Treball final Programació I

TBMaze - Laberint en Java

A. Antonio Boutaour Sanchez
43202227Q
tboutaour@gmail.com

Bernat Pericàs Serra
43212796M
bernatpericasserra97@gmail.com

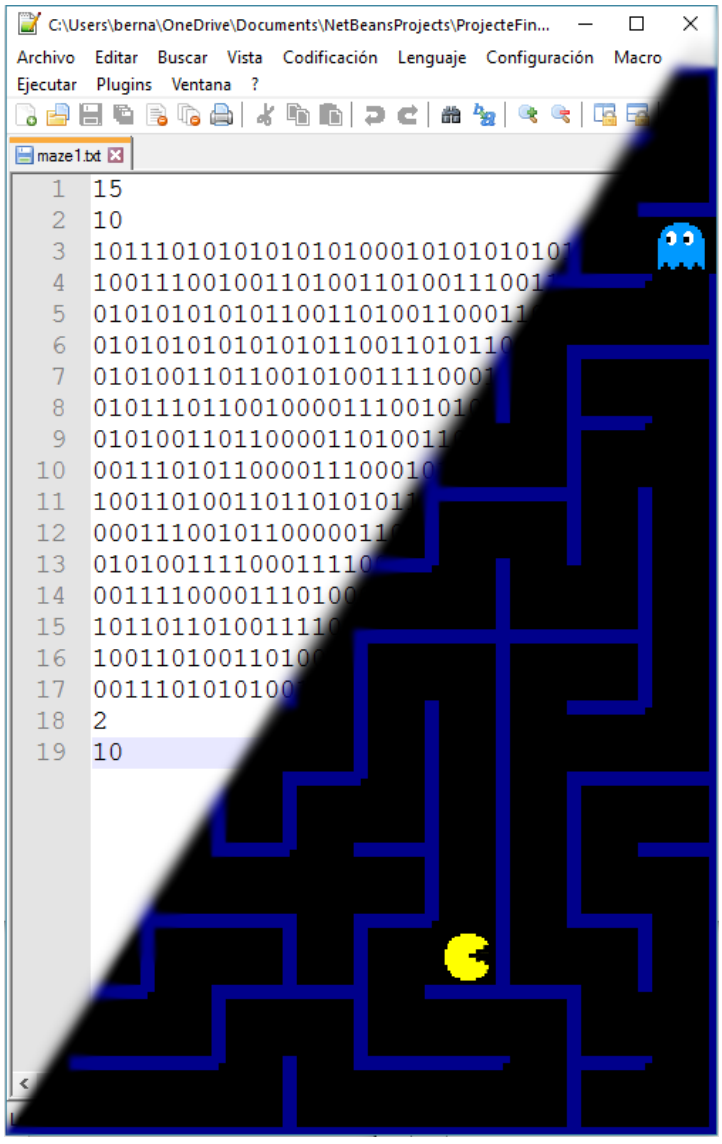
Índex

Introducció

Per a aquest projecte, es demana la creació d'un programa en Java capaç d'interpretar dades d'un fitxer i transformar-les en un laberint. Aquest fitxer d'extensió *.txt* conté tant l'estructura de les parets com la posició de la sortida. També ha de permetre moure una fitxa fins a la sortida mitjançant el teclat de l'ordinador (les lletres W, A, S i D) en cas que no hi hagi paret que ho impedeixi. A més, el joc ha de fer possible obrir un altre laberint (un altre fitxer de dades de les mateixes característiques), reiniciar la posició de la fitxa i acabar l'execució del programa en qualsevol moment sense que això sigui un problema. En arribar a la sortida s'ha de mostrar un missatge per pantalla en forma de finestra emergent.

Cal tenir en compte que la posició inicial de la fitxa no ve donada pel fitxer de dades, sinó que es generarà a partir de dos pseudoaleatoris corresponents a una fila i a una columna del laberint.

A la Il·lustració 1 es mostra l'equivalència entre el fitxer de dades i el resultat final.



Il·lustració 1. Estructura de dades / Visualització

Disseny i procés seguit

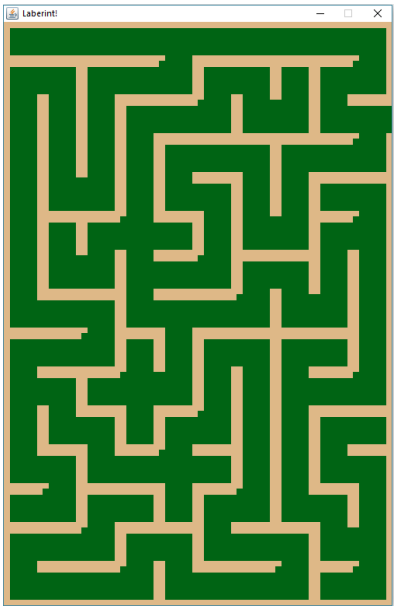
Gràcies al disseny descendent s'ha pogut arribar a la solució final de manera escalonada i ordenada. Aquest ha permès anar madurant, de mica en mica, la idea inicial del programa fins a la solució definitiva. Així, s'ha elaborat el programa des de la part de més alt nivell, el laberint en sí, fins a la part més concreta i de més baix nivell, essent aquest darrer nivell l'assignació de les característiques físiques de les fitxes, tractades a nivell de píxel.

Podem observar llavors una estructura amb unes parts ben definides, el laberint està format d'elements casella i cada una d'aquestes pot (o no) tenir una fitxa associada.

A continuació es descriuen les passen seguides per a la resolució del problema.

Representar l'estructura de dades

Aquesta fou la primera passa que es va dur a terme i que va servir com a punt de partida per a la implementació de les altres funcions. El procés d'abstracció d'una matriu de caselles fou un gran avanç ja que fins a aquell punt no s'havia aconseguit cap solució factible. Apareix llavors una estructura on el laberint deixa de ser una part unitària per transformar-se en una suma d'elements casella independents un dels altres i els quals units formen la graella del joc.



Il·lustració 2.

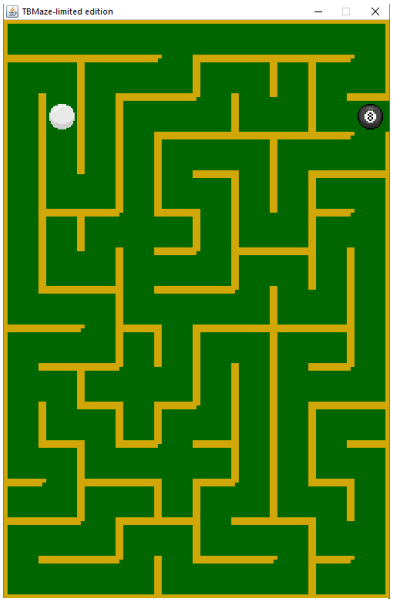
Si bé emmagatzemar les dades en una matriu no fou difícil, pintar aquestes dades sí va comportar unes hores de dedicació.

El resultat s'observa en la Il·lustració 2.Il·lustració 2.

Col·locar la fitxa i sortida

La segona passa fou col·locar les dues imatges del joc: la fitxa controlada pel teclat es mourà fins a la fitxa de la sortida, que pel contrari es mantindrà estàtica. Al principi es van ubicar a l'arrel de la carpeta del programa però quan es van afegir més imatges es va decidir ficar-les dins una subcarpeta per a la millor organització. Simplement amb la classe Fitxa i la classe ja creada *FitxerIn* es va implementar la possibilitat d'importar imatges. Aquest procés ocorre totalment al marge de l'usuari. Finalment, amb el mètode *paintComponent* de la mateixa classe Fitxa es pinten les imatges.

El resultat es pot observar a la Il·lustració 3.



Il·lustració 4

Moure la fitxa

Per a dur a terme el moviment de la fitxa, o més ben dit, la seva simulació, es pensaren vàries solucions. Una era la de pintar l'espai que ocupa la fitxa en el temps *t* així com estava al temps t-1 i pintar aqueixa a la posició desitjada. El problema d'aquesta solució és la seva dificultat i, a causa del poc temps disponible, es va optar per a una solució més pràctica.

Aquesta solució alternativa era la de repintar tot el laberint amb la fitxa a la posició desitjada, encara que no és la solució que requereixi menys treball del processador i estalvi de recursos, encara que amb aquest joc tan simple no s'apreciaria gaire diferència. En moure la fitxa necessitàrem per primer cop escoltar els esdeveniments de teclat i, depenent de la tecla clicada, realitzar una acció o una altra. Una vegada més, la implementació d'aquesta funcionalitat fou possible gràcies a la quantitat de documentació que existeix a internet i els coneixements adquirits a classe.

El resultat, després de permetre el moviment de la fitxa, es pot observar al vídeo de l'execució del programa.

Reiniciar la posició

Reiniciar la fitxa a una altra posició pseudoaleatòria fou gairebé trivial després d'haver implementat el mètode per pintar-la. L'únic que faltà programar fou un altre mètode per a llevar la fitxa actual, és a dir, canviar el booleà *ocupada* de la classe *Casella* a fals. Aquesta opció del joc fou la primera que necessità l'existència d'un menú des d'on activar-la. Aquesta opció del menú fou el referent per a implementar posteriorment altres opcions. Un cop es va permetre reiniciar la posició, degut a la quantitat de vegades que vam utilitzar l'opció de reiniciar, vam adonar-nos que faltava alguna mena de drecera de teclat per a accelerar l'opció de reiniciar, així com també les altres. Fou pràcticament instantani afegir l'escoltador de teclat i assignar-li l'opció de reiniciar, ja que és similar als escoltadors de teclat pel moviment de la fitxa.

El missatge final

La creació d'aquest missatge va ser bastant senzilla ja que ens vam poder basar amb els exemples vists a classe. Vam optar per fer una classe Missatge ja que el missatge final no seria l'únic missatge que es mostraria per pantalla durant l'execució del programa, creiem que la comunicació programa-usuari és fonamental.

Obrir altres laberints

Aquesta funcionalitat fou segurament la que més problemes suposà. En primer lloc era la primera vegada que necessitàvem d'alguna finestra que permetés la selecció de fitxers i de tots els errors que això produeix.

Canviar el tema

Aquesta opció fou implementada només per entreteniment, ja que no es demana a l'enunciat. Inicialment havíem triat la temàtica del billar però ens vam decantar per la del joc "Pac-man". Per utilitzar les icones ja creades del billar es va decidir incloure la possibilitat de canviar de temàtica. Fou una tasca afegida que aporta al laberint certa relació amb l'usuari, ja que pot personalitzar la part estètica segons el seus gustos.

Descripció de les classes

A continuació s'ofereix una descripció breu de les classes que formen el joc.

Main

Aquesta és la classe principal i la que conté l'estructura del programa a alt nivell. És un JFrame i implementa un KeyListener. El JFrame conté un menú (*JMenu*) i el panell del laberint (*JPanel*). Aquesta classe està formada pel *Main()*, el que podríem dir que és el programa a més alt nivell. El mètode *initComponents()* defineix els elements del JFrame, els assigna una acció mitjançant *ActionListener*s i *ActionPermormed*s.

Trobam tots els atributs privats excepte filaLab i columnaLab que són públics degut a la seva constant modificació per part de Laberint, a més, els trobam com a estàtics, elements comuns a totes les classes.

Laberint

La classe Laberint és un JPanel que conté el mètode *PintaLaberint* que llegeix i crea la matriu de caselles amb les seves corresponents variables sobre el tipus de casella, la posició i les parets a representar.

Casella

Aquesta classe és l'encarregada de donar la separació en unitats del panell laberint, i dóna llibertat a cada una d'aquestes amb un constructor que importa elements gràfics. És aquí on es realitza la tasca de pintar el laberint pintant o no les parets depenent dels booleans dels punts cardinals.

Finalment, l'atribut "ocupada" dóna la capacitat de definir aquesta casella com a contenidora (o no) tant de la fitxa principal com de la fitxa de sortida i és a partir d'aquest atribut que es realitza la impressió (o no) de l'element que es mou.

Cal destacar l' àmbit de visibilitat privat de tots els atributs d'aquesta classe, que els blinda de modificacions.

Fitxa

Com a element més baix de l'escala d' abstracció trobam la fitxa. La classe "Fitxa" conté tots els atributs necessaris per al seu correcte funcionament.

Com a atributs privats trobam les coordenades FILA-COLUMNA de la fitxa. Podem destacar la visibilitat estàtica de les dues fitxes ("Fitxa" i "Sortida") degut a que són elements comuns per a totes les classes.

Incidim llavors en el treball realitzat a nivell de píxel per ajustar la imatge a les dimensions de la casella i que, amb els nivells d'abstracció fan impossible de veure'l en una part superior del projecte.

FitxerIn

Aquesta és una classe creada per al tractament amb comoditat de fitxers de dades. S'encarreguen de la cridada tant de FileReaders com de BufferedReaders, a més del tancament d'aquests. És llavors una classe estàndard que ja s'havia usat en altres projectes.

Missatge

S'ha programat de tal manera que basta crear un objecte missatge passant-li un String perquè una finestra amb el text de l'*String* es mostri per pantalla, a més, contendrà un botó "Acceptar" que tancarà la finestra.

Tema

Com a darrera classe afegida al projecte i totalment voluntària, veiem el tractament que hem trobat adient per realitzar l'administració d'estils.

Com a únic atribut comptem amb un array d'objectes que contendrà només 4 elements: Color del fons, color de les parets, la fitxa de moviment i la de la sortida. Aquest array és l'encarregat d'emmagatzemar les característiques estètiques del laberint. Trobem els diferents sets individualitzats a cada posició de l'array que permeten la modificació de les diferents parts individualment.

Aquesta classe realitza la funció de tenir un sol element contenidor per a tot el laberint i és a partir d'aquest que pinta el laberint segons pertoqui.

Joc de proves

Per a comprovar el seu correcte funcionament s'han realitzat una sèrie de proves de les seves distintes opcions i del joc en sí.

Durant la fase de representació de l'estructura de dades, es va modificar un dels quatre fitxers de text que formen els laberints comprovant així que el programa representa l'estructura de forma correcta.

Una altra prova realitzada fou la d'intentar obrir un arxiu que no compleixi les especificacions del format del laberint, en aquest cas el laberint ha d'informar de l'error amb un missatge (finestra emergent) i continuar la seva execució.

A més de les proves prèviament descrites, per a comprovar que el programa fos intuïtiu i fàcil d'usar, una persona amb coneixements molt bàsics d'informàtica va tenir la possibilitat de jugar-hi. Aquesta persona (aliena a la UIB i al curs de programació) va servir per demostrar-nos de manera objectiva que el joc és apte per a tothom que sàpiga usar un ordinador.

Encara que es tracti d'un joc molt simple, si no estàs estructurat de manera correcta i no presentés cap tipus de comunicació amb l'usuari, probablement no seria capaç d'utilitzar-lo. Per aquest motiu hi ha creades opcions com la finestra d'ajuda o el missatge que apareix quan acabes el laberint.

Conclusions

La realització d'aquest projecte en forma de joc ha permès assolir conceptes com per exemple el tipus abstracte de dades, ja que els fitxers dels laberints són inintel·ligibles sense un programa dissenyat per a la seva interpretació. També ha permès consolidar tota la part de l'entorn gràfic de Java, perquè gran part del programa és algun component d'aquest entorn.

Pel que fa als problemes trobats, cal destacar l'error causat pel botó de cancel·lar del *FileChooser,* es va trobar la solució durant la realització d'un altre exercici que també implementava el FileChooser, el Taller 2. Després només va caldre extrapolar-ho a aquesta pràctica.

La tasca de tractament d'errors es va realitzar a consciència. S'han intentat tractar tots els errors possibles. S'ha aconseguit un nivell de fiabilitat alt, ja que cap dels dos components del grup ni la persona usada al jocs de proves han pogut produir un error no tractat.

Un altre del les dificultats fou la de pintar les imatges de la fitxa i de la sortida de forma correcta, ja que es pintaven només parcialment sobre la casella. Es solucionà a base de prova i error i tractament a nivell de píxel .

Aquest programa ha implicat un esforç de recerca dins la API i diversos fòrums sobre Java per a implementar opcions com el FileChooser, a més de basar-nos en alguns de molts exemples de programes que treballen amb entorns gràfics que hi ha disponibles.

Per a concloure, caldria justificar el nom del laberint (TBMaze), que no és més que la primera lletra dels noms dels dos components del grup junt a la paraula "laberint" en anglès, tasca que, com s'observa, no fou gaire complicada.

Nota dels autors: hi ha disponible un vídeo a YouTube (ocult) sobre l'execució del programa. Si l' hipervincle no està disponible, l'enllaç és el següent:

https://www.youtube.com/watch?v=dUmFLI0ldDc

Codi font

Main

```
/**
 * Aquesta classe es l'executable del projecte, s'inicia aqui i tots els
 * elements de la part "jugable" del projecte estan situats aquí.
 * Es tracta, llavors, de la classe que uneix tots les altres.
 */
package projectefinallaberint;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.io.File;
import java.io.IOException;
import javax.swing.*;

/**
 * @authors Toni Boutaour, Bernat Pericàs
 * @DNIs 43202227Q, 43212796M
 */
public class Main extends JFrame implements KeyListener {

    private final Laberint lab;

    private JMenuBar jmbarLab;
    private JMenu jmenuJoc, jmenuMode;
    private JMenuItem jmItemSortir, jmItemReiniciar, jmItemModeBillar,
            jmItemModePacman, jmItemEscollir, jmItemHelp, jmItemModePokemon;

    static int filaLab = (int) Math.floor(Math.random() * (15 + 1));
    static int columnaLab = (int) Math.floor(Math.random() * 10);

    /**
     * El constructor del main s'encarrega de l'inici del joc, possa a punt
els
     * diferents elements de l'entorn gràfic amb la cridada a initsComponents
i
     * la colocació de les fitxes.
     */
    public Main() {
        super("TBMaze-Edició limitada");
        initComponents();
        lab = new Laberint();
        this.addKeyListener(this);

        lab.coloca(Fitxa.fitxa, filaLab, columnaLab);
        // com que la sortida està indicada
        //una posició enllà respecte a les columnes,s'ha de tenir en compte.
        lab.coloca(Fitxa.sortida, lab.getSortidaX(), lab.getSortidaY() - 1);

        this.getContentPane().add(lab);
        this.setSize(lab.getPreferredSize());
        this.setResizable(false);
        this.pack();
        setLocationRelativeTo(null);//la finestra surt al centre
        this.setDefaultCloseOperation(Main.EXIT_ON_CLOSE);
    }

    /**
     * initComponents s'encarrega de la gestió de tots els elements de
l'entorn
```

```
     * gràfic de la barra al JFrame.
     */
    private void initComponents() {
        jmbarLab = new JMenuBar();
        jmenuJoc = new JMenu();
        jmenuMode = new JMenu();
        jmItemReiniciar = new JMenuItem();
        jmItemSortir = new JMenuItem();
        jmItemModeBillar = new JMenuItem();
        jmItemModePacman = new JMenuItem();
        jmItemModePokemon = new JMenuItem();
        jmItemEscollir = new JMenuItem();
        jmItemHelp = new JMenuItem();

        this.getContentPane().add(jmbarLab);
        jmItemReiniciar.setText("Reiniciar (tecla R)");
        jmItemReiniciar.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent evt) {
                jmItemReiniciaActionPerformed(evt);
            }
        });

        jmItemSortir.setText("Sortir (tecla Supr)");
        jmItemSortir.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent evt) {
                jmItemSortirActionPerformed(evt);
            }
        });

        jmItemModeBillar.setText("Mode Billar");
        jmItemModeBillar.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent evt) {
                jmItemModeBillarActionPerformed(evt);
            }
        });

        jmItemModePacman.setText("Mode Pacman");
        jmItemModePacman.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent evt) {
                jmItemModePacmanActionPerformed(evt);
            }
        });

        jmItemModePokemon.setText("Mode Pokemon");
        jmItemModePokemon.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jmItemModePokemonActionPerformed(evt);
            }
        });

        jmItemHelp.setText("Ajuda (tecla H)");
        jmItemHelp.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent evt) {
                jmItemHelpActionPerformed(evt);
            }
        });

        jmItemEscollir.setText("Escollir (tecla C)");
```

```
        jmItemEscollir.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent evt) {
                jmItemEscollirActionPerformed(evt);
            }
        });

        jmenuJoc.setText("Opcions");
        jmenuMode.setText("Canviar tema");

        jmenuJoc.add(jmItemReiniciar);
        jmenuJoc.add(jmItemSortir);
        jmenuJoc.add(jmItemEscollir);
        jmenuJoc.add(jmItemHelp);

        jmenuMode.add(jmItemModeBillar);
        jmenuMode.add(jmItemModePacman);
        jmenuMode.add(jmItemModePokemon);

        jmbarLab.add(jmenuJoc);
        jmbarLab.add(jmenuMode);
        jmbarLab.setBounds(0, 0, 492, 36);

        setJMenuBar(jmbarLab);
    }

    /*
    *Tots els actionPerformedEncarregats de realitzar una tasca al laberint:
     */
    private void jmItemSortirActionPerformed(ActionEvent evt) {
        System.exit(0);
    }

    private void jmItemModeBillarActionPerformed(ActionEvent evt) {
        lab.canviarTema(new Color(0, 102, 0), new Color(208, 167, 6),
                "imatges/blanca.png", "imatges/8.png");
    }

    private void jmItemModePacmanActionPerformed(ActionEvent evt) {
        lab.canviarTema(Color.BLACK, new Color(0, 0, 139),
                "imatges/pacman.png", "imatges/ghost.png");
    }

    private void jmItemModePokemonActionPerformed(ActionEvent evt) {
        lab.canviarTema(Color.yellow, Color.black,
                "imatges/pokeball.png", "imatges/pikachu.png");
    }

    private void jmItemReiniciaActionPerformed(ActionEvent evt) {
        //Tornar a posar la fitxa a una posició random.
        lab.reiniciarLaberint();
    }

    private void jmItemHelpActionPerformed(ActionEvent evt) {
        MostraAjuda();
    }

    private void jmItemEscollirActionPerformed(ActionEvent evt) {
        EscollirLaberint();
    }

    /**
     * Realització de tota la part de jugabilitat: segons la pulsació de les
     * diferents tecles asignades per al moviment es realitza aquest.
     *
     * @param ke event de teclat
     */
```

```
@Override
public void keyPressed(KeyEvent ke) {
    try {
        switch (ke.getKeyCode()) {
            case KeyEvent.VK_W:
                System.out.println("W");
                if (!lab.muradaSuperior(filaLab, columnaLab)) {
                    System.out.println("Pot pujar");
                    lab.llevaFitxa(filaLab, columnaLab);
                    filaLab = filaLab - 1;
                    lab.coloca(Fitxa.fitxa, filaLab, columnaLab);
                    repaint();
                    lab.incrementarMoviments();
                    System.out.println(lab.getMoviments());
                } else {
                    System.out.println("No pot pujar");
                }
                if (lab.esSortida(filaLab, columnaLab)) {
                    Finalitza();
                }
                break;

            case KeyEvent.VK_D:
                System.out.println("D");
                if (!lab.muradaDreta(filaLab, columnaLab)) {
                    System.out.println("Pot anar a la dreta");
                    lab.llevaFitxa(filaLab, columnaLab);
                    columnaLab = columnaLab + 1;
                    lab.coloca(Fitxa.fitxa, filaLab, columnaLab);
                    repaint();
                    lab.incrementarMoviments();
                    System.out.println(lab.getMoviments());
                } else {
                    System.out.println("No pot anar a la dreta");
                }
                if (lab.esSortida(filaLab, columnaLab)) {
                    Finalitza();
                }
                break;

            case KeyEvent.VK_S:
                System.out.println("S");
                if (!lab.muradaInferior(filaLab, columnaLab)) {
                    System.out.println("Pot baixar");
                    lab.llevaFitxa(filaLab, columnaLab);
                    filaLab = filaLab + 1;
                    lab.coloca(Fitxa.fitxa, filaLab, columnaLab);
                    repaint();
                    lab.incrementarMoviments();
                    System.out.println(lab.getMoviments());
                } else {
                    System.out.println("No pot baixar");
                }
                if (lab.esSortida(filaLab, columnaLab)) {
                    Finalitza();
                }
                break;

            case KeyEvent.VK_A:
                System.out.println("A");
                if (!lab.muradaEsquerra(filaLab, columnaLab)) {
                    System.out.println("Pot anar a l'esquerra");
                    lab.llevaFitxa(filaLab, columnaLab);
                    columnaLab = columnaLab - 1;
                    lab.coloca(Fitxa.fitxa, filaLab, columnaLab);
                    repaint();
                    lab.incrementarMoviments();
                    System.out.println(lab.getMoviments());
```

```
                    } else {
                        System.out.println("No pot anar a l'esquerra");
                    }
                    if (lab.esSortida(filaLab, columnaLab)) {
                        Finalitza();
                    }
                    break;

                case KeyEvent.VK_R:
                    lab.reiniciarLaberint();
                    break;
                case KeyEvent.VK_DELETE:
                    Laberint.tancarLaberint();
                    break;
                case KeyEvent.VK_H:
                    MostraAjuda();
                    break;
                case KeyEvent.VK_C:
                    EscollirLaberint();
                    break;
            }
        } catch (Exception e) {
            lab.incrementarMoviments();
        }
    }

    /**
     * Sense utilitat, però el mètode s'ha d'implementar obligatòriament.
     *
     * @param key event de teclat
     */
    @Override
    public void keyReleased(KeyEvent key) {
    }

    /**
     * Sense utilitat, però el mètode s'ha d'implementar obligatòriament.
     *
     * @param ke event de teclat
     */
    @Override
    public void keyTyped(KeyEvent ke) {
    }

    private void MostraAjuda() {
        JOptionPane.showMessageDialog(lab, //Finestra relativa al JFrame lab.
                "Benvingut al gran laberint.\n"
                + "Els següents controls defineixen el moviment de la
fitxa:\n"
                + "A: Moviment a l'esquerra.\n"
                + "W: Moviment cap amunt.\n"
                + "D: Moviment a la dreta.\n"
                + "S: Moviment cap avall.\n"
                + "El joc consisteix en arribar en el menor nombre \nde"
                + " moviments a la sortida.\n \n"
                + "Molta sort!\n",
                "Panell auxiliar per a l'ajut de l'usuari", //Capçalera
                JOptionPane.INFORMATION_MESSAGE);           //Icona
    }

    /**
     * Mètode que s'encarrega del tractament de l'elecció d'un nou laberint.
     * Aquest realitza la operació d'apertura d'un fileChooser que serà la
     * interfície per agafar un nou element Laberint.
     *
     * S'ha tractat l'error d'elecció un fitxer que no és correcte
emmagatzemant
     * un un string la direcció del laberint actual i realitzar un repintat
```

```
     * d'aquest en el moment que l'usuari no agafi un laberint vàlid.
     */
    private void EscollirLaberint() {
        JFileChooser chooser = new JFileChooser();
        chooser.setCurrentDirectory(new File(System.getProperty("user.dir")
                + "/laberints"));
        String s;
        //String auxiliar que conte el nom del laberint actual.
        String aux = lab.getNomLab();
        try {
            int chooseraux = chooser.showOpenDialog(this);
            if (chooseraux == JFileChooser.APPROVE_OPTION) {
                s = chooser.getSelectedFile().getCanonicalPath();
                lab.changeLab(s);
                lab.reiniciarLaberint();
            }
            //Si apareix cap error, se reestableix el lab anterior.
        } catch (IOException | NullPointerException |
                ArrayIndexOutOfBoundsException ex) {
            JOptionPane.showMessageDialog(lab, //Finestra relativa al JFrame
lab
                    "Disculpi,\n"
                    + "el fitxer no compleix les propietats necesaries per a "
                    + "ser tractat.",
                    "Error a l'execució", //Capçalera
                    JOptionPane.INFORMATION_MESSAGE);
            lab.changeLab(aux);
            lab.reiniciarLaberint();
        }
    }

    /**
     * Mètode que realitza l'execució del missatge de finalitzat del joc.
     */
    public void Finalitza() {
        Missatge miss = new Missatge("Has acabat en "
                + lab.getMoviments() + " moviments");
        lab.canviarLaberint();
        lab.coloca(Fitxa.sortida, lab.getSortidaX(),
                lab.getSortidaY() - 1);
    }

    /**
     * Inicia el programa
     *
     * @param args
     */
    public static void main(String[] args) {
        Main inici = new Main();
        inici.setVisible(true);
    }
}
```

Laberint

```
/**
 * La classe Laberint es tracta de la classe contenidora dels elements que
 * treballen conjuntament per formar un sol element.
 * Es defineix com un Panell que manté dins un array bidimensional de
casselles
 * que posteriorment amb la inseció de la fitxa de moviment i de sortida
donaràn
 * la capacitat de jugabilitat.
 * Com atributs principals trobam el costat de la casella, que donarà la
 * distribució espacial al panell i el nombre de files i columnes que deixaran
 * preparat a l'element Laberint per al seu desenvolupament.
 */
package projectefinallaberint;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.geom.Rectangle2D;
import java.io.IOException;
import java.util.Random;
import javax.swing.JPanel;

/**
 * @authors Toni Boutaour, Bernat Pericàs
 * @DNIs 43202227Q, 43212796M
 */
public class Laberint extends JPanel {

    //Cream estilLab com un objecte Tema, que és un array d'objectes per tal
de
    //tenir una bona organització d'aquestos.
    public static Tema estilLab = new Tema(Color.BLACK, new Color(0, 0, 139),
            "imatges/pacman.png", "imatges/ghost.png");

    private static final int COSTAT = 50;
    public static int files, columnes;
    private int sortidaY, sortidaX;
    private Casella[][] arrLab;
    private Casella casellafinal;
    private String nomLab = "laberints/Maze1.txt";
    private FitxerIn f;
    private int moviments = 0;

    /**
     * Constructor que crita a la creació d'aquest.
     */
    public Laberint() {
        PintaLaberint();

    }

    /**
     * Mètode principal, crea un array bidimensional que donara forma al
tauler
     * pricipal del joc, format de la unió dels elements casella. Aquest
mètode
     * crea a partir de la lectura del fitxer corresponent al laberint triat
la
     * impressió de les seves característiques. Hem valorat llegir la linea i,
     * com sabem la longitud d'aquest ficar-ho completament dins un array que
     * controlarem totalment.
     *
     * Se ha realitzat el tractament dels error necesaris per a la millor
     * experiència possible per l'usuari.
     */
```

```
public void PintaLaberint() {
    //Llegir nombre de files i columnes
    try {
        //Crear fitxer a partir del nom.
        f = new FitxerIn(nomLab);
        Laberint.files = Integer.parseInt(f.getBr().readLine());
        Laberint.columnes = Integer.parseInt(f.getBr().readLine());
    } catch (IOException | NumberFormatException ex) {
        System.out.println("Error llegint les files i/o columnes");
    }
    System.out.println("#files:" + files + " #columnes:" + columnes);
    //Cream array de Caselles
    arrLab = new Casella[files][columnes];
    int y = 0;
    for (int i = 0; i < files; i++) {
        int x = 0;
        char[] Array = null;

        try {
            Array = LiniadeBits(columnes, f.getBr().readLine());
        } catch (IOException ex) {
            System.out.println("Error llegint una linia del fitxer");
        }

        int contador = 0;
        for (int j = 0; j < columnes; j++) {
            Rectangle2D.Float r = new Rectangle2D.Float(x, y, COSTAT,
                    COSTAT);

            boolean nord = cercaU(Array, contador);
            contador++;
            boolean east = cercaU(Array, contador);
            contador++;
            boolean south = cercaU(Array, contador);
            contador++;
            boolean west = cercaU(Array, contador);
            contador++;

            arrLab[i][j] = new Casella(r, x, y, nord, east, south, west,
                    estilLab.getColorFons());
            x += COSTAT;
        }
        y += COSTAT;
    }
    //LLegir posició sortida
    try {
        //llegir fila(EIX -OY)
        this.sortidaY = Integer.parseInt(f.getBr().readLine());
        //llegir columna(EIX -OX)
        this.sortidaX = Integer.parseInt(f.getBr().readLine());
        casellafinal = arrLab[sortidaY][sortidaX - 1];
    } catch (IOException ex) {
        System.out.println("Error llegint la posició de la sortida");
        System.out.println(ex.getLocalizedMessage());
        Missatge mis = new Missatge("Error llegint el fitxer");
    }

    System.out.println("sortidaFILA: " + sortidaY
            + " sortidaCOLUMNA: " + sortidaX);

    f.tancarFitxer();
}

/**
 * Aquest mètode modifica el nom del laberint que s'usa per realitzar el
 * pintat d'aquest.
 *
 * @param g
```

```
 */
public void changeLab(String g) {
    nomLab = g;//Actualitza nom laberint
    PintaLaberint(); //Tornar a dibuixar
    repaint();
}

/**
 * Realitza la conversió d'un string a un Array de caràcters.
 *
 * @param columnes
 * @param s
 * @return
 */
public static char[] LiniadeBits(int columnes, String s) {
    char[] Array = s.toCharArray();
    return Array;
}

/**
 * Mètod que retorna true en trobar 1 a la cadena de caràcters passats per
 * paràmetre.
 *
 * @param erre
 * @param i
 * @return
 */
public static boolean cercaU(char[] erre, int i) {
    return erre[i] == '1';
}

/**
 * Impressió de les caselles una a una fins a completar el panell.
 *
 * @param g
 */
@Override
public void paintComponent(Graphics g) {
    try {
        for (int i = 0; i < files; i++) {
            for (int j = 0; j < columnes; j++) {
                arrLab[i][j].paintComponent(g);
            }
        }
    } catch (NullPointerException ex) {
    }
}

/**
 * Mètode de col·locació una fitxa dins una casella.
 *
 * @param s nom de la fitxa
 * @param x coordenada x
 * @param y coordenada y
 */
public void coloca(String s, int x, int y) {
    arrLab[x][y].setFitxa(new Fitxa(s));
}

/**
 * Mètode per a l' extracció de la fitxa.
 *
 * @param x
 * @param y
 */
public void llevaFitxa(int x, int y) {
    arrLab[x][y].foraFitxa();
}
```

```
/**
 *
 * @return
 */
public int getSortidaX() {
    return sortidaY;
}

public int getSortidaY() {
    return sortidaX;
}

public static int getCostat() {
    return COSTAT;
}

public int getFiles() {
    return files;
}

public int getColumnes() {
    return columnes;
}

/**
 *
 * @return
 */
@Override
public Dimension getPreferredSize() {
    return new Dimension(columnes * COSTAT, files * COSTAT);

}

/**
 * Mètodes que aporten una abstracció important en la qual no es veu
 * inicialment com es realitza el proces de definir la murada.
 *
 * @param x
 * @param y
 * @return
 */
public boolean muradaSuperior(int x, int y) {
    return arrLab[x][y].muradaNord();
}

public boolean muradaDreta(int x, int y) {
    return arrLab[x][y].muradaEst();
}

public boolean muradaInferior(int x, int y) {
    return arrLab[x][y].muradaSud();
}

public boolean muradaEsquerra(int x, int y) {
    return arrLab[x][y].muradaOest();
}

/**
 * Mètode per a la reinicialització del laberint. Aquest realitza la
 * initzialització de nou de la fitxa a una posició nova aleatòria i el
 * comptador a cero per al nou intent.
 */
public void reiniciarLaberint() {
    //Crea unes posicions random per a la fitxa
    Random rndFiles = new Random();
    Random rndColumnes = new Random();
```

```
        //Comptador de moviments a 0
        this.moviments = 0;
        //Llevar tant la fitxa com la posició de sortida.(universal per poder
        //fer servir aquest mètode per canviar de laberint)
        try {
            llevaFitxa(Main.filaLab, Main.columnaLab);
            llevaFitxa(sortidaY, sortidaX - 1);
        } catch (ArrayIndexOutOfBoundsException ex) {
        }
        Main.filaLab = rndFiles.nextInt(files);
        Main.columnaLab = rndColumnes.nextInt(columnes - 1);
        //Possam la fitxa als llocs marcats per els randoms.
        coloca(Fitxa.fitxa, Main.filaLab, Main.columnaLab);
        coloca(Fitxa.sortida, sortidaY, sortidaX - 1);
        repaint();
    }

    /**
     * Una vegada acabat el laberint, es caviarà automàticament, Tenim en
compte
     * que pot apareixer el mateix laberint que s'ha fet.
     */
    public void canviarLaberint() {
        Random rnd1 = new Random();
        int i = rnd1.nextInt(2) + 1;
        String s = "laberints/Maze" + i + ".txt";
        changeLab(s);
        reiniciarLaberint();
    }

    /**
     * Tancament del Laberint
     */
    public static void tancarLaberint() {
        System.exit(0);
    }

    /**
     * Increment del comptador de moviments
     */
    public void incrementarMoviments() {
        this.moviments = moviments + 1;
    }

    /**
     * Mètode que retorna un boolean que descriu si s'ha arribat o no a la
     * casella definida com sortida.
     *
     * @param x coordenada x
     * @param y coordenada y
     * @return
     */
    public boolean esSortida(int x, int y) {
        Casella cas = arrLab[x][y];
        return cas == casellafinal;

    }

    /**
     * Implementació de metode que mostra una pantalla amb els moviments fets.
     *
     */
    public void tractamentSortida() {
        Missatge miss = new Missatge("Has acabat en "
                + moviments + " moviments");

        canviarLaberint();
        coloca(Fitxa.sortida, sortidaX,
```

```
                sortidaY - 1);
    }

    public void canviarTema(Color colFons, Color colParet, String fitxaMov,
            String fitxaSortida) {
        estilLab.setColorFons(colFons);
        estilLab.setColorParet(colParet);
        estilLab.setFitxaMoviment(fitxaMov);
        estilLab.setFitxaSortida(fitxaSortida);

        llevaFitxa(Main.filaLab, Main.columnaLab);
        coloca(Fitxa.fitxa, Main.filaLab, Main.columnaLab);
        llevaFitxa(sortidaY, sortidaX - 1);
        coloca(Fitxa.fitxa, sortidaY, sortidaX - 1);
        PintaLaberint();
        reiniciarLaberint();
    }

    /*
     * Getters i setters.
     *
     */
    public int getMoviments() {
        return moviments;
    }

    public String getNomLab() {
        return nomLab;
    }
}
```

Casella

```
/**
 * Classe que presenta totes les característiques tant físiques com lógiques
de
 * la casella.
 * Aquesta casella es tracta d'un quadrat pintat segons es desitja i que, a
més
 * conté o no les parets corresponents que constitueixen el laberint.
 * Cal destacar la inserció d'un objecte fitxa que permetrà la interacció
entre
 * aquest dos elements i la capacitat de moviment de la fitxa principal a
partir
 * del booleà "ocupada".
 */
package projectefinallaberint;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Rectangle2D;

/**
 * @authors Toni Boutaour, Bernat Pericàs
 * @dnis 43202227Q, 43212796M
 */
public class Casella {

    private final Rectangle2D.Float rect;
    private final Color col;
    private boolean ocupada;
    private Fitxa fitxa;

    private final boolean nord, est, sud, oest;

    private final int x, y;
    private final int costat = Laberint.getCostat();

    /**
     * Constructor encarregat de definir tots els atributs d'aquesta clase.
     *
     * @param r rectangle
     * @param x coordenada x
     * @param y coordenada y
     * @param nord paret adalt
     * @param est paret dreta
     * @param sud paret abaix
     * @param oest paret esquerra
     * @param c
     */
    public Casella(Rectangle2D.Float r, int x, int y,
            boolean nord, boolean est, boolean sud, boolean oest, Color c) {
        this.rect = r;
        this.x = x;
        this.y = y;
        this.nord = nord;
        this.est = est;
        this.sud = sud;
        this.oest = oest;
        this.fitxa = null;
        this.col = c;
    }

    /**
     * Aquest PaintComponent s'encarrega de pintar tant el fons com les linies
```

```
     * que conformen el laberint. La inserció o no de la paret ve donada per
els
     * atributs de posició (Nord, est,sud,oest).
     *
     * @param g objecte grafic
     */
    public void paintComponent(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
        g2d.setColor(this.col);
        g2d.fill(this.rect);
//---------------PINTAR LINIES---------------------------------
        g2d.setColor(Laberint.estilLab.getColorParet());
        g2d.setStroke(new BasicStroke(10));
        if (nord == true) {
            g2d.drawLine(x, y, x + costat, y);
        }
        if (est == true) {
            g2d.drawLine(x + costat, y, x + costat, y + costat);
        }
        if (sud == true) {
            g2d.drawLine(x, y + costat, x + costat, y + costat);
        }
        if (oest == true) {
            g2d.drawLine(x, y, x, y + costat);
        }
        if (this.ocupada) {
            this.fitxa.paintComponent(g, x, y);
        }
    }

    //Getters i setters dels diferents atributs de la classe.
    public void setFitxa(Fitxa f) {
        this.fitxa = f;
        this.ocupada = true;
    }

    public void foraFitxa() {
        this.ocupada = false;
    }

    public boolean muradaNord() {
        return this.nord;
    }

    public boolean muradaEst() {
        return this.est;
    }

    public boolean muradaSud() {
        return this.sud;
    }

    public boolean muradaOest() {
        return this.oest;
    }
}
```

Fitxa

```
/**
 * Classe que conté els elements relacionats amb la fitxa i tots els atributs
 * d'aquesta.
 *´Ës la classe situada al element més baix de la nostra escala de abstracció.
 */
package projectefinallaberint;

import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

/**
 * @authors Toni Boutaour, Bernat Pericàs
 * @DNIs 43202227Q, 43212796M
 */
public class Fitxa {

    private int fitxaX = (int) Math.floor((Math.random()) *
Laberint.columnes);
    private int fitxaY = (int) Math.floor((Math.random()) * Laberint.files);

    public static String fitxa = Laberint.estilLab.getFitxaMoviment();
    public static String sortida = Laberint.estilLab.getFitxaSortida();

    private BufferedImage imatge;

    /**
     * Constructor que representa la imatge continguda al paràmetre String.
     *
     * @param s nom del fitxer
     */
    public Fitxa(String s) {
        //Actualutzació dels paràmetres SORTIDA i F
        sortida = Laberint.estilLab.getFitxaSortida();
        fitxa = Laberint.estilLab.getFitxaMoviment();
        try {
            imatge = ImageIO.read(new File(s));
        } catch (IOException ex) {
            System.out.println("Hi ha hagut un problema "
                    + "llegint la imatge de la fitxa");
        }
    }

    /**
     * S'ha realitzat la edició pixel a pixel de les fitxes.
     *
     * @param g objecte grafic
     * @param x coordenada x
     * @param y coordenada y
     */
    public void paintComponent(Graphics g, float x, float y) {
        g.drawImage(imatge, (int) x + 3, (int) y + 3, 80 - 35, 80 - 35, null);
    }
}
```

FitxerIn

```java
/**
 * Classe que emmagatzema el tractament dels fitxers d'entrada.
 * Aqui es poden veure tots els elements de creació i tancament de fitxers.
 */
package projectefinallaberint;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

/**
 * @authors Toni Boutaour, Bernat Pericàs
 * @DNIs 43202227Q, 43212796M
 */
public class FitxerIn {

    private FileReader fr;
    private BufferedReader br;

    /**
     * Mètode que obre els fluxes d'entrada per un fitxer
     *
     * @param nom del fitxer
     */
    public FitxerIn(String nom) {
        try {
            fr = new FileReader(nom);
            br = new BufferedReader(fr);
        } catch (FileNotFoundException ex) {
            System.out.println(ex.getLocalizedMessage());
        }
    }

    /**
     * Metode per a tancar els fluxes d'entrada
     */
    public void tancarFitxer() {
        try {
            fr.close();
            br.close();
        } catch (IOException ex) {
            System.out.println(ex.getLocalizedMessage());
        }
    }

    /**
     * Get del br
     *
     * @return BufferedReader del fitxer
     */
    public BufferedReader getBr() {
        return br;
    }
}
```

Missatge

```
/*
Classe usada per a mostrar missatges per pantalla en forma de finestres
emergents, és un Jdialog.
 */
package projectefinallaberint;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;

/**
 * @authors Toni Boutaour, Bernat Pericàs
 * @DNIs 43202227Q, 43212796M
 */
class Missatge extends JDialog {

    private JLabel jlbGuanyador;
    private JButton jbtOk;
//    private JButton jbtNewLab;
    private final String text;

    /**
     * Constructor de missatges
     * @param text String que es mostrarà per pantalla
     */
    public Missatge(String text) {
        this.text = text;
        initComponents();
        this.pack();
        this.setLocationRelativeTo(this);
        this.setResizable(false);
        this.setVisible(true);
        this.setDefaultCloseOperation(HIDE_ON_CLOSE);

        jbtOk.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
            }
        });
    }

    public void initComponents() {
        JPanel JpEtiqueta = new JPanel();
        jlbGuanyador = new JLabel(text, SwingConstants.CENTER);
        jlbGuanyador.setFont(new Font("arial", 1, 30));
        JpEtiqueta.setLayout(new BorderLayout());
        JpEtiqueta.add(jlbGuanyador, BorderLayout.NORTH);
        JPanel JpBotons = new JPanel();
        JpBotons.setLayout(new FlowLayout());
        jbtOk = new JButton("Ok");
        jbtOk.setPreferredSize(new Dimension(100, 50));
        JpBotons.add(jbtOk);
        JpEtiqueta.add(JpBotons, BorderLayout.SOUTH);
        getContentPane().add(JpEtiqueta);
    }
}
```

Tema

```
/**
 * Aquesta classe se realitza per tenir d'una forma organitzada l'estil del
 * laberint de manera que tots els elements d'estil estan emmagatzemats dins
un ARRAY i realitzar d'una forma més compacta l'edició de l'estil del
laberint.
 */
package projectefinallaberint;

import java.awt.Color;

/**
 * @authors Toni Boutaour, Bernat Pericàs
 * @DNIs 43202227Q, 43212796M
 */
public class Tema {

    private final Object[] estil = new Object[4];

    /**
     * Constructor de temes
     *
     * @param cF Color del Fons
     * @param cP Color de la Paret
     * @param fm nom de la Fitxa de Moviment
     * @param fS nom de la Fitxa de Sortida
     */
    public Tema(Color cF, Color cP, String fm, String fS) {
        estil[0] = (Object) cF;
        estil[1] = cP;
        estil[2] = fm;
        estil[3] = fS;
    }
    //Getters i setters necessaris
    public Color getColorFons() {
        return (Color) estil[0];
    }

    public Color getColorParet() {
        return (Color) estil[1];
    }

    public String getFitxaMoviment() {
        return (String) estil[2];
    }

    public String getFitxaSortida() {
        return (String) estil[3];
    }

    public void setColorFons(Color cF) {
        this.estil[0] = cF;
    }

    public void setColorParet(Color cP) {
        this.estil[1] = cP;
    }

    public void setFitxaMoviment(String s) {
        this.estil[2] = s;
    }

    public void setFitxaSortida(String s) {
        this.estil[3] = s;
    }
}
```