PROYECTO PROGRAMACIÓN TEMA 4

JUGANDO AL BUSCAMINAS (315)

En este proyecto se nos pide realizar un buscaminas “funcional” que no pueda tener más de 50 filas/columna. En la entrada del programa debemos empezar introduciendo las filas y columnas (por ejemplo: 8 8), después, debemos de introducir manualmente nuestro tablero del buscaminas, usando “*” para indicar que hay una bomba y “-“ para indicar que no hay nada.

Después de tener nuestro buscaminas, debemos de introducir las veces que queremos buscar en una celda, y procedemos a buscar celdas hasta que hayamos llegado a las veces que dijimos. El índice empieza en 1-1, no en 0-0.

Ejemplo de entrada:

```
8 8
-*---*--
--------
**------
---*----
---*---*
-*--*---
----*---
--------
4
1 1
1 8
8 8
8 1
```

En la salida, nos deberá de salir el campo de minas después de haber seleccionado las celdas que quisimos: Si una celda seleccionada tenía bombas alrededor, esa celda tendrá el valor de las bombas que tenga a su alrededor. Si la celda seleccionada es una bomba, la salida será un “GAME OVER”, y si la celda estaba vacía y sin bombas alrededor, se procederá a hacer visibles las celdas de alrededor hasta bordear todas las bombas posibles. Aquí un ejemplo de salida después de haber seleccionado las celdas 1-1,1-8,8-8,8-1:

```
1XXXXX1-
XXXX111-
XXXX1---
XXXX2-11
XXXX311X
XXXXX211
1112X2--
---1X1--
```

Código versión juez:

```java
package org.example;

import java.util.InputMismatchException;
import java.util.Random;
import java.util.Scanner;

public class BuscaminasJuez {

    public static String[][] matrizReal;


    static int numFilas = 0;
    static int numColumnas = 0;
    static int celdasPorDescubrir = 0;
    static String[][] matrizVisible;
    static Scanner read = new Scanner(System.in);
    static Random aleatorio = new Random();


    public static void main(String[] args) {
        numFilas = read.nextInt();
        numColumnas = read.nextInt();


        matrizReal = new String[numFilas][numColumnas];
        rellenarMatriz();

        matrizVisible = crearMatrizVisible();

        celdasPorDescubrir = read.nextInt();
        read.nextLine();
        pedirCeldas(celdasPorDescubrir);

    }



    public static void rellenarMatriz () {

        int filaActual = 0;
        externo:
        for (int i = 0; i < matrizReal.length; i++) {
            String valoresFilaActual[] = read.next().split("");

            for (int j = 0; j < matrizReal[0].length; j++) {
                matrizReal[i][j] = valoresFilaActual[j];
            }
            filaActual++;
        }


    }

    public static void imprimirMatriz () {
        System.out.println();
        for (int i = 0; i < matrizReal.length; i++) {
            for (int j = 0; j < matrizReal[0].length; j++) {
                System.out.print(matrizReal[i][j]);
            }
            System.out.println();
        }
    }


    public static String[][] crearMatrizVisible() {
        String[][] matrizVisible = new String[numFilas][numColumnas];

        for (int i = 0; i < matrizVisible.length; i++) {
            for (int j = 0; j < matrizVisible[0].length; j++) {
                matrizVisible[i][j] = "X";
            }
        }

        return matrizVisible;
    }

    public static void pedirCeldas (int celdasPorDescubrir) {



        for (int i = 0; i < celdasPorDescubrir; i++) {
            String[] celdaSeleccionada = read.nextLine().split(" ");
            int fila = Integer.parseInt(celdaSeleccionada[0])-1;
            int columna = Integer.parseInt(celdaSeleccionada[1])-1;
            asignarValorCelda(fila, columna);

            if (!comprobarGameOver(celdaSeleccionada)) {
                System.out.println("GAME OVER");
                break;
            }

        }

        imprimirMatrizVisible(matrizVisible);
    }

    public static boolean comprobarGameOver  (String[] celdaSeleccionada) {
        int fila = Integer.parseInt(celdaSeleccionada[0]) - 1;
        int columna = Integer.parseInt(celdaSeleccionada[1]) - 1;

        if (matrizReal[fila][columna].equals("*"))  {
            return false;
        }

        return true;
    }


    public static void asignarValorCelda(int fila, int columna) {

        int maximo = numFilas*numColumnas;

        int[] filasPendientes = new int[maximo];
        int[] columnasPendientes = new int[maximo];

        int celdasPendientes = 1;

        filasPendientes[0] = fila;
        columnasPendientes[0] = columna;

        if (matrizReal[fila][columna].equals("*")) matrizVisible[fila][columna] = "*";
        else {

            for(int indice = 0; indice < celdasPendientes; indice++ ) {


                int filaActual = filasPendientes[indice];
                int columnaActual = columnasPendientes[indice];

                int contadorBombas = 0;


                for (int i = filaActual -1; i <= filaActual +1; i++) {
                    for (int j = columnaActual -1; j <= columnaActual +1 ; j++) {

                        if (i < 0 || i >= numFilas || j < 0 || j >= numColumnas) continue;

                        if (matrizReal[i][j].equals("*")) contadorBombas++;

                    }
                }

                if (contadorBombas != 0) {
                    matrizVisible[filaActual][columnaActual] = Integer.toString(contadorBombas);
                } else {

                    matrizVisible[filaActual][columnaActual] = "-";

                    for (int i = filaActual -1; i <= filaActual +1; i++) {
                        for (int j = columnaActual -1; j <= columnaActual +1 ; j++) {

                            if (i < 0 || i >= numFilas || j < 0 || j >= numColumnas) continue;

                            if (matrizVisible[i][j].equals("X")) {
                                filasPendientes[celdasPendientes] = i;
                                columnasPendientes[celdasPendientes] = j;
                                celdasPendientes++;
                            }

                        }
                    }


                }

            }
        }

    }

    public static void imprimirMatrizVisible (String[][] matriz) {
        for(String fila[] : matriz) {
            for (String valor : fila) {
                System.out.print(valor);
            }
            System.out.println();
        }
    }

}
```

Código versión “bonita”:

```java
package org.example;

import java.util.InputMismatchException;
import java.util.Random;
import java.util.Scanner;

/**
 * Esta clase contiene los métodos utilizados para poder realizar el ejercicio 315 de Acepta el reto
 * @author david
 * @version 1.0
 */
public class Buscaminas {

    public static String[][] matrizReal;


    static int numFilas = 0;
    static int numColumnas = 0;
    static int celdasPorDescubrir = 0;
    static String[][] matrizVisible;
    static Scanner read = new Scanner(System.in);
    static Random aleatorio = new Random();


    static void main(String[] args) {

        numFilas = pedirEntero("Introduce el número de filas: ");
        numColumnas = pedirEntero("Introduce el número de columnas: ");
        read.nextLine();

        if (numFilas > 50 || numColumnas > 50) {
            System.out.println("No puede haber más de 50 filas ni 50 columnas");
            return;
        }

        matrizReal = new String[numFilas][numColumnas];
        rellenarMatriz();

        imprimirMatriz(matrizReal);

        matrizVisible = crearMatrizVisible();

        celdasPorDescubrir = pedirEntero("Introduce el número de celdas a descubrir: ");
        read.nextLine();
        pedirCeldas(celdasPorDescubrir);

    }
    /**
     * Pide al usuario un número entero por consola y comprueba que el número
     * introducido sea válido.
     * @param texto El mensaje que se muestra al usuario para pedir el número.
     * @return El número entero introducido por el usuario.
     */
    public static int pedirEntero(String texto) {
        Scanner read = new Scanner(System.in);

        int num = 0;
        boolean seguir = true;


        do {
            System.out.print(texto);
            try {
                num = read.nextInt();
                seguir = false;
            } catch (InputMismatchException e) {
                System.out.println("Error. Debes introducir un número entero");
                read.nextLine();
            }
        } while (seguir);

        return num;
    }


    /**
     * Este método sirve para rellenar la matriz del buscaminas (todas las celdas)
     * y comprobar que lo que introduce el usuario es correcto
     */
    public static void rellenarMatriz () {

        int filaActual = 0;
        externo:
        for (int i = 0; i < matrizReal.length; i++) {
            System.out.println("Introduce los " + numColumnas + " carácteres de la fila " + filaActual);
            String valoresFilaActual[] = read.next().split("");

            if (valoresFilaActual.length != numColumnas) {
                i--;
                System.out.println("La fila debe de tener " + numColumnas + " carácteres.");
                continue;
            }
            for (int j = 0; j < matrizReal[0].length; j++) {
                if (!valoresFilaActual[j].equals("*") && !valoresFilaActual[j].equals("-")) {
                    System.out.println("La fila debe de tener solo '-' o '*'. Cerrando el programa.");
                    i--;
                    continue externo;
                }
                matrizReal[i][j] = valoresFilaActual[j];
            }
            filaActual++;
        }


    }

    /**
     *
     * Este método sirve para mostrar en consola la matriz que introduzca el usuario
     * @param matriz La matriz que el usuario desea que se muestre por consola
     *
     */
    public static void imprimirMatriz (String[][] matriz) {
        for(String fila[] : matriz) {
            for (String valor : fila) {
                System.out.print(valor);
            }
            System.out.println();
        }
    }

    /**
     *
     * Esta función sirve para crear la matriz visible dentro del mismo método, y devolver la matriz resultante,
     * que debe de ser con todas las celdas teniendo un valor de "X". La matriz se crea a partir del numFilas y
     * numColumnas que ha introducido el usuario anteriormente.
     *
     * @return Devuelve la matriz visible (osea, la matriz que empieza siendo todo "X")
     */
    public static String[][] crearMatrizVisible() {
        String[][] matrizVisible = new String[numFilas][numColumnas];

        for (int i = 0; i < matrizVisible.length; i++) {
            for (int j = 0; j < matrizVisible[0].length; j++) {
                matrizVisible[i][j] = "X";
            }
        }

        return matrizVisible;
    }


    /**
     * Este método se utiliza para pedir al usuario las celdas que quiere descubrir hasta completar
     * la cantidad indicada por celdasPorDescubrir. Por cada celda seleccionada, se guardan los valores
     * para utilizarlos posteriormente en el método asignarValorCelda. Al final, se imprime la matriz
     * resultante y, si la celda contiene una bomba (usando el booleano que devuelve el método
     * comprobarGameOver), se muestra "GAME OVER".
     *
     * @param celdasPorDescubrir La cantidad de celdas que el usuario quiere descubrir.
     */

    public static void pedirCeldas (int celdasPorDescubrir) {

        for (int i = 0; i < celdasPorDescubrir; i++) {
            System.out.print("Qué celda quieres descubrir?: ");
            String[] celdaSeleccionada = read.nextLine().split(" ");
            int fila = Integer.parseInt(celdaSeleccionada[0])-1;
            int columna = Integer.parseInt(celdaSeleccionada[1])-1;

            asignarValorCelda(fila, columna);

            imprimirMatriz(matrizVisible);

            if (!comprobarGameOver(celdaSeleccionada)) {
                System.out.println("GAME OVER");
                break;
            }

        }
    }
```

```java
/**
 * Este método comprueba que la celda que ha decidido buscar el usuario
 * contenga una bomba o no, devolviendo el resultado en forma de booleano.
 * @param celdaSeleccionada La celda que ha escogido ver el usuario
 * @return Devuelve o false (Game Over), o true (no Game Over)
 */
public static boolean comprobarGameOver  (String[] celdaSeleccionada) {
    int fila = Integer.parseInt(celdaSeleccionada[0])-1;
    int columna = Integer.parseInt(celdaSeleccionada[1])-1;

    if (matrizReal[fila][columna].equals("*"))  {
        return false;
    }

    return true;
}

/**
 * Este método se utiliza para asignar el valor que corresponde a la celda que ha seleccionado el usuario,
 * comprobando cuántas bombas hay alrededor. Si la celda tiene bombas a su alrededor, se coloca en la celda
 * el número de bombas. Si no, se marca con "-" y se van descubriendo las celdas que aún no están descubiertas,
 * metiendo su posición (fila y columna) en un array para fila y otro para columna. Teniendo estos arrays, podemos
 * recorrer todas las celdas no descubiertas hasta haber conseguido mostrar todas las celdas que deban de ser
 * descubiertas alrededor de la celda inicial seleccionada.
 *
 * @param fila La fila de la celda que el usuario quiere descubrir.
 * @param columna La columna de la celda que el usuario quiere descubrir.
 */

public static void asignarValorCelda(int fila, int columna) {

    int maximo = numFilas*numColumnas;

    int[] filasPendientes = new int[maximo];
    int[] columnasPendientes = new int[maximo];

    int celdasPendientes = 1;

    filasPendientes[0] = fila;
    columnasPendientes[0] = columna;

    if (matrizReal[fila][columna].equals("*")) matrizVisible[fila][columna] = "*";
    else {

        for(int indice = 0; indice < celdasPendientes; indice++ ) {


            int filaActual = filasPendientes[indice];
            int columnaActual = columnasPendientes[indice];

            int contadorBombas = 0;


            for (int i = filaActual -1; i <= filaActual +1; i++) {
                for (int j = columnaActual -1; j <= columnaActual +1 ; j++) {

                    if (i < 0 || i >= numFilas || j < 0 || j >= numColumnas) continue;

                    if (matrizReal[i][j].equals("*")) contadorBombas++;

                }
            }

            if (contadorBombas != 0) {
                matrizVisible[filaActual][columnaActual] = Integer.toString(contadorBombas);
            } else {

                matrizVisible[filaActual][columnaActual] = "-";

                for (int i = filaActual -1; i <= filaActual +1; i++) {
                    for (int j = columnaActual -1; j <= columnaActual +1 ; j++) {

                        if (i < 0 || i >= numFilas || j < 0 || j >= numColumnas) continue;

                        if (matrizVisible[i][j].equals("X")) {
                            filasPendientes[celdasPendientes] = i;
                            columnasPendientes[celdasPendientes] = j;
                            celdasPendientes++;
                        }

                    }
                }

            }

        }
    }
}
```

Casos de prueba (Realizados en la versión “bonita”):

Camino feliz: En esta prueba, voy a crear una matriz 8x8, y voy a seleccionar 4 celdas (Como en el ejemplo de aceptaelreto), ninguna va a ser una bomba.

```
/Users/david/Library/Java/JavaVirtualMachines/op
Introduce el número de filas: 8
Introduce el número de columnas: 8
Introduce los 8 carácteres de la fila 0
-*---*--
Introduce los 8 carácteres de la fila 1
--------
Introduce los 8 carácteres de la fila 2
**------
Introduce los 8 carácteres de la fila 3
---*----
Introduce los 8 carácteres de la fila 4
---*---*
Introduce los 8 carácteres de la fila 5
-*--*---
Introduce los 8 carácteres de la fila 6
----*---
Introduce los 8 carácteres de la fila 7
--------
-*---*--
--------
**------
---*----
---*---*
-*--*---
----*---
--------
Introduce el número de celdas a descubrir: 4
Qué celda quieres descubrir?: 1 1
1XXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
```

```
Qué celda quieres descubrir?: 1 1
1XXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
Qué celda quieres descubrir?: 1 8
1XXXXX1-
XXXX111-
XXXX1---
XXXX2-11
XXXX311X
XXXXXXXX
XXXXXXXX
XXXXXXXX
Qué celda quieres descubrir?: 8 8
1XXXXX1-
XXXX111-
XXXX1---
XXXX2-11
XXXX311X
XXXXX211
XXXXX2--
XXXXX1--
Qué celda quieres descubrir?: 8 1
1XXXXX1-
XXXX111-
XXXX1---
XXXX2-11
XXXX311X
XXXXX211
1112X2--
---1X1--
```

Seleccionar celda fuera del rango, y seleccionar celda que contiene una bomba:

En la versión “bonita”, al seleccionar una bomba, te aparece la matriz antes del game over, en la versión juez, no es así.

```
/Users/david/Library/Java/JavaVirtualMachines/openjd
Introduce el número de filas: 4
Introduce el número de columnas: 4
Introduce los 4 carácteres de la fila 0
--*-
Introduce los 4 carácteres de la fila 1
----
Introduce los 4 carácteres de la fila 2
-*--
Introduce los 4 carácteres de la fila 3
----
--*-
----
-*--
----
Introduce el número de celdas a descubrir: 2
Qué celda quieres descubrir?: 10 20
Error. La celda seleccionada está fuera del rango.
Qué celda quieres descubrir?: 1 3
XX*X
XXXX
XXXX
XXXX
GAME OVER

Process finished with exit code 0
```

Version juez:

```
4 4
--*-
----
-*--
----
1
1 3
GAME OVER
```

Introducir caracteres no válidos y en la matriz poner más/menos caracteres de los requeridos:

```
/Users/david/Library/Java/JavaVirtualMachines/openjdk-25.0.1/Con
Introduce el número de filas: 4
Introduce el número de columnas: hola
Error. Debes introducir un número entero
Introduce el número de columnas: 4
Introduce los 4 carácteres de la fila 0
h
La fila debe de tener 4 carácteres.
Introduce los 4 carácteres de la fila 0
hhhh
La fila debe de tener solo '-' o '*'. Cerrando el programa.
Introduce los 4 carácteres de la fila 0
-----
La fila debe de tener 4 carácteres.
Introduce los 4 carácteres de la fila 0
----
Introduce los 4 carácteres de la fila 1
****
Introduce los 4 carácteres de la fila 2
d
La fila debe de tener 4 carácteres.
Introduce los 4 carácteres de la fila 2
----
Introduce los 4 carácteres de la fila 3
----
----
****
----
----
Introduce el número de celdas a descubrir: 1
Qué celda quieres descubrir?: 1 1
2XXX
XXXX
XXXX
XXXX
```

Seleccionar una celda que ya ha sido descubierta:

```
/Users/david/Library/Java/JavaVirtualMachines/openjdk-25.0.1
Introduce el número de filas: 4
Introduce el número de columnas: 4
Introduce los 4 carácteres de la fila 0
----
-*--
----
----Introduce los 4 carácteres de la fila 1
Introduce los 4 carácteres de la fila 2
Introduce los 4 carácteres de la fila 3

----
-*--
----
----
Introduce el número de celdas a descubrir: 5
Qué celda quieres descubrir?: 1 1
1XXX
XXXX
XXXX
XXXX
Qué celda quieres descubrir?: 1 2
11XX
XXXX
XXXX
XXXX
Qué celda quieres descubrir?: 1 1
La celda seleccionada ya ha sido vista. Selecciona otra
Qué celda quieres descubrir?: 1 1
La celda seleccionada ya ha sido vista. Selecciona otra
Qué celda quieres descubrir?: 2 2
11XX
X*XX
XXXX
XXXX
GAME OVER
```