar(): Unit` generamos el tablero, y nos vamos a nuestro "bucle", indicando por parámetro el tablero y el número de vidas iniciales (5). Dentro de esta función, si el número de vidas es 0, la partida ha de finalizar. En caso contrario, se pide al usuario indicar una casilla. Si el caramelo de dicha casilla no es especial, se busca el camino, se rompen los caramelos, se recoloca el tablero, y

se vuelve a llamar a nuestro bucle con el nuevo tablero, y el mismo número de vidas en caso de que haya existido camino, y $n-1$, en caso de que no. Si el caramelo es especial, se aplica su efecto, se recoloca, y se llama al bucle con el nuevo tablero y el mismo número de vidas.

Todos estos métodos que estamos nombrando, son todos recursivos, y son los que se comentarán en la Sección 1.2.3.

1.2.2. Métodos auxiliares

Ya mostrado cómo funciona el flujo de ejecución del juego, comentaremos una serie de funciones auxiliares que hemos usado y que se encuentran en `Operaciones.scala`, siendo todas ellas recursivas:

- `longitud(array: List[Any]): Int`: calcula la longitud de una lista.
- `concatenar(array1: List[Char], array2: List[Char]): List[Char]`: concatena dos subtableros.
- `menor(lista: List[Int]): Int`: encuentra el menor entero de una lista.
- `union(l1: List[Int], l2: List[Int]): List[Int]`: dadas dos listas de enteros, calcula la unión conjuntista.
- `calcularDificultad(elems: Int): Int`: calcula la dificultad en función del número de elementos diferentes de un tablero.
- `generarElemento(dificultad: Int): Char`: genera un caramelo de forma aleatoria, en función de la dificultad.
- `boolToInt(booleano: Boolean): Int`: si recibe `true` devuelve 1, y 0 en aso contrario.
- `obtenerPosicionLista(tablero: List[Char], i: Int): Char`: devuelve la posición de `i` en caso de que $i \in T$.
- `contiene(lista: List[Int], elemento: Int): Boolean`: determina si un elemento pertenece a una lista.
- `reemplazarPosicion(lista: List[Char], pos: Int, valor: Char): List[Char]`: dado, un caramelo, una posición, y un tablero, lo reemplaza en dicha posición.

1.2.3. Operaciones con el tablero

Una vez explicadas funciones de "*utilidades genéricas*", pasamos a explicar funciones que han sido expresamente diseñadas para trabajar con el tablero del juego:

- `obtenerCaramelo(tablero: List[Char], fila: Int, col: Int): Char`: dada una fila, y una columna de un tablero, devuelve el valor del caramelo que se encuentra en dicha posición.

- `contarX(tablero: List[Char]): Int`: cuenta los huecos vacíos en un tablero.
- `generarTablero(i: Int, dif: Int, f: Int =>Char): List[Char]`: dado un número de elementos, una dificultad, y una función generadora de caramelos, construye el tablero.
- `mostrarTableroAux(tablero: List[Char], i: Int, m: Int, mFila: Boolean): String`, `cabeceraTablero(m: Int, inicio: Int, total: Int): String`, `mostrarTablero(tablero: List[Char], vidas: Int, m: Int): String`, e `imprimirCaramelo(caramelo: Int): String`: se encargan de la impresión por pantalla del tablero y vidas que hemos visto en los ejemplos previos de ejecución.
- analizarVecinos(vecinos: List[Int], selec: Int, i: Int): List[Int]: dado un elemento y una lista de sus vecinos, comprueba de forma recursiva si estos no "se salen" por encima, debajo, izquierda, o derecha, de acuerdo a las siguientes reglas:
  - **Encima**: si estamos en la posición i, y tenemos m columnas, entonces debe cumplirse que $i - m > 0$.
  - **Debajo**: si estamos en la posición i, y tenemos n filas y m columnas, entonces $i + m < nm$.
  - **Izquierda**: si estamos en la posición i, entonces i no puede ser múltiplo de m, por lo que debe cumplirse que $i \pmod m \not\equiv 0$.
  - **Derecha**: si estamos en la posición i, entonces si nos desplazamos una posición, no puede ser múltiplo del número de columnas, por lo que debe cumplirse que $i + 1 \pmod m \not\equiv 0$.
- `explorarVecinos(tablero: List[Char], camino: List[Int], visitados: List[Int], vecinos: List[Int], selec: Int): List[Int]` y `buscarCamino(tablero: List[Char], selec: Int): List[Int]`: implementan un algoritmo de *backtracking* de manera que desde el elemento que ha sido seleccionado, se calculan sus posibles vecinos, y en caso de compartir tipo, se añaden a un camino y se aplica recursivamente este algoritmo a cada uno de ellos. Finalmente devuelve una lista con los índices del camino.
- `marcar(tablero: List[Char], camino: List[Int],n:Int): List[Char]`: dado un tablero y un camino, borra los caramelos que el camino indica, dejando una marca `X`.
- `contarXdebajo(tablero: List[Char], elem: Int): Int`, `contarNoXencima(tablero: List[Char], elem: Int): Int`, `recolocarTableroAux(tablero: List[Char], elem: Int): List[List[Int]]`, `dejarDefinitivos(actualizaciones: List[List[Int]], actualizacionesDefinitivas: List[List[Int]]): List[List[Int]]`, `cambiosTablero(tablero: List[Char]): List[List[Int]]`, `actualizarTablero(nTablero: List[Char], actualizaciones: List[List[Int]]): List[Char]`: son funciones auxiliares que van llamándose unas a otras para poder implementar de forma recursiva la caída de los caramelos. Se calcula a dónde deberá caer cada caramelo y si tendrán que entrar nuevos por encima, en función de cuántos huecos hay debajo de él, y cuántos caramelos quedan encima. Se generará una lista de actualizaciones, donde una actualización es una lista de dos elementos, con posición y tipo de caramelo a colocar. Se aplica cada actualización sobre el tablero.

- `reemplazarCaramelo(tablero: List[Char], seleccionado: Int, borrados: Int, pos: Int): List[Char]`: si al tocar una posición se borran suficientes caramelos como para que se genere un elemento especial, se coloca.
- `bloqueBomba(tablero: List[Char], elem: Int): List[Char]`: aplica el efecto de la bomba. Al explotar elige si borra fila o columna.
- `bloqueTNT(tablero: List[Char], elem: Int): List[Char]`: aplica el efecto del TNT. Al explotar borra todos los elementos en un radio de 4.
- `bloqueRompecabezas(tablero: List[Char], elem: Int, dif: Int): List[Char]`: aplica el efecto del rompecabezas. Al tocarlo, elige al azar un tipo de caramelo y borra todos esos.

Capítulo 2

Implementación optimizada

Para esta segunda parte de la práctica, se ha solicitado que en el modo automático, en vez de elegirse una posición aleatoria, se elija aquella que se prevé que borre un mayor número de caramelos. De forma resumida, lo que se hace es devolver el índice del valor máximo de una lista, donde cada elemento $\mathcal{B}_i$ de esa lista representa la cantidad de caramelos que borra el caramelo i del tablero. Para generar dicha lista, debemos distinguir dos casos:

- El elemento es un **caramelo no especial** o **TNT**: añadimos a nuestra lista la longitud del camino desde esa posición; o si es un TNT, sobre un tablero auxiliar aplicamos su efecto, y contamos el número de `X` que se generan.
- El elemento es un **caramelo especial**: hasta que no se produce el efecto, no se sabe cuántos caramelos serán borrados, pero sí conocemos un rango de valores.
    - **Rompecabezas**: supongamos que tenemos n tipos de caramelos, y que del tipo $\mathcal{C}_i$ existen más que de ningún otro. En este caso, $\mathcal{B}_i = \text{mín}\{|\mathcal{C}_1|, |\mathcal{C}_2|, \cdots, |\mathcal{C}_n|\}$. Esto se debe a que la probabilidad de romper el tipo $\mathcal{C}_i$ con el rompecabezas es $\frac{1}{n}$, o dicho de otra forma, existiría una probabilidad de $\frac{n-1}{n}$ de tras elegir un rompecabezas (si este marcase siempre el $\mathcal{C}_i$), borrar menos caramelos de los que podrían borrarse.
    - **Bomba**: en el momento de explotar, borrará una fila o columna de nuestro tablero de dimensiones $n \times m$, y por el mismo motivo, nos pondremos en el peor de los casos, y asignaremos $\mathcal{B}_i = \text{mín}\{n, m\}$.

Todo esto se implementa también mediante funciones recursivas que se pueden encontrar al final del fichero `Operaciones.scala` del segundo proyecto, quedando la "llamada principal" desde la función `mejorCamino(tablero: List[Char]): Int`, donde ese `Int` que devuelve es $\underset{i}{\text{argmáx}}\, \mathcal{B}_i$.

Capítulo 3

Interfaz gráfica

En esta tercera parte de la práctica, se solicita llevar a cabo una interfaz gráfica para lo desarrollado en la segunda parte. Para llevar esto a cabo, por un lado se realizan todas las ventanas y componentes de la interfaz, y por el otro la conexión con la lógica del juego.

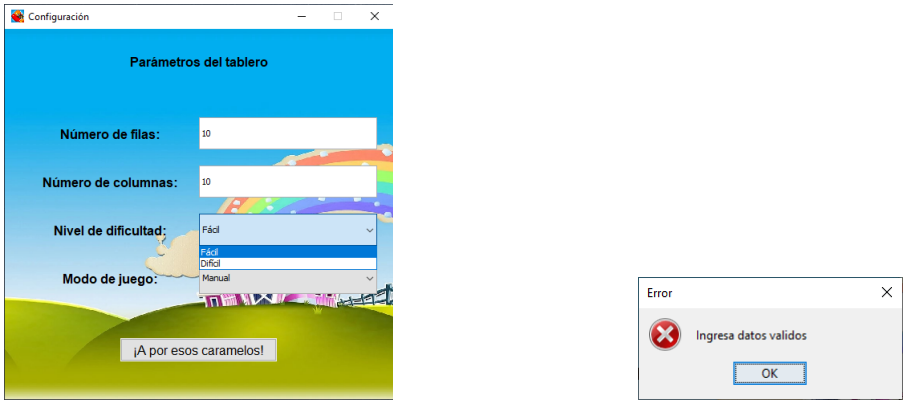
3.1. Ventanas y componentes

Primero veamos cómo son las ventanas y sus distintos componentes. Se cuenta con un total de tres ventanas, una para la portada, otra para la obtención de los distintos parámetros del juego y una tercera ventana con lo que sería el juego.

- **Scala Candy Crush:** esta es la ventana inicial que se muestra al iniciar el juego. Tiene una imagen de fondo además de un botón "`JUGAR`". La funcionalidad de este es muy simple, ya que lo único que hace es llevarnos a la siguiente ventana.

Figura 3.1: Ventana de bienvenida

- **Configuración:** en esta segunda ventana nos vamos a encontrar en primer lugar con un título indicando que en esta ventana se recogerán los distintos parámetros del juego, los cuales son `número de filas`, `número de columnas`, `nivel de dificultad` y `modo de juego`. Además de esto, se encuentra el botón `¡A por esos caramelos!`, que al pulsarlo, verificará que todos los campos hayan sido completados, para posteriormente recoger dichos datos y enviarlos a la ventana del juego, pero si hay algún tipo de error con los datos, se nos mostrará un mensaje de advertencia. Cabe destacar que cuando se llega a esta ventana, empieza a sonar una música de fondo, de forma que la experiencia sea más inmersiva.



(a) Ventana (b) Error

Figura 3.2: Recogida de parámetros

- **Candy Crush:** esta es la ventana donde se va a llevar a cabo toda la partida (Figura 3.3). La ventana se divide en 3 secciones, la primera de ellas la que contiene el propio tablero con los distintos caramalos, los cuales tienen una imagen para cada tipo, incluyendo los especiales. La segunda nos muestra las vidas representadas mediante 5 corazones que inicialmente están rojos, y a medida que se falla, dichos corazones se vuelven blancos, simulando la pérdida de vidas. Con respecto a la tercera sección, esta es la que contiene el sistema de puntuación, el cual va aumentando en cada jugada. Cabe destacar que siempre que se seleccione un caramelo, se escucharán distintos sonidos, imitando así la versión original de juego.

3.2. Conexión con la lógica

Para poder llevar a cabo la conexión entre lógica e interfaz, han sido necesarias varias modificaciones con respecto a la segunda parte de la práctica.

- **Cambio de elementos a `var`**: se han modificado el tipo del tablero y de las vidas, además de haberse declarado los puntos de esta manera, para así poder llevar a cabo un correcto funcionamiento y actualización de elementos sin la necesidad de una constante creación de ventanas con los elementos nuevos, de forma que no se sature la memoria.

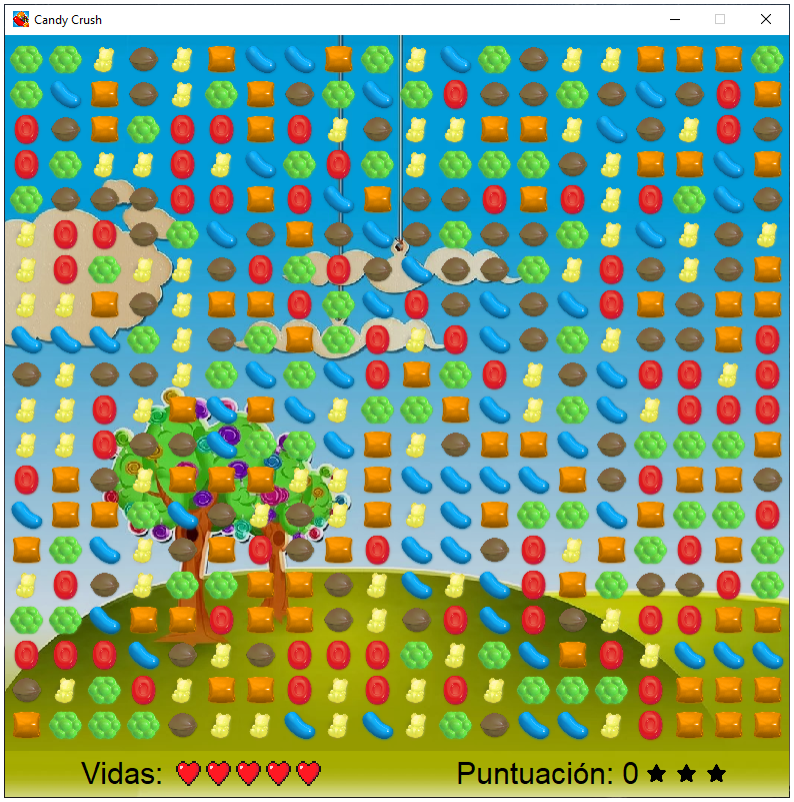


Figura 3.3: Tablero del juego

- **Creación método `recargarComponentes()`**: este es el método al que se irá llamando a medida que se interactúe con nuestro tablero, ya que incluirá la llamada a los métodos encargados de la actualización de las vidas, de los puntos y del propio tablero en sí, para terminar pintando nuevamente la ventana con los nuevos valores actualizados.

- **Métodos de actualización**: estos métodos son principalmente tres, y como hemos comentado arriba, sirven para la correcta actualización de las vidas, puntos y tablero en cada una de las distintas jugadas que se lleven a cabo a lo largo de la partida.

  - `actualizarTablero(tablero)`: aquí lo único que se hace es de forma recursiva, recorrer el tablero actualizando las imágenes de cada uno de los caramelos.
  - `actualizarVidas(vidasJuego)`: como hemos comentado anteriormente, las vidas inicialmente se establecen como 5 corazones rojos. Con respecto a su actualización, lo que se hace es recorrer los corazones y cambiar el último de ellos por un corazón vacío en caso de fallar.
  - `actualizarPuntos(puntos)`: los puntos funcionan de forma similar a las vidas, inicialmente se establece como 0, además de la inicialización de las estrellas, que inicialmente se representan mediante estrellas vacías. Con respecto a sus actualizaciones, los puntos aumentan en cantidades de 150 por cada caramelo eliminado y se van acumulando haciendo así que las estrellas se iluminen a medida que alcanzamos distintas cantidades de puntos, como 15.000, 30.000 y 45.000 puntos

para completar la tercera estrella. Además de eso, se ha añadido como guiño al juego original la posibilidad de modificar el aspecto amarillo de las estrellas por uno azul al alcanzar los 100000 puntos.

- **Creación método `modoAutomatico()` y clase `HiloJuego`:** comenzando con el método del modo automático, este es activado cuando en la ventana de selección de los parámetros se selecciona dicho modo, ya que se activa la clase "`HiloJuego`", la cual llama a dicho método. Funciona de forma similar a la parte 2, solo que el hilo que lo ejecuta queda bloqueado durante la ejecución de dicho modo, por lo que ha sido necesaria la creación de la clase mencionada anteriormente, de forma que lo que hace es simplemente ir llamando cada un intervalo de dos segundos al método, para que así este obtenga la posición óptima en cada jugada.