

Ampliación de Ingeniería del Software

Proyecto Buscaminas

Sergio Ibarra Méndez
Carlos Rodrigo Fernández Sánchez
Javier Andreu Gómez
Marcos Arquero Castillo

Campus Vicálvaro
URJC 2018

Estructura Organizativa

Tabla de Organización	
Actividad	Responsable
Calendario y diagramas	Sergio Ibarra Méndez
PGPS	Javier Andreu Gómez
PGCS	Carlos Rodrigo Fernández Sánchez
Versión de código mejorado	Marcos Arquero Castillo
Presentación Power Point	Todos los miembros del grupo

Herramientas Utilizadas



GitHub

Commits on Apr 20, 2018

Corregido error Experto ...



marcosarquero committed 4 days ago



7b25c52



Añadidos archivos de partida y mejores tiempos



marcosarquero committed 4 days ago



03bd60f



Borrador de version final



marcosarquero committed 4 days ago



12cae50



Commits on Apr 18, 2018

Guardar / cargar partida



marcosarquero committed 6 days ago



78be44f



GitHub

- ❖ Versión: AIS-B-D-04-0418
- ❖ Versión: AIS-B-P-04-0418

Gestión de Riesgos

- ❖ Problema con el modo experto y personalizado del código
- ❖ Falta de tiempo
- ❖ Falta de personal al principio del proyecto

Mantenimiento

- ❖ Mantenimiento correctivo
- ❖ Mantenimiento preventivo
- ❖ Mantenimiento estructural

Variables

```
public class Buscaminas extends JFrame implements ActionListener, MouseListener

    Timer t;
    JMenuBar menuBar;
    JMenu menu, submenu;
    JMenuItem tiempos, guardar, cargar;
    JRadioButtonMenuItem principiante, intermedio, experto, personalizado;
    ButtonGroup group;
    JButton b[][];
    JButton reiniciar;
    JPanel tablero;
    JPanel botonera;
    JTextField minasRestantes, tiempo;
    LinkedList<Dupla> mejoresPrincipiante, mejoresIntermedio, mejoresExperto;
    int nomines = 40;
    int restantes;
    int n = 16;
    int m = 16;
    int row;
    int column;
    int guesses[][];
    int perm[][];
    int[][] mines;
    boolean enabledBool[][];
    String tmp;
    boolean found = false;
    boolean allmines, personalizadoBool;
    int deltax[] = {-1, 0, 1, -1, 1, -1, 0, 1};
    int deltay[] = {-1, -1, -1, 0, 0, 1, 1, 1};
    double starttime, endtime, currenttime;
```


Constructor

```
public Buscaminas() {  
    //Inicializacion de variables  
    this.setDefaultCloseOperation(EXIT_ON_CLOSE);  
    t = new Timer(1000, reloj);  
    perm = new int[n][m];  
    allmines = false;  
    personalizadoBool = false;  
    restantes = nomines;  
    guesses = new int[n + 2][m + 2];  
    mines = new int[n + 2][m + 2];  
    b = new JButton[n][m];  
    this.setLayout(new BorderLayout()); //Layout del JFrame  
}
```

Constructor

```
//Creacion de la barra de menu con sus item y submenus
menuBar = new JMenuBar();
menu = new JMenu("Opciones");
menuBar.add(menu);
guardar = new JMenuItem("Guardar partida");
menu.add(guardar);
cargar = new JMenuItem("Cargar partida");
menu.add(cargar);
tiempos = new JMenuItem("Mejores tiempos");
menu.add(tiempos);
menu.addSeparator();
submenu = new JMenu("Dificultad");
group = new ButtonGroup();
principiante = new JRadioButtonMenuItem("Principiante (10x10, 10 minas)");
group.add(principiante);
submenu.add(principiante);
intermedio = new JRadioButtonMenuItem("Intermedio (16x16, 40 minas)");
intermedio.setSelected(true);
group.add(intermedio);
submenu.add(intermedio);
experto = new JRadioButtonMenuItem("Experto (32x16, 99 minas)");
group.add(experto);
submenu.add(experto);
personalizado = new JRadioButtonMenuItem("Personalizado");
group.add(personalizado);
submenu.add(personalizado);
menu.add(submenu);
this.setJMenuBar(menuBar);
```




Cargar partida

Mejores tiempos

Dificultad

... 37

Tiempo:

30

Principiante (10x10, 10 minas)

✓ Intermedio (16x16, 40 minas)

Experto (32x16, 99 minas)

Personalizado

2	1	1			1	2	2	1
---	---	---	--	--	---	---	---	---

2	x	1			2
---	---	---	--	--	---

3	2			2
---	---	--	--	---

2				1
---	--	--	--	---

3	1	1	1	1
---	---	---	---	---

Constructor

```
//Adicion de ActionListener a cada elemento
cargar.addActionListener(menus);
guardar.addActionListener(menus);
tiempos.addActionListener(menus);
principiante.addActionListener(dif);
intermedio.addActionListener(dif);
experto.addActionListener(dif);
personalizado.addActionListener(dif);

//Creacion de la botonera
reiniciar = new JButton("Reiniciar"); //Boton de reiniciar
reiniciar.addActionListener(this);
reiniciar.addMouseListener(this);
reiniciar.setEnabled(true);
minasRestantes = new JTextField(); //Campo de texto minas restantes
minasRestantes.setEnabled(false);
minasRestantes.setText(String.valueOf(restantes));
tiempo = new JTextField(); //Campo de texto de minas restantes
tiempo.setEnabled(false);
botonera = new JPanel();
botonera.setVisible(true);
botonera.setLayout(new GridLayout(1, 5));
botonera.add(reiniciar);
botonera.add(new JLabel("Minas restantes: "));
botonera.add(minasRestantes);
botonera.add(new JLabel("Tiempo: "));
botonera.add(tiempo);
```




Minas resta... 37

Tiempo:

23

[illegible]

Constructor

```
//Creacion tablero de minas m x n, con GridLayout
tablero = new JPanel();
tablero.setVisible(true);
tablero.setLayout(new GridLayout(m, n));

//Integracion de botonera y tablero en el JFrame
add(botonera, BorderLayout.PAGE_START);
add(tablero, BorderLayout.CENTER);
```

```
//Inicializacion de mines y guesses
for (int y = 0; y < m + 2; y++) {
    mines[0][y] = 3;
    mines[n + 1][y] = 3;
    guesses[0][y] = 3;
    guesses[n + 1][y] = 3;
}
for (int x = 0; x < n + 2; x++) {
    mines[x][0] = 3;
    mines[x][m + 1] = 3;
    guesses[x][0] = 3;
    guesses[x][m + 1] = 3;
}
}
```


ActionListeners

```
private ActionListener reloj = new ActionListener() {  
    //Cada segundo actualiza el tiempo transcurrido de partida  
    @Override  
    public void actionPerformed(ActionEvent ae) {  
        currenttime = System.nanoTime();  
        tiempo.setText(String.valueOf((int) ((currenttime - starttime) / 1000000000)));  
    }  
};
```

```

private ActionListener menus = new ActionListener() {
    //ActionListener para la barra de menu
    @Override
    public void actionPerformed(ActionEvent ae) {
        File f;
        FileInputStream fis;
        FileOutputStream fos;
        ObjectInputStream ois;
        ObjectOutputStream oos;
        JMenuItem current = (JMenuItem) ae.getSource();
        if (current == guardar) {
            // Guardamos los objetos importantes en el fichero "partida.obj"
            try {
                currenttime = System.nanoTime();
                enabledBool = new boolean[n][m];
                for (int y = 0; y < m; y++) {
                    for (int x = 0; x < n; x++) {
                        enabledBool[x][y] = b[x][y].isEnabled();
                    } //end inner for
                } //end for
                f = new File("partida.obj");
                fos = new FileOutputStream(f);
                oos = new ObjectOutputStream(fos);
                oos.writeObject(nomines);
                oos.writeObject(restantes);
                oos.writeObject(n);
                oos.writeObject(m);
                oos.writeObject(guesses);
                oos.writeObject(perm);
                oos.writeObject(mines);
                oos.writeObject(enabledBool);
                oos.writeObject(personalizadoBool);
                oos.writeObject((double) (currenttime - starttime));
                oos.close();
                fos.close();
            } catch (IOException ex) {
                Logger.getLogger(Buscaminas.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}

```



```

} else if (current == cargar) {
    //Leemos los objetos importantes del fichero "partida.obj"
    currenttime = System.nanoTime();
    try {
        f = new File("partida.obj");
        fis = new FileInputStream(f);
        ois = new ObjectInputStream(fis);
        nomines = (int) ois.readObject();
        restantes = (int) ois.readObject();
        n = (int) ois.readObject();
        m = (int) ois.readObject();
        guesses = (int[][] ois.readObject());
        perm = (int[][] ois.readObject());
        mines = (int[][] ois.readObject());
        enabledBool = (boolean[]) ois.readObject();
        personalizadoBool = (boolean) ois.readObject();
        starttime = currenttime - (double) ois.readObject();
        ois.close();
        fis.close();
    } catch (IOException ex) {
        Logger.getLogger(Buscaminas.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Buscaminas.class.getName()).log(Level.SEVERE, null, ex);
    }
    cargarBotones(); //Invoca el metodo que carga el nuevo tablero
    t.start();
} else if (current == tiempos) {
    cargarMejores(); // Invoca el metodo que carga los mejores tiempos del fichero "mejores.obj"
    mejoresTiempos(); // Muestra una ventana emergente con los mejores tiempos
} else {
    System.out.println("Error en boton. Menu principal");
    System.exit(-1);
}

```

```

private ActionListener dif = new ActionListener() {
    //ActionListener para el menu de dificultades
    @Override
    public void actionPerformed(ActionEvent ae) {
        JRadioButtonMenuItem current = (JRadioButtonMenuItem) ae.getSource();
        //Cada opcion del if-elseif setea los valores de n, m y nomines
        if (current == principiante) {
            personalizadoBool = false;
            n = 10;
            m = 10;
            nomines = 10;
            reinicio();
        } else if (current == intermedio) {
            personalizadoBool = false;
            n = 16;
            m = 16;
            nomines = 40;
            reinicio();
        } else if (current == experto) {
            personalizadoBool = false;
            n = 16;
            m = 32;
            nomines = 99;
            reinicio();
        } else if (current == personalizado) {
            //En este caso, preguntamos al usuario mediante una ventana emergente por los valores
            personalizadoBool = true;
            n = Integer.valueOf(JOptionPane.showInputDialog("Type number of rows: "));
            m = Integer.valueOf(JOptionPane.showInputDialog("Type number of columns: "));
            nomines = Integer.valueOf(JOptionPane.showInputDialog("Type number of mines: "));
            reinicio();
        } else {
            System.out.println("Error en boton. Menu principal");
            System.exit(-1);
        }
    }
}

```

cargarBotones()

```
public void cargarBotones() {  
    //Nos deshacemos del tablero anterior  
    tablero.setVisible(false);  
    remove(tablero);  
    //Cambiamos minas restantes  
    minasRestantes.setText(String.valueOf(restantes));  
    //Creamos el nuevo tablero  
    tablero = new JPanel();  
    tablero.setVisible(true);  
    tablero.setLayout(new GridLayout(m, n));  
    this.add(tablero, BorderLayout.CENTER);  
    b = new JButton[n][m];  
}
```

```

for (int y = 0; y < m; y++) {
    for (int x = 0; x < n; x++) {
        b[x][y] = new JButton(" ");
        b[x][y].addActionListener(this);
        b[x][y].addMouseListener(this);
        if (!enabledBool[x][y]) {
            /*Si el boton estaba "disabled" en la partida original, es que ya esta pulsado
            Por tanto, puede tener como valor un entero positivo o ' '
            */
            tmp = Integer.toString(perm[x][y]);
            if (perm[x][y] == 0) {
                tmp = " ";
            }
            b[x][y].setText(tmp);
            b[x][y].setEnabled(false);
        } else if (guesses[x + 1][y + 1] == 1) {
            // Si guesses vale 1 en esa posicion, es que ya lo habiamos marcado como mina
            b[x][y].setText("x");
            b[x][y].setEnabled(true);
            b[x][y].setBackground(Color.orange);
        } else {
            // En cualquier otro caso, la casilla esta sin explorar
            b[x][y].setText(" ");
            b[x][y].setEnabled(true);
            b[x][y].setBackground(null);
        }
        tablero.add(b[x][y]);
    } //end inner for
} //end for
pack();
tablero.setVisible(true);
setVisible(true);
}

```


mejoresTiempos()

```
public void mejoresTiempos() {
    /*Incluimos los mejores tiempos de cada categoria en el string texto, y lo mostramos
    Estan implementados como LinkedList de duplas (nombre, tiempo).
    */
    Dupla d;
    String texto;
    texto = "NIVEL PRINCIPIANTE\n";
    while (!mejoresPrincipiante.isEmpty()) {
        d = mejoresPrincipiante.pop();
        texto = texto + " " + d.getTiempo() + " segundos - " + d.getNombre() + "\n";
    }
    texto = texto + "\n NIVEL INTERMEDIO\n";
    while (!mejoresIntermedio.isEmpty()) {
        d = mejoresIntermedio.pop();
        texto = texto + " " + d.getTiempo() + " segundos - " + d.getNombre() + "\n";
    }
    texto = texto + "\n NIVEL EXPERTO\n";
    while (!mejoresExperto.isEmpty()) {
        d = mejoresExperto.pop();
        texto = texto + " " + d.getTiempo() + " segundos - " + d.getNombre() + "\n";
    }
    JOptionPane.showMessageDialog(null, texto);
}
```

cargarMejores()

```
public void cargarMejores() {
    /*Cargamos los mejores tiempos de "mejores.obj". Estan implementados
    como LinkedList de duplas (nombre, tiempo).
    */
    File f2;
    FileInputStream fis2;
    ObjectInputStream ois2;
    mejoresPrincipiante = new LinkedList();
    mejoresIntermedio = new LinkedList();
    mejoresExperto = new LinkedList();
    try {
        f2 = new File("mejores.obj");
        fis2 = new FileInputStream(f2);
        ois2 = new ObjectInputStream(fis2);
        mejoresPrincipiante = (LinkedList) ois2.readObject();
        mejoresIntermedio = (LinkedList) ois2.readObject();
        mejoresExperto = (LinkedList) ois2.readObject();
        ois2.close();
        fis2.close();
    } catch (IOException ex) {
        Logger.getLogger(Buscaminas.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Buscaminas.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```


actionListener this

```
public void actionPerformed(ActionEvent e) {  
    //ActionListener del JFrame  
    JButton current = (JButton) e.getSource();  
    found = false;  
    if (current == reiniciar) { //Si pulsamos reiniciar, se reinicia el tablero  
        System.out.println("REINICIO");  
        reinicio();  
    } else { //Si es otro boton, buscamos cual es con este for anidado  
        for (int y = 0; y < m; y++) {  
            for (int x = 0; x < n; x++) {  
                JButton t = b[x][y];  
                if (t == current) {  
                    row = x;  
                    column = y;  
                    found = true;  
                }  
            } //end inner for  
        } //end for  
        if (!found) {  
            System.out.println("didn't find the button, there was an error ");  
            System.exit(-1);  
        }  
        Component temporaryLostComponent = null;  
    }
```

actionListener this

```
,
Component temporaryLostComponent = null;
if (b[row][column].getBackground() == Color.orange) {
    //Si esta marcado como mina (BackgroundColor orange), no hacemos nada
    return;
} else if (mines[row + 1][column + 1] == 1) {
    //Si habia una mina, mostramos mensaje y reiniciamos tablero
    JOptionPane.showMessageDialog(temporaryLostComponent, "You set off a Mine!!!!.");
    reinicio();
} else {
    //Si no habia mina, revelamos la celda
    tmp = Integer.toString(perm[row][column]);
    if (perm[row][column] == 0) {
        tmp = " ";
    }
    b[row][column].setText(tmp);
    b[row][column].setEnabled(false);
    checkifend(); // Comprueba si hemos ganado
    if (perm[row][column] == 0) {
        scan(row, column); //Si tiene 0 minas alrededor, revelamos sus colaterales
        checkifend();
    }
}
}
}
```


reinicio()

```
public void reinicio() {
    //Reseteamos las variables
    allmines = false;
    restantes = nomines;
    minasRestantes.setText(String.valueOf(restantes));
    b = new JButton[n][m];
    perm = new int[n][m];
    guesses = new int[n + 2][m + 2];
    mines = new int[n + 2][m + 2];
    tablero.setVisible(false);
    this.remove(tablero);
    tablero = new JPanel();
    tablero.setVisible(true);
    tablero.setLayout(new GridLayout(m, n));
    add(tablero, BorderLayout.CENTER);

    //Construimos el tablero igual que en el constructor inicial
    //Primero inicializamos mines y guesses
    for (int y = 0; y < m + 2; y++) {
        mines[0][y] = 3;
        mines[n + 1][y] = 3;
        guesses[0][y] = 3;
        guesses[n + 1][y] = 3;
    }
    for (int x = 0; x < n + 2; x++) {
        mines[x][0] = 3;
        mines[x][m + 1] = 3;
        guesses[x][0] = 3;
        guesses[x][m + 1] = 3;
    }
}
```

reinicio()

```
do {
    int check = 0;
    for (int y = 1; y < m + 1; y++) {
        for (int x = 1; x < n + 1; x++) {
            mines[x][y] = 0;
            guesses[x][y] = 0;
        }
    }
    for (int x = 0; x < nomines; x++) {
        mines[(int) (Math.random() * (n) + 1)][(int) (Math.random() * (m) + 1)] = 1;
    }
    for (int x = 0; x < n; x++) {
        for (int y = 0; y < m; y++) {
            if (mines[x + 1][y + 1] == 1) {
                check++;
            }
        }
    }
    if (check == nomines) {
        allmines = true;
    }
} while (allmines == false);
```

reinicio()

```
//Inicializamos perm
for (int y = 0; y < m; y++) {
    for (int x = 0; x < n; x++) {
        if ((mines[x + 1][y + 1] == 0) || (mines[x + 1][y + 1] == 1)) {
            perm[x][y] = perimcheck(x, y);
        }
        b[x][y] = new JButton(" ");
        b[x][y].addActionListener(this);
        b[x][y].addMouseListener(this);
        tablero.add(b[x][y]);
        b[x][y].setEnabled(true);
    } //end inner for
} //end for
for (int y = 0; y < m + 2; y++) {
    for (int x = 0; x < n + 2; x++) {
        System.out.print(mines[x][y]);
    }
    System.out.println("");
}
starttime = System.nanoTime();
t.start();
```


checkifend()

```
public void checkifend() {
    //Comprueba si hemos ganado la partida (si solo quedan sin explorar las casillas con mina)
    int check = 0;
    for (int y = 0; y < m; y++) {
        for (int x = 0; x < n; x++) {
            if (b[x][y].isEnabled()) {
                check++;
            }
        }
    }
    if (check == nomines) {
        //Paramos el cronometro
        endtime = System.nanoTime();
        t.stop();
        //Calculamos la duracion de la partida
        int time = (int) ((currenttime - starttime) / 1000000000);
        //Si era dificultad personalizada, mostramos el tiempo invertido
        if (personalizadoBool) {
            JOptionPane.showMessageDialog(null, "Congratulations you won!!! It took you " + time + " seconds!");
        } else {
            //Si era otra dificultad, ademas guardamos el tiempo en la lista de mejores
            String input = null;
            input = JOptionPane.showInputDialog("Congratulations you won!!! It took you " + time + " seconds! Type
            if (input != null) {
                cargarMejores(); //Cargamos los mejores tiempos de "mejores.obj"
            }
        }
    }
}
```

checkifend()

```
switch (nomines) {
    /*En cada caso, añadimos el tiempo actual a la lista de mejores, la ordenamos
    y en caso de que tenga tamaño > 10, borramos el peor tiempo
    */
    case 10: //10 minas es nivel principiante
        mejoresPrincipiante.add(new Dupla(time, input));
        Collections.sort(mejoresPrincipiante);
        if (mejoresPrincipiante.size() > 10) {
            mejoresPrincipiante.removeLast();
        }
        break;
    case 40: //40 minas es nivel intermedio
        mejoresIntermedio.add(new Dupla(time, input));
        Collections.sort(mejoresIntermedio);
        if (mejoresIntermedio.size() > 10) {
            mejoresIntermedio.removeLast();
        }
        break;
    case 99: //99 minas es nivel experto
        mejoresExperto.add(new Dupla(time, input));
        Collections.sort(mejoresExperto);
        if (mejoresExperto.size() > 10) {
            mejoresExperto.removeLast();
        }
        break;
}
```

checkifend()

```
try {
    //Finalmente, guardamos las listas de mejores tiempos en "mejores.obj"
    File f2 = new File("mejores.obj");
    FileOutputStream fos2 = new FileOutputStream(f2);
    ObjectOutputStream oos2 = new ObjectOutputStream(fos2);
    oos2.writeObject(mejoresPrincipiante);
    oos2.writeObject(mejoresIntermedio);
    oos2.writeObject(mejoresExperto);
    oos2.close();
    fos2.close();
} catch (IOException ex) {
    Logger.getLogger(Buscaminas.class.getName()).log(Level.SEVERE, null, ex);
}
mejoresTiempos();
}
```