

Buscaminas

Susana
Olga
Miguel

Sprint 1

Hu2 : Generar Buscaminas del Nivel Seleccionado.

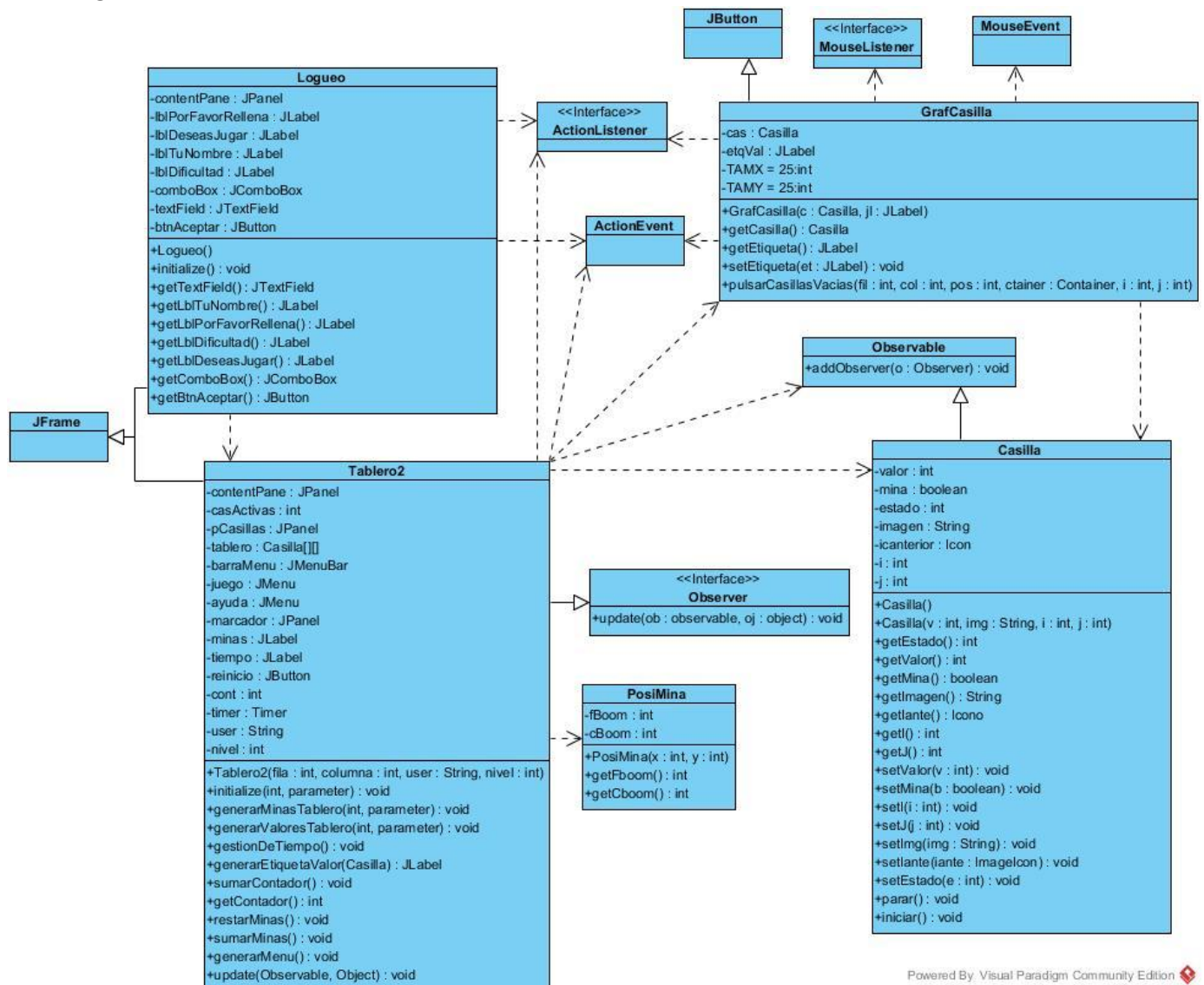
Generar un Buscaminas con el nivel solicitado y mostrar en pantalla el buscaminas.

Hu3 : Descubrir Casilla.

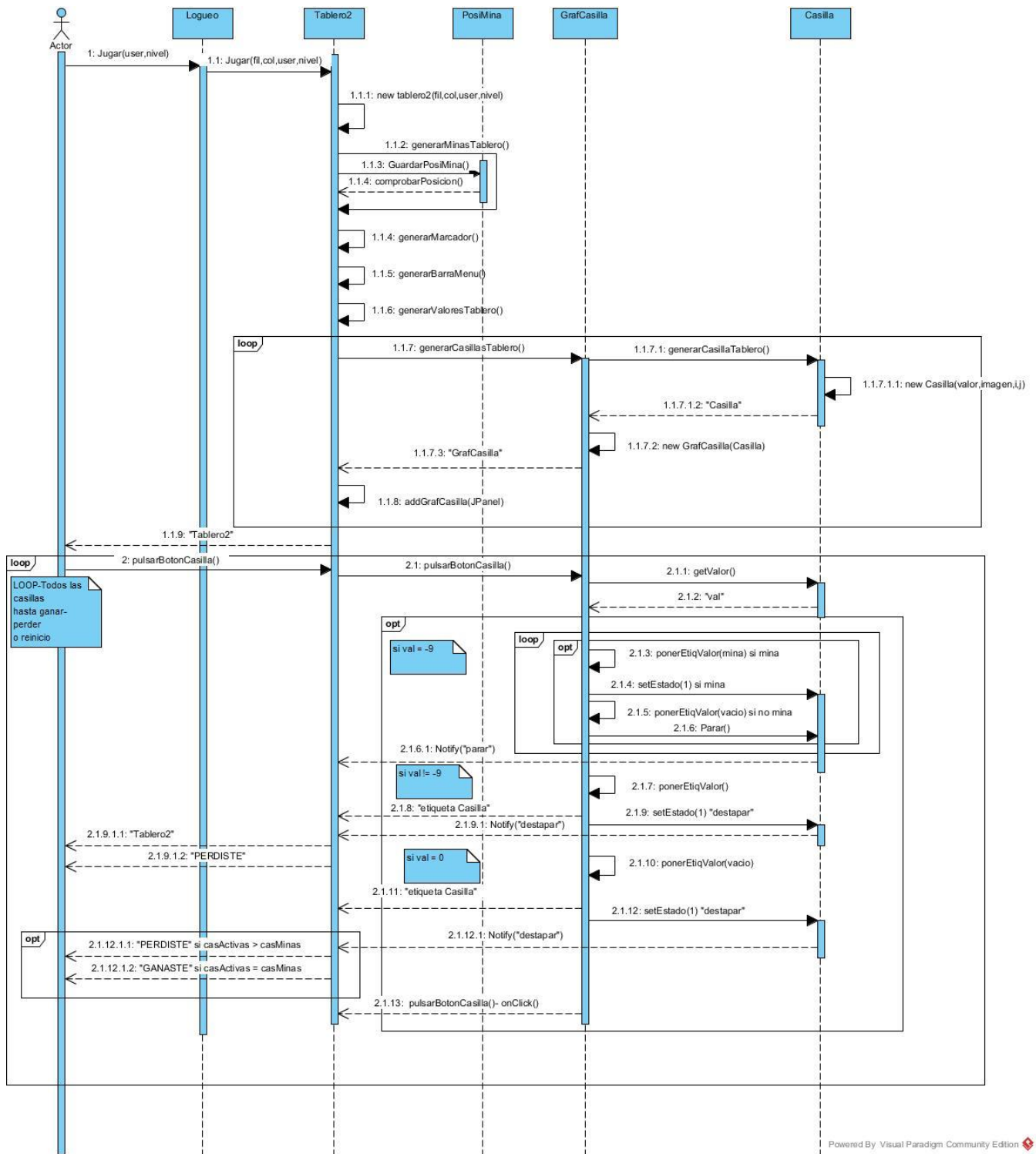
El usuario puede descubrir una casilla del buscaminas, pinchando sobre ella. Si después de esta acción sólo quedan cerradas las casillas con mina, la partida finaliza con éxito. Si en la Casilla hay una mina el juego finaliza con fracaso. Además, si la casilla descubierta está vacía (no contiene una Mina y no es vecina de ninguna casilla con mina), se abrirán todas las casillas vacías de su alrededor. También se abrirán recursivamente las casillas de alrededor de las que se van descubriendo hasta alcanzar casillas que tengan algún vecina con mina.

1. Diseño del modelo (Diagrama de clases y Diagramas de secuencia)

- UML

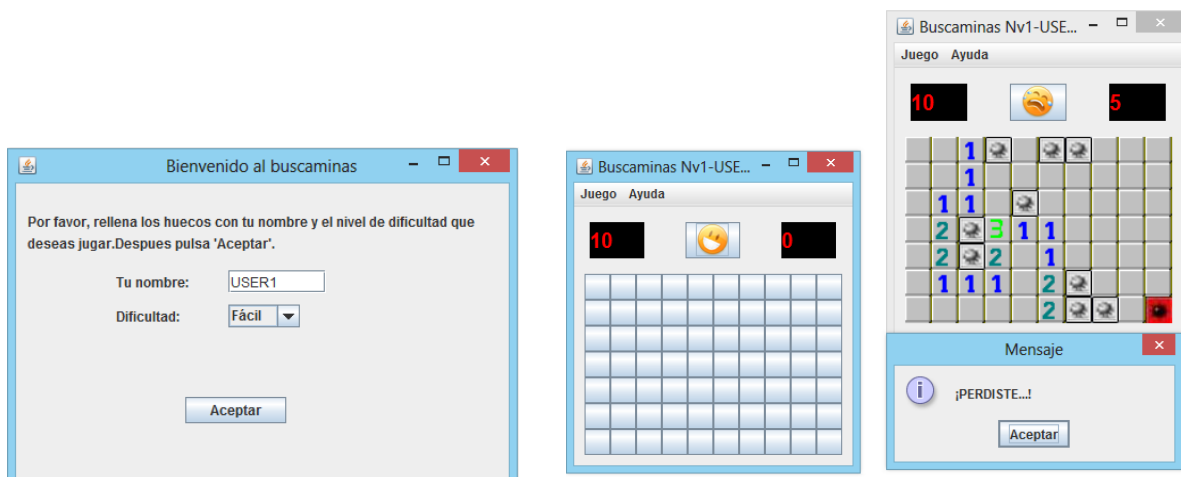


- Diagrama de secuencia.

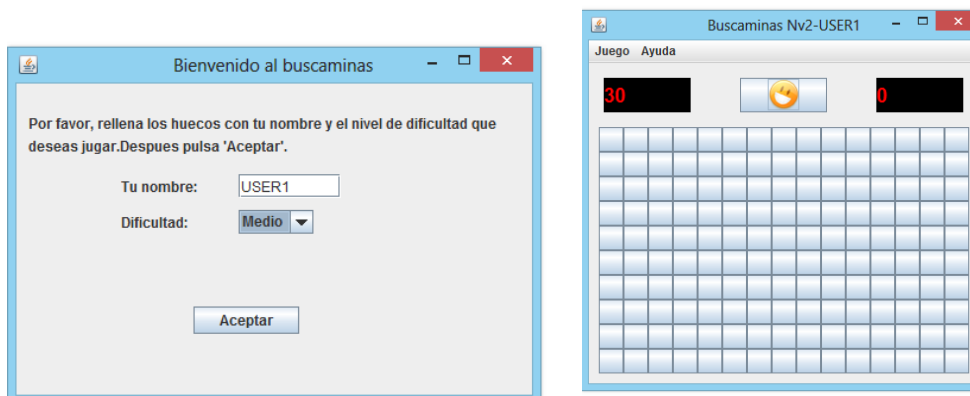


2. Diseño de la Interfaz Gráfica.

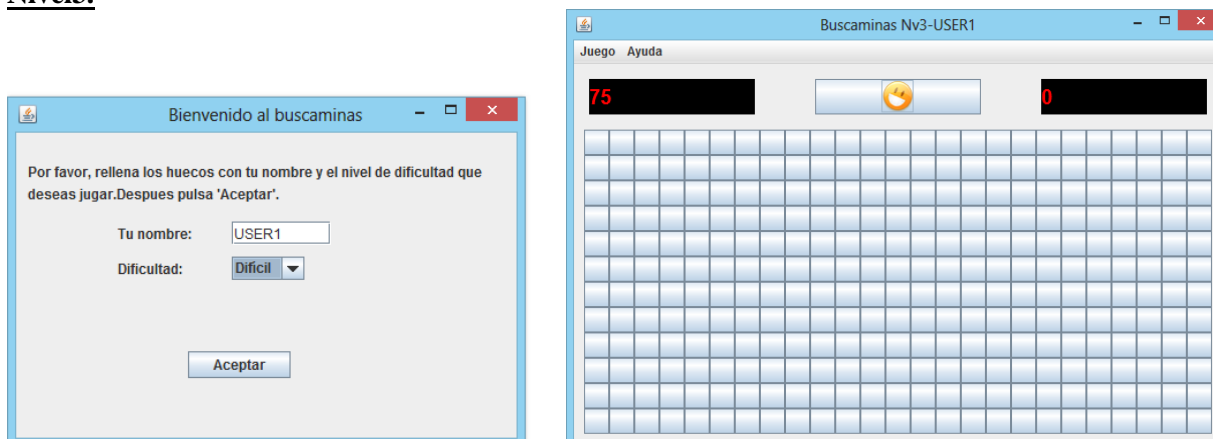
Nivel1



















Nivel2.



Nivel3.



Imágenes.

	0.png		9.png
	1.png		9R.png
	2.png		Perdio.png
	3.png		nueva.png
	4.png		gano.png
	5.png		Boom.mp3
	6.png		b1.png
	7.png		
	8.png		

3. Prototipo.

Se implementa un prototipo tomando los datos de usuario y nivel desde Consola y sin etiquetas graficas, solamente sacando el valor de la casillas en formato texto desde la pantalla.

Las clases intervinientes son las que se muestran en la Figura 1, y en la Figura 2 se ve en panel de juego con los datos de inicio introducidos desde Consola.

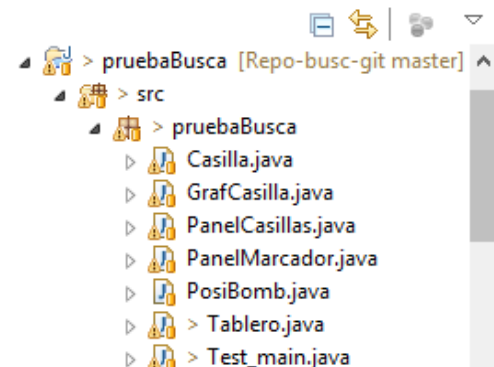


Figura 1

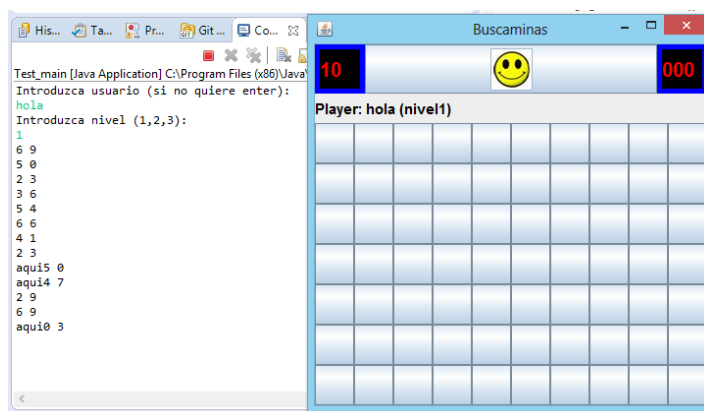


Figura 2

En la figura 3 vemos una evolución del juego, se ha pulsado la (0,0) generando múltiples clicks de las casillas vecinas vacías hasta encontrarse con las minas, posteriormente se ha pulsado una mina y el juego ha terminado a los 3 segundos, con cara de “no felicidad”, y con las casillas no destapadas en color verde.



Figura 3.

El prototipo esta implementado en el repositorio [ssh://git@github.com/sabascal/Buscaminas1.git](ssh://git@github.com:sabascal/Buscaminas1.git)

Codigo del prototipo.

```
package pruebaBusca;
import java.awt.event.*;
import java.awt.*;

import javax.swing.*;
import java.util.*;

public class Casilla extends Observable{
    private int valor;
    private int x;
    private int y;
    private boolean bandera;
    private boolean bomba;
    private String imagen;
    private Icon icanterior;
    private boolean oculta;
    private boolean primera;

    public Casilla (int pV, int pX, int pY, String plmg){
        bandera = false;
        bomba = false;
        oculta = true;
        if (pV == -9){
            bomba = true;
        }
        valor = pV;
        x = pX;
        y = pY;
        imagen = plmg;
        icanterior = new ImageIcon(plmg);
        primera = false;
    }

    public int getValor(){
        return valor;
    }

    public int getFila(){
        return x;
    }

    public int getColum(){
        return y;
    }

    public String getImagen(){
```

```

        return imagen;
    }
    public boolean getBandera(){
        return bandera;
    }
    public boolean getBomba(){
        return bomba;
    }
    public Icon getIcante(){
        return icanterior;
    }
    public boolean getOculto(){
        return oculta;
    }
}
    public void setX(int x){
        this.x = x;
    }
    public void setY(int y){
        this.y = y;
    }
    }
    public void setImg (String plmg){
        icanterior = new ImageIcon(imagen);
        imagen = plmg;
    }
    public void setValor(int v){
        valor = v;
    }
    }
    public void setBandera(boolean b){
        bandera = b;
        setChanged();
        this.notifyObservers(b);
    }
    public void setBomba(boolean b){
        bomba = b;
    }
    }
    public void setIcante(ImageIcon img){
        icanterior = img;
    }
    }
    public boolean getPrimera(){
        return primera;
    }
    }
    public void setPrimera(boolean b){
        primera = b;
        setChanged();
        this.notifyObservers("iniciar");
    }
    }

    public void setPrimera(boolean b){
        primera = b;
    }
    }
    public void setOculto(){
        oculta = false;
    }
    }
    public void parar(){
        setChanged();
        this.notifyObservers("parar");
    }
    }

}

package pruebaBusca;
import java.awt.event.*;
import java.awt.*;

import javax.swing.*;

public class GrafCasilla extends JButton{

    private Casilla c;

    public GrafCasilla (Casilla cas){
        c = cas;
        setName("C"+c.getColum()/41+c.getFila()/40);
    }
}

```

```

setBounds(c.getFila(), c.getColum(), 40, 41);
setVisible(true);
setEnabled(true);
setPreferredSize(new Dimension(40,41));
setFont(new Font("Dialog", Font.BOLD, 11));
addActionListener(new manejadorBoton());
addMouseListener(new botonesMousse());
}

public Casilla getCasilla(){
    return c;
}

public class manejadorBoton implements ActionListener{

    public void actionPerformed(ActionEvent e) {
        String sv = "";
        String nomb = null;
        GrafCasilla csPrim = null;
        sv = Integer.toString(c.getValor());
        int col1 = 0;
        int col2 = 0;
        int fil1 = 0;
        int fil2 = 0;
        System.out.println("Botón pulsado");
        //bc.setLabel("8");
        if (!c.getPrimera()){
            c.setPrimeralnic(true);
            Component listCasillas [] = ((JPanel) getParent()).getComponents();
            for (int i = 0; i < listCasillas.length; i++){
                ((GrafCasilla)listCasillas[i]).getCasilla().setPrimera(true);
            }
        }
        if (!c.getBandera()){
            if (c.getValor() == -9){
                //setOpaque(false);
                setIcon(new ImageIcon("bomba.jpg"));
                setBackground(Color.RED);
                setEnabled(false);

                Component listCas [] = ((JPanel) getParent()).getComponents();
                for (int i = 0; i < listCas.length; i++){
                    if (((GrafCasilla)listCas[i]).getCasilla().getValor() == -9){
                        ((GrafCasilla)listCas[i]).setOpaque(true);
                        ((GrafCasilla)listCas[i]).setBackground(Color.RED);
                        ((GrafCasilla)listCas[i]).setForeground(Color.RED);
                        ((GrafCasilla)listCas[i]).setIcon(new ImageIcon("bomba.jpg"));
                        ((GrafCasilla)listCas[i]).setEnabled(false);
                    }
                    else{
                        ((GrafCasilla)listCas[i]).setBackground(Color.GREEN);
                        ((GrafCasilla)listCas[i]).setEnabled(false);
                    }
                }
            }
            c.parar();
        }
        else{
            setOpaque(false);
            if (sv.equals("0")){
                sv = "";
                // setEnabled(false);
                // setFocusable(false);
                setText(sv);

                setEnabled(false);

                if (e.getModifiers() != 0){
                    csPrim = (GrafCasilla) e.getSource();
                }
                GrafCasilla cs = (GrafCasilla) e.getSource();
                JPanel p = (JPanel) cs.getParent();
            }
        }
    }
}

```



```

int nv = ((PanelCasillas)p).getNivel();
int f1 = 0;
int c1 = 0;
if (nv == 1){
    f1 = 7;
    c1 = 10;
}
if (nv == 2){
    f1 = 10;
    c1 = 15;
}
if (nv == 3){
    f1 = 12;
    c1 = 25;
}
// cs.setX(((Casilla) e.getSource()).getX()+41);
// cs.setY(((Casilla) e.getSource()).getY()+41);

/* if (cs.getName().charAt(2) != '0' ||
    cs.getName().charAt(2) != '9' ||
    cs.getName().charAt(1) != '0' ||
    cs.getName().charAt(1) != '7'){*/
int casCorrente = p.getComponentZOrder(cs);
int casVeciDrcha = casCorrente + f1;
int casVecilzq = casCorrente - f1;
int casVeciSup = casCorrente - 1;
int casVeciInf = casCorrente + 1;
int casVeciSupl = casCorrente - f1 - 1;
int casVeciSupDr = casCorrente + f1 - 1;
int casVeciInfl = casCorrente - f1 + 1;
int casVeciInfDr = casCorrente + f1 + 1;
boolean dere = false;
boolean izq = false;
boolean izsup = false;
boolean izinf = false;
boolean drsup = false;
boolean drinf = false;
boolean arriba = false;
boolean abajo = false;
col1 = Integer.parseInt(String.valueOf(cs.getName().charAt(2)));
fil1 = Integer.parseInt(String.valueOf(cs.getName().charAt(1)));
if (casVeciDrcha >= 0 && casVeciDrcha < f1*c1){
    cs = (GrafCasilla) p.getComponent(casVeciDrcha);
    col2 = Integer.parseInt(String.valueOf(cs.getName().charAt(2)));
}
if (col2 == col1 + 1)
if (cs.getCasilla().getValor() != -9 && cs.isEnabled())
    cs.doClick();

if (casVecilzq >= 0 && casVecilzq < f1*c1) {
    cs = (GrafCasilla) p.getComponent(casVecilzq);
    col2 = Integer.parseInt(String.valueOf(cs.getName().charAt(2)));
}
if (col2 == col1 - 1)
if (cs.getCasilla().getValor() != -9 && cs.isEnabled())
    cs.doClick();

if (casVeciSup >= 0 && casVeciSup < f1*c1) {
    cs = (GrafCasilla) p.getComponent(casVeciSup);
    fil2 = Integer.parseInt(String.valueOf(cs.getName().charAt(1)));
}
if (fil2 == fil1 - 1)
if (cs.getCasilla().getValor() != -9 && cs.isEnabled())
    cs.doClick();

if (casVeciInf >= 0 && casVeciInf < f1*c1) {
    cs = (GrafCasilla) p.getComponent(casVeciInf);
    fil2 = Integer.parseInt(String.valueOf(cs.getName().charAt(1)));
}
if (fil2 == fil1 + 1)
if (cs.getCasilla().getValor() != -9 && cs.isEnabled())
    cs.doClick();

if (casVeciSupDr >= 0 && casVeciSupDr < f1*c1) {

```

```

        cs = (GrafCasilla) p.getComponent(casVeciSupDr);
        fil2 = Integer.parseInt(String.valueOf(cs.getName().charAt(1)));
        col2 = Integer.parseInt(String.valueOf(cs.getName().charAt(2)));
    }
    if (fil2 == fil1 - 1 && col2 == col1 + 1 )
    if (cs.getCasilla().getValor() != -9 && cs.isEnabled())
        cs.doClick();

    if (casVeciSupl2 >= 0 && casVeciSupl2 < f1*c1) {
        cs = (GrafCasilla) p.getComponent(casVeciSupl2);
        fil2 = Integer.parseInt(String.valueOf(cs.getName().charAt(1)));
        col2 = Integer.parseInt(String.valueOf(cs.getName().charAt(2)));
    }
    if (fil2 == fil1 - 1 && col2 == col1 - 1)
    if (cs.getCasilla().getValor() != -9 && cs.isEnabled())
        cs.doClick();

    if (casVeciInfDr >= 0 && casVeciInfDr < f1*c1){
        cs = (GrafCasilla) p.getComponent(casVeciInfDr);
        fil2 = Integer.parseInt(String.valueOf(cs.getName().charAt(1)));
        col2 = Integer.parseInt(String.valueOf(cs.getName().charAt(2)));
    }
    if (fil2 == fil1 + 1 && col2 == col1 + 1)
    if (cs.getCasilla().getValor() != -9 && cs.isEnabled())
        cs.doClick();

    if (casVeciInfLz >= 0 && casVeciInfLz < f1*c1){
        cs = (GrafCasilla) p.getComponent(casVeciInfLz);
        fil2 = Integer.parseInt(String.valueOf(cs.getName().charAt(1)));
        col2 = Integer.parseInt(String.valueOf(cs.getName().charAt(2)));
    }
    if (fil2 == fil1 + 1 && col2 == col1 - 1)
    if (cs.getCasilla().getValor() != -9 && cs.isEnabled())
        cs.doClick();
    //}
    System.out.println(getX()+" "+getY()+" "+e.toString());
    System.out.println(getX()+" "+getY()+" "+e.toString()+
        " "+e.getSource());

    }
    else{
        setText(sv);
        setEnabled(false);
        if (e.getModifiers() == 0){
            csPrim.doClick();
        }
    }
}
setFocusable(false);
getCasilla().setOculta();

}

}

public class botonesMousse implements MouseListener{

    public void mousePressed(MouseEvent me) {
        if (isEnabled()){
            if ((me.getModifiers() & InputEvent.BUTTON3_MASK) == InputEvent.BUTTON3_MASK) {
                if (!c.getBanderita()){
                    c.setlante((Imagelcon)getIcon());;
                    setIcon(new Imagelcon("banderaBusc.jpg"));
                    c.setBanderita(true);
                }

                else{
                    setIcon((Imagelcon)c.getIcon());;
                    c.setBanderita(false);
                }
            }
        }
    }

    public void mouseReleased(MouseEvent me) {
    }
}

```

```

        public void mouseExited(MouseEvent me) {
        }
        public void mouseEntered(MouseEvent me) {
        }
        public void mouseClicked(MouseEvent me) {
        }
    }

}

package pruebaBusca;
import java.util.Random;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.*;

import javax.swing.*;

import pruebaBusca.GrafCasilla.manejadorBoton;

public class PanelCasillas extends JPanel {
    private int nivel;

    public PanelCasillas (int n){
        nivel = n;
        //super (new BorderLayout());
        //jp = new JPanel();
        //jp.setLayout(new BorderLayout(10, 7 ));
        int cb = 0;
        int f = 0;
        int c = 0;
        switch (n){
            case 1:
                f = 7;
                c = 10;

                break;
            case 2:
                f= 10;
                c= 15;

                break;
            case 3:
                f=12;
                c=25;

                break;
            default:
                break;
        }
        cb = c*n;
        Random r = new Random();
        int vf = 0;
        int vc = 0;
        int fila = 0;
        int colum = 0;

        PosiBomb [] pb = new PosiBomb[cb];
        for (int s = 0; s < cb; s++){
            pb[s]= new PosiBomb(0,0);
        }
        int ipb = 0;

        int [][] cas = new int [f][c];
        for (int k = 0; k < cb; k++){
            vf = r.nextInt(f);
            vc = r.nextInt(c);
            System.out.println (vf+" "+vc);
            int l = 0;
            while (l < cb){
                PosiBomb psb = new PosiBomb(vf,vc);
            }
        }
    }
}

```

```

        if ((pb [l].getFbomb() == psb.getFbomb()) &&
            (pb [l].getCbomb() == psb.getCbomb())) {
            vf = r.nextInt(f);
            vc = r.nextInt(c);
            l = 0;
            System.out.println ("aqui"+vf+" "+vc);
        }
        else
            l = l + 1;
    }
    cas [vf][vc] = -9;
    pb [ipb] = new PosiBomb(vf,vc);
    ipb = ipb + 1;

    fila = vf+1;
    colum = vc+1;
    if (fila < f && colum < c && fila >=0 && colum >=0)
        if (cas[fila][colum] != -9)
            cas [fila][colum] = cas [fila][colum] +1;

    if (fila < f && vc < c && fila >=0 && vc>=0)
        if (cas[fila][vc] != -9)
            cas [fila][vc] = cas [fila][vc] +1;

    if (vf < f && colum < c && vf >=0 && colum>=0)
        if (cas[vf][colum] != -9)
            cas [vf][colum] = cas [vf][colum] +1;

    fila = vf -1;
    if (fila < f && vc < c && fila >=0 && vc>=0)
        if (cas[fila][vc] != -9)
            cas [fila][vc] = cas [fila][vc] +1;

    colum = vc -1;
    if (vf < f && colum < c && vf >=0 && colum>=0)
        if (cas[vf][colum] != -9)
            cas [vf][colum] = cas [vf][colum] +1;

    fila = vf -1;
    colum = vc - 1;
    if (fila < f && colum < c && fila >=0 && colum>=0)
        if (cas[fila][colum] != -9)
            cas [fila][colum] = cas [fila][colum] +1;

    fila = vf +1;
    colum = vc -1;
    if (fila < f && colum < c && fila >=0 && colum>=0)
        if (cas[fila][colum] != -9)
            cas [fila][colum] = cas [fila][colum] +1;

    fila = vf -1;
    colum = vc +1;
    if (fila < f && colum < c && fila >=0 && colum>=0)
        if (cas[fila][colum] != -9)
            cas [fila][colum] = cas [fila][colum] +1;

    }

    //jp.setLayout(new GridLayout(10,7000));
    //setLayout(new GridBagLayout());
    //GridLayout g = new GridLayout(f,c);
    Dimension d = new Dimension (40*c,41*f);
    setPreferredSize(d);
    //FlowLayout flow = new FlowLayout();

    setLayout(null);

    Casilla c1;
    GrafCasilla gc1;
    for (int i= 0; i < c; i++)
        for (int j = 0; j < f; j++){
            c1 = new Casilla (cas[j][i],i*40,j*41,"");

```

```

        gc1 = new GrafCasilla(c1);
        add(gc1, null);

    }
    setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
    //jp.setVisible(true);

}
public int getNivel(){
    return nivel;
}

}

package pruebaBusca;

import javax.management.timer.Timer;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import java.awt.*;
import java.awt.event.*;

import javax.swing.border.*;
import javax.swing.*;

import java.util.*;

public class PanelMarcador extends JPanel implements Observer{
    //private static final long serialVersionUID = 8313442043392839053L;
    private int nivel;
    private JButton reinicio;
    private JLabel tminas;
    private JLabel ttiempo;
    private int segundos;
    private javax.swing.Timer tim;
    public PanelMarcador(int n){
        nivel = n;
        int m = 0;
        if (n == 1){
            m = n * 10;
        }
        if (n == 2){
            m = n * 15;
        }
        if (n == 3){
            m = n * 25;
        }
        tminas = new JLabel(String.valueOf(m));
        ttiempo = new JLabel("000");
        reinicio = new JButton();
        ttiempo.setOpaque(true);
        tminas.setOpaque(true);
        setPreferredSize(new Dimension(40*10, 50));
        //setLayout(new GridLayout(1,3,30*n,1));
        setLayout(new BorderLayout());
        add(tminas, BorderLayout.WEST);
        add(reinicio, BorderLayout.CENTER);
        add(ttiempo, BorderLayout.EAST);

        reinicio.setIcon(new ImageIcon("cfeliz.jpg"));
        reinicio.setEnabled(true);
        reinicio.setPreferredSize(new Dimension(41,41));
        tminas.setFont(new Font("Dialog", Font.BOLD, 20));
        tminas.setEnabled(true);
        ttiempo.setFont(new Font("Dialog", Font.BOLD, 20));
        ttiempo.setEnabled(true);
        Border brd = BorderFactory.createLineBorder(Color.BLUE, 5);
        ttiempo.setBorder(brd);
        tminas.setBorder(brd);
        tminas.setPreferredSize(new Dimension(51,51));
    }
}

```

```

        ttiempo.setPreferredSize(new Dimension(51,51));
        ttiempo.setForeground(Color.RED);
        ttiempo.setBackground(Color.BLACK);
        tminas.setForeground(Color.RED);
        tminas.setBackground(Color.BLACK);
        segundos = 0;
        tim = new javax.swing.Timer (1000, new ActionListener(){
            public void actionPerformed(ActionEvent evt) {
                setTiempo();
            }
        });

        reinicio.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent evt){
                System.out.println("REINICIAR JUEGO....");
                String jug = ((Tablero)(getParent().getParent().getParent().getParent())).getJugador();
                ((Tablero)(getParent().getParent().getParent().getParent())).cerrar();

                Tablero t = new Tablero(n);
                t.setJugador(jug);
            }
        });

    }

    public int getMinas(){
        return Integer.parseInt(tminas.getText());
    }

    public int getTiempo(){
        return Integer.parseInt(ttiempo.getText());
    }

    public void sumaMinas(){
        tminas.setText(String.valueOf(Integer.parseInt(tminas.getText())+1));
    }

    public void restaMinas(){
        tminas.setText(String.valueOf(Integer.parseInt(tminas.getText())-1));
    }

    public void setTiempo(){
        segundos = segundos + 1;
        ttiempo.setText(String.valueOf(segundos));
    }

    public void evaluarJuego(){
        int co = 0;
        int cb = 0;
        Component lCas []=((JPanel)(getParent()).getComponent(2)).getComponents();
        for (int i = 0; i < lCas.length; i++){
            if (((GrafCasilla)lCas[i]).getCasilla().getBandera()){
                cb = cb + 1;
            }
            if (((GrafCasilla)lCas[i]).getCasilla().getOculto()){
                co = co + 1;
            }
        }

        if (Integer.parseInt(tminas.getText()) != 0 &&
            !(Integer.parseInt(tminas.getText()) + cb == 10 && co == 0) &&
            Integer.parseInt(tminas.getText()) != co
        )
            reinicio.setIcon(new ImageIcon("Criste.jpg"));
    }

    public void update (Observable o, Object arg){
        if (arg.equals("iniciar")){
            tim.start();
        }
        else{
            if (arg.equals("parar")){
                tim.stop();
                evaluarJuego();
            }
            else
                if ((boolean)arg == true)
                    restaMinas();
                else
                    sumaMinas();
        }
    }
}

```

```

}

package pruebaBusca;

public class PosiBomb {

    private int fBomb;
    private int cBomb;
    public PosiBomb(int f, int c){
        fBomb = f;
        cBomb = c;
    }
    public int getFbomb(){
        return fBomb;
    }
    public int getCbomb(){
        return cBomb;
    }
}

public class Tablero extends JFrame{
    private int nivel;
    private PanelMarcador pm;
    private JLabel jugador;
    private PanelCasillas pc;
    public Tablero(int n){
        this.setTitle("Buscaminas");
        // setLayout(new BoxLayout(getContentPane(),BoxLayout.Y_AXIS));
        setLayout(new BorderLayout());
        pm = new PanelMarcador(n);
        pc = new PanelCasillas(n);
        for (int k = 0; k < pc.getComponentCount(); k++){
            ((GrafCasilla)pc.getComponent(k)).getCasilla().addObserver(pm);
        }
        jugador = new JLabel();
        jugador.setFont(new Font("Dialog", Font.BOLD, 15));
        jugador.setForeground(Color.BLACK);

        jugador.setPreferredSize(new Dimension(30,30));

        add(pm, BorderLayout.NORTH);
        add(jugador, BorderLayout.CENTER);
        add(pc, BorderLayout.SOUTH);

        pack();
        setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        //setLocationRelativeTo(null);

    }
    public String getJugador(){
        return jugador.getText();
    }
    public void setJugador(String jug){
        jugador.setText(jug);
    }
    public void cerrar(){
        this.dispose();
    }
}

package pruebaBusca;
import java.util.*;
public class Test_main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner s = new Scanner(System.in);
        String jug = "";
    }
}

```

```

        int n = 0;
        System.out.println("Introduzca usuario (si no quiere enter): ");
        jug = s.nextLine();
        System.out.println("Introduzca nivel (1,2,3): ");
        n = s.nextInt();
        Tablero t = new Tablero(n);
        t.setJugador("Player: "+jug+" (nivel"+n+"");
    }
}

```

4. Implementación del modelo.

Modelo implementado según Patrón MVC, donde el Modelo casilla es el objeto observable, y la vista (Grafica) Tablero2 es el objeto Observador.

Clases: Logueo, Tablero2, Casilla, GrafCasilla y PosiMina.

Funciones de la Vista (Grafica) Logueo:

- Es la ventana inicial donde se piden los datos de nombre de usuario y nivel, al pulsar “Aceptar” los datos pasan al panel “Tablero2” llamando a su constructor pasando numero de filas, columnas, y usuario, para construir el panel de juego.

Funciones de la Vista (Grafica) GrafCasilla:

- Su función es dar propiedades de Botón al modelo Casilla, es decir los atributos del modelo casilla se conservaran (valor, imagen, estado, ...) y además tendrá las propiedades de botón es decir podrá ser presionado (onClick) , cambiar de imagen (setIcon), etc.; propiedades que deben utilizarse en el juego. En esta clase se implementan las respuestas a los eventos de “pulsar boton” (ActionListener(), ActionEvent()), podemos decir que contiene codigo controlador para la clase Casilla.

Funciones del modelo PosiMina:

- Su función es la de guardar la posición de la mina generada (fila, columna), para controlar que si el Random da una posición de mina igual a una ya dada, entonces deberá llamarse nuevamente a la función para que de otra posición, es decir en resumen es necesaria para guardar las posiciones de las minas generadas para volver a generar otra posición si es que el Random repite posición de mina.

Funciones del Modelo Casilla:

- En el momento que se pulsa la primera casilla del panel, la casilla notifica al observador que el juego se debe iniciar, notificándole string “iniciar”.
- Cuando se pulsa una tecla con valor mina (valor -9), la casilla notifica al observador que el juego debe finalizar, notificando string “parar”.
- Cuando la casilla modifica su estado a destapada (estado=1), la casilla notifica dicho hecho, indicando string “destapar”.

Funciones de la Vista (Grafica) Tablero2:

- Construye el panel de juego, con dos subpaneles, uno para las funciones de “marcador”, donde se encontraran los campos utilizados para visualizar las minas del panel, el botón de reinicio y el campo para visualizar el tiempo que se lleva de partida de juego; el otro subpanel será el “panel de los botones casillas”, que interactuaran con el juego a través de clicks de ratón.
 - Se generaran de forma aleatoria las posiciones de las minas.
 - Se generaran los valores de las otras casillas, teniendo en cuenta cuantas minas vecinas tiene cada casilla
 - Se genera un timer para controlar el tiempo, que se activara con las ordenes de los observables.
- Cuando recibe la notificación de “iniciar”, el tablero pone en marcha el cronometro “timer”, mostrandolo en la esquina superior derecha del tablero.
- Cuando recibe la notificación de “parar”, se para el cronometro “timer”.
- Cuando recibe la notificación de “destapar”, resta uno a las casillas activas del panel (casillas no destapadas) y comprueba si el numero de casillas activas es igual o no al numero de minas del panel, si es que es igual el juego finaliza por opción ganadora, entonces se para el timer y se sacara el mensaje de dialogo “!Ganaste!” y la imagen del botón reinicio cambiara a “cara feliz”.

El modelo esta implementado utilizando la herramienta de control de versiones GitHub con Eclipse en el repositorio.: [ssh://git@github.com/olgayp/BuscaminasMOS.git](https://git@github.com/olgayp/BuscaminasMOS.git)

4.1. Generacion proyecto Maven.

En la Figura 1 se muestra la estructura de carpetas para el proyecto.

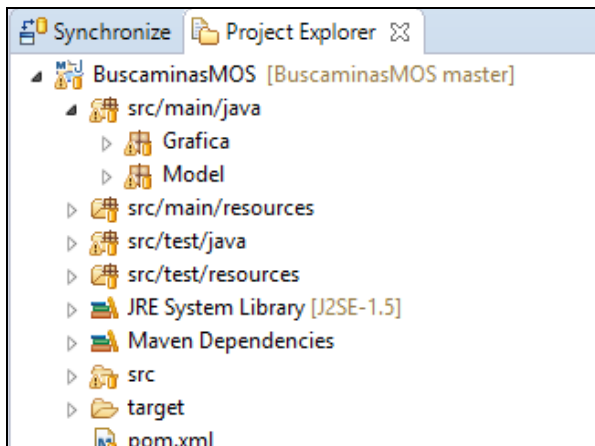


Figura 1

4.2. Implementacion de las clases iniciales (Clase Logueo, Clase Tablero2 y Clase Casilla).

Clase Logeo. Visualización del panel de entrada de datos.

Al pulsar el boton “Aceptar”, se llamara al constructor de la Clase Tablero2 pasandoles las filas, las columnas, el usuario y el nivel; según datos de entrada.

- Si nivel 1, se pasa: filas=7, columnas=10, usuario introducido en panel y nivel = 1.

- Si nivel 2, se pasa: filas=10, columnas=15, usuario introducido en panel y nivel = 2.
- Si nivel 3, se pasa: filas=12, columnas=25, usuario introducido en panel y nivel = 3.

Clase Tablero2. Visualización del panel de juego.

Se generará un panel de juego, compuesto por número de minas, cronómetro, botón de reinicio y matriz de casillas, según nivel que nos viene dado desde Logueo.

En esta etapa, se ha generado una matriz de casillas `[][]`, para posteriormente sacar las casillas panel de juego enlazadas con algún componente gráfico (GrafCasilla- clase gráfica que se implementará posteriormente), aún no hemos generado el panel gráficamente completo.

Clase Casilla. Elemento interno del panel (no Gráfico).

Elemento interno del panel de juego, dicho elemento tiene un valor, una imagen, un estado (tapada/destapada/marcada) y si es o no posición de mina.

4.3. Implementación de los procedimientos para generar valores en las casillas y sacar panel inicial.

Clase GrafCasilla. Propiedades gráficas al elemento Casilla.

Esta clase hereda de JButton propiedades gráficas que enlaza con su atributo Casilla, estos elementos son los que estarán en la matriz de casillas de Tablero2.

Clase PosiMina. Guarda la posición de Mina Generada.

Clase utilizada para guardar la posición (fila, columna) de cada mina generada (valor casilla -9) por si el Random diera un valor igual a otro generado descubrirlo y llamar nuevamente a la función Random para que diese otra posición de casilla Mina.

Método generarMinasTablero(). Genera las posiciones (fila, columna) de las minas en el tablero.

Utilizando la función Random generará posiciones enteras aleatorias entre los valores 0-fila y 0-columna para posicionar las minas en el tablero y lo guardará dicho valor (-9) en la matriz "Casillas[][]".

Método generarValoresTablero(). Genera los valores entre 1-8 de las casillas que no contengan mina.

Estudiando las posiciones de las casillas con mina en la matriz "Casillas[][]", generará valores en dicha matriz teniendo en cuenta que si existe una casilla vecina con mina se debe incrementar en uno su valor inicial (0-cero) y así sucesivamente comprobando todas las casillas vecinas con mina que tenga una determinada casilla, si una casilla no tiene vecinas minas se quedará con valor cero.

Añadir a la clase Tablero2 los atributos “marcador” (JPanel) y “pCasillas”(JPanel).

El JPanel marcador contendrá los elementos propios de un marcador, número de minas, botón de reinicio y cronómetro de partida.

El JPanel pCasillas, contendrá la matriz de todas las casillas según nivel pedido.

Método generarMenu(). Genera la barra de menú del juego.

Se genera la barra de menú con los elementos “Archivo” y “Juego”.

4.4. Implementación métodos para cronometrar tiempo de la partida.

Se implementa un método para cronometrar el tiempo de la partida, utilizando la clase Timer, TimerTask y un contador de tiempo atributo de la clase Tablero2 que se inicializará a 1 y se actualizará según el Timer de tiempo programado (se debe programar que se actualice cada segundo), en cada actualización se sacará el valor del contador al campo tiempo del JPanel “marcador” de la ventana Tablero2.

4.5. Implementación de procedimientos relacionados con pulsar una casilla y su visualización cuando se pulsa.

Método pulsarCasillasVacias().Pulsa todas las casillas vecinas a una vacía hasta encontrarse con una mina.

Busca todas las casillas vecinas a una con valor cero y las pulsa recursivamente mediante un doClick(), al pulsar la casilla se vuelve nuevamente al actionPerformed() y se vuelve a repetir el proceso para la casilla vecina, así sucesivamente hasta encontrar un valor mina como casilla vecina.

Método generarEtiquetaValor().Genera la etiqueta que va a sustituir al botón.

Se genera la etiqueta que va a sustituir a la casilla-botón, con el valor de dicha casilla.

Tratamiento Imágenes.

Se realizan tratamientos de las imágenes para ajustarlas al tamaño del botón-casilla.

4.6. Método update() del observador Tablero2.

Se implementa el método update () de Tablero2 (sobrescribir método) para dar respuesta a los eventos del objeto observable.

- Si llega la notificación de “iniciar”, se pone en marcha el Timer.
- Si llega la notificación “parar” se para el Timer y se saca el panel de perdedor.
- Si llega la notificación “destapar”, se resta uno a las casillas activas y se comprueba si esas casillas activas son iguales al número de minas, si son iguales se saca el mensaje de ganador y si son diferentes continúa el juego.

5. Pruebas Unitarias JUnit.

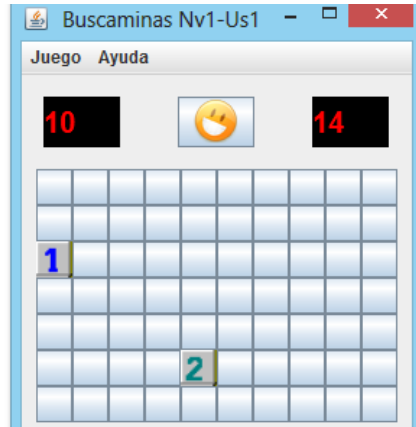
Pruebas JUnit de lo elementos no graficos, Casilla y PosiMina.

Prueba	Resultado esperado	Resultado Obtenido
new Casilla(true)	assertNotNull	assertNotNull
new Casilla(-9,"bomba.jpg",0,0))	assertNotNull	assertNotNull
Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(4, c1.getValor())	assertEquals(4, c1.getValor())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(0, c1.getEstado())	assertEquals(0, c1.getEstado())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertFalse(c1.getMina())	assertFalse(c1.getMina())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(0, c1.getI())	assertEquals(0, c1.getI())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(1, c1.getJ())	assertEquals(1, c1.getJ())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals("4.PNG", c1.getImagen())	assertEquals("4.PNG", c1.getImagen())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setImg("9.PNG");	assertEquals("9.PNG", c1.getImagen())	assertEquals("9.PNG", c1.getImagen())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setValor(9);	assertEquals(9, c1.getValor())	assertEquals(9, c1.getValor())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setEstado(1);	assertEquals(1, c1.getEstado())	assertEquals(1, c1.getEstado())
Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setMina(true);	assertTrue(c1.getMina());	assertTrue(c1.getMina());
Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setI(2);	assertEquals(2, c1.getI());	assertEquals(2, c1.getI());
Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(3, c1.getJ())	assertEquals(3, c1.getJ())
new PosiMina(1,2)	assertNotNull	assertNotNull

[illegible]

Con todo esto, podemos realizar un simulacro y comprobar los resultados.

- pulsar casilla (2,0) → tiene que ponerse en marcha el contador de tiempo, y aparecer el valor 1.
- Pulsar casilla (5,4) → el contador de tiempo continuará y la casilla debera de tener un valor 2.

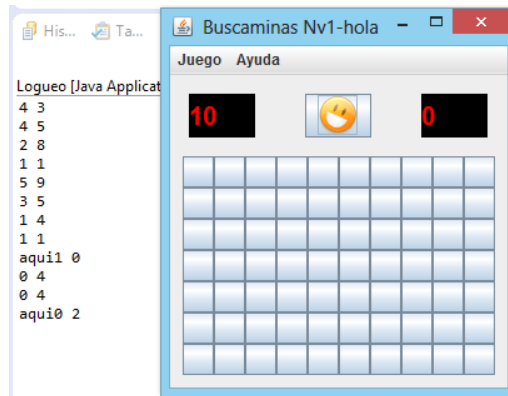


Observamos que el resultado es el esperado.

- Pulsaremos la (0,0) y comprobamos que se ha hecho un contorno con todas las casillas con valor cero (sin minas vecinas) hasta llegar a casillas minas vecinas; posteriormente pulsamos la (6,2) y el juego terminara “perdiendo” ya que he encontrado una mina cuando aún me quedan más de diez casillas por destapar; el juego terminara sacando la imagen “triste” en el boton de reinicio y sacando el mensaje “!PERDISTE...!” en un cuadro de Dialogo.



- Ahora pasaremos a probar la opcion ganadora.



A la izq. de la pantalla podemos ver la lista de posiciones de mina, vemos unos casos especiales marcado con “aquí”, esto es debido a aque la funcion Randon ha dado una mima posición varias veces, vemos que ha repetido la 1-1 y entonces ha dado una nueva posicion la 1-0, lo mismo ha ocurrido con la 0-4 y entonces ha dado la 0-2.



Podemos ver que en 169 segundos hemos conseguido discriminar todas las casillas con minas, viendo que 10 son las casillas no descubiertas y que esas casillas coinciden con las posiciones de las minas sacadas a consola. Entonces la imagen del boton reinicio se torna “Feliz” y se saca a pantalla el mensaje de dialogo “ ¡GANASTE...!”

7. Corrección de errores en Implementación.

Corrección de algunos errores, descubierto durante las pruebas.

8. Finalizar Documentación.

Finalizar Documentación de la practica.

CODIGO.

Codigo de las clases intervinientes en las historia HU2 y HU3 (Logueo, Tablero2, Casilla, GrafCasilla y PosiMina)

```
package Model;
import java.awt.event.*;

import java.awt.*;

import javax.swing.*;

import java.util.*;

//A cada hueco de la matriz que va a crear el tablero se le asigna una casilla

public class Casilla extends Observable{
    //Almacena el valor de la casilla (1-8 --> casilla normal)
    //          (-9 --> casilla con mina)
    private int valor;
    //Almacena si esta tapada destapada o marcada
    //0=tapada 1=destapada 2=marcada
    private int estado;
    //Almacena si tiene una bomba
    //true = tiene mina false= no tiene mina
    private boolean mina;
    //posicion i, j de la mina
    private int i;
    private int j;
    //Sirven para administrar la imagen de la casilla
    //INVESTIGAR MAS ESTO
    private String imagen;
    private Icon icanterior;

    public Casilla (boolean caboom){
        estado=0;
        mina=caboom;
        i = 0;
        j = 0;
    }

    public Casilla (int v, String img, int i, int j){
        valor = v;
        if (valor == -9)
            mina = true;
        imagen=img;
        icanterior = new ImageIcon(img);
        estado=0;
        mina=false;
        this.i = i;
        this.j = j;
    }

    public int getValor(){
        return valor;
    }

    public int getEstado(){
        return estado;
    }

    public boolean getMina(){
        return mina;
    }

    public int getI(){
        return i;
    }

    public int getJ(){
        return j;
    }

    public String getImagen(){
        return imagen;
    }
}
```



```

    }

    public Icon getIcante(){
        return icanterior;
    }
    public void setImg (String plmg){
        icanterior = new ImagenIcon(imagen);
        imagen = plmg;
    }
    public void setValor(int v){
        valor = v;
    }
    public void setEstado(int e){
        int old=0;
        old = estado;
        estado = e;
        if (estado == 2){ //si marcada informar
            setChanged();
            notifyObservers("marcar");
        }
        else{
            if (old == 2){
                setChanged();
                notifyObservers("desmarcar");
            }
            if (estado == 1 && valor != -9){
                setChanged();
                notifyObservers("destapar");
            }
        }
    }

    }

    public void setMina(boolean b){
        mina = b;
    }
    public void setI(int i){
        this.i = i;
    }
    public void setJ(int j){
        this.j = j;
    }
    public void setIante(ImagenIcon img){
        icanterior = img;
    }
    }

    public void parar(int n){
        setChanged();
        if (n == -9)
            this.notifyObservers("pararPierdo");
        else
            this.notifyObservers("pararGano");
    }
    public void iniciar(){
        System.out.println("INICIO RECIBIDO...");
        setChanged();
        this.notifyObservers("iniciar");
    }
}

package Grafica;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.Frame;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import javax.swing.JButton;

```

```

import javax.swing.DefaultComboBoxModel;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.rmi.server.Operation;

public class Logueo extends JFrame {

    private JPanel contentPane;
    private JLabel lblPorFavorRellena;
    private JLabel lblDeseasJugar;
    private JLabel lblTuNombre;
    private JLabel lblDificultad;
    private JComboBox comboBox;
    private JTextField textField;
    private JButton btnAceptar;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Logueo frame = new Logueo();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public Logueo() {
        initialize();
        setLocationRelativeTo(null); //centra el JFrame
    }
    private void initialize() {
        setTitle("Bienvenido al buscaminas");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        contentPane.add(getLblPorFavorRellena());
        contentPane.add(getLblDeseasJugar());
        contentPane.add(getLblTuNombre());
        contentPane.add(getLblDificultad());
        contentPane.add(getComboBox());
        contentPane.add(getTextField());
        contentPane.add(getBtnAceptar());
    }
    private JLabel getLblPorFavorRellena() {
        if (lblPorFavorRellena == null) {
            lblPorFavorRellena = new JLabel("Por favor, rellena los huecos con tu nombre y el nivel de dificultad que");
            lblPorFavorRellena.setBounds(10, 11, 395, 44);
        }
        return lblPorFavorRellena;
    }
    private JLabel getLblDeseasJugar() {
        if (lblDeseasJugar == null) {
            lblDeseasJugar = new JLabel("deseas jugar.Despues pulsa 'Aceptar'.");
            lblDeseasJugar.setBounds(10, 41, 352, 23);
        }
        return lblDeseasJugar;
    }
    private JLabel getLblTuNombre() {
        if (lblTuNombre == null) {
            lblTuNombre = new JLabel("Tu nombre:");
            lblTuNombre.setBounds(88, 75, 89, 23);
        }
    }

```

```

        }
        return lblTuNombre;
    }
    private JLabel getLblDificultad() {
        if (lblDificultad == null) {
            lblDificultad = new JLabel("Dificultad:");
            lblDificultad.setBounds(88, 109, 63, 14);
        }
        return lblDificultad;
    }
    private JComboBox getComboBox() {
        if (comboBox == null) {
            comboBox = new JComboBox();
            comboBox.setModel(new DefaultComboBoxModel(new String[] { "F\u00E1cil", "Medio", "Dif\u00EDcil" }));
            comboBox.setToolTipText("");
            comboBox.setBounds(187, 106, 63, 20);
        }
        return comboBox;
    }
    private JTextField getTextField() {
        if (textField == null) {
            textField = new JTextField();
            textField.setBounds(187, 76, 86, 20);
            textField.setColumns(10);
        }
        return textField;
    }
    private JButton getBtnAceptar() {
        if (btnAceptar == null) {
            btnAceptar = new JButton("Aceptar");
            btnAceptar.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    if(textField.getText().isEmpty()==false){

                        if(comboBox.getSelectedIndex()==0){
                            Tablero2 frame= new Tablero2(7,10,textField.getText(),1);
                            frame.setVisible(true);
                        }
                        if(comboBox.getSelectedIndex()==1){
                            Tablero2 frame = new Tablero2(10,15,textField.getText(),2);
                            frame.setVisible(true);
                        }
                        if(comboBox.getSelectedIndex()==2){
                            Tablero2 frame= new Tablero2(12,25,textField.getText(),3);
                            frame.setVisible(true);
                        }
                        setVisible(false);
                    }
                }
            });
            btnAceptar.setBounds(149, 187, 89, 23);
        }
        return btnAceptar;
    }
}

package Grafica;
import java.awt.BorderLayout;
import Model.Casilla;
import java.util.Observable;
import java.util.Observer;
import java.util.Timer;
import java.util.TimerTask;
import java.util.Random;
import java.util.*;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.GraphicsConfiguration;

import javax.swing.JFrame;
import javax.swing.JPanel;

```

```

import javax.swing.border.*;
import javax.swing.BoxLayout;

import java.awt.GridLayout;
import java.awt.*;
import javax.swing.border.SoftBevelBorder;

import Model.Casilla;
import Model.PosiMina;

import javax.swing.border.BevelBorder;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Tablero2 extends JFrame implements Observer {
    //Panel general
    private JPanel contentPane;
    //
    private int casActivas; // numero de casillas activas del panel
    //
    //Panel para casillas
    //Campos utilizados para casillas
    //
    private JPanel pCasillas;
    private Casilla [][] tablero ;
    //
    // Barra de menu
    // Campos para elementos de Menu
    //
    private JMenuBar barraMenu;
    private JMenu juego;
    private JMenu ayuda;
    //
    //Panel para marcador
    //Campos utilizados para el marcador, minas, tiempo y boton de reinicio
    //
    private JPanel marcador;
    private JLabel minas;
    private JLabel tiempo;
    private JButton reinicio;
    //
    // Campos para contador de tiempo de partida
    //
    private int cont;
    private Timer timer = null;

    //
    //User es el nombre del jugador que esta jugando, se debe pasar a la pantalla de continuar cuando
    //acabe el juego
    private String user;
    private int nivel;

    /**
     * Launch the application.
     */
    //public static void main(String[] args,final int fila, final int columna,final String user,final int nivel) {
    public static void main (String [] args){
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    //Tablero2 frame = new Tablero2(fila,columna,user,nivel );
                    Tablero2 frame = new Tablero2(7,10,"hola",1);
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

```

```

        }
    });
}

/**
 * Create the frame.
 * @param user
 */
public Tablero2(int fila,int columna, String user,int nivel) {
    this.user=user;
    this.nivel=nivel;
    casActivas = fila*columna;
    //Generar Barra de Menu
    generarMenu();
    //inicializar
    initialize(fila, columna);
    pack();
}

private void initialize(int fila,int columna) {

    setTitle("Buscaminas Nv"+nivel+"-"+user);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);

    marcador = new JPanel();
    marcador.setBorder(new EmptyBorder(10,10,10,10));

    minas = new JLabel(String.valueOf(columna*nivel));
    minas.setPreferredSize(new Dimension(35,35));
    minas.setForeground(Color.RED);
    minas.setFont(new Font("Dialog", Font.BOLD, 20));
    minas.setOpaque(true);
    minas.setBackground(Color.BLACK);

    tiempo = new JLabel("0");
    tiempo.setPreferredSize(new Dimension(35,35));
    tiempo.setForeground(Color.RED);
    tiempo.setFont(new Font("Dialog", Font.BOLD, 20));
    tiempo.setOpaque(true);
    tiempo.setBackground(Color.BLACK);

    reinicio = new JButton();
    String s = getClass().getClassLoader().getResource("nueva.png").toString();
    reinicio.setIcon(new ImageIcon(s.substring(6)));
    reinicio.setPreferredSize(new Dimension(35,35));

    marcador.setLayout(new GridLayout(1,3,30+(10*nivel),10));
    marcador.add(minas, BorderLayout.WEST);
    marcador.add(reinicio, BorderLayout.CENTER);
    marcador.add(tiempo, BorderLayout.EAST);
    add(marcador);

    pCasillas = new JPanel();
    pCasillas.setBorder(new EmptyBorder(5,5,5,5));
    pCasillas.setLayout(new GridLayout(fila, columna, 0, 0));
    add(pCasillas);

    //
    // Inicializar tablero con casillas vacias
    //
    tablero = new Casilla [fila][columna];
    for (int i = 0; i < fila; i++){
        for (int j = 0; j < columna; j++){
            tablero[i][j] = new Casilla(0, "", i, j);
        }
    }
    // Posicionar minas en tablero
    generarMinasTablero(fila,columna);
}

```

```

//generar valores de las casillas del tablero
generarValoresTablero(fila,columna);

// añadir casillas-botones al panel
for (int i = 0; i < fila; i++) {
    for (int j = 0; j < columna; j++) {
        //Añade el boton
        //final JButton btnNewButton = new JButton(i+ ", "+ j);

        //add(btnNewButton);

        //Generar Etiqueta Vacía para el boton-Casilla
        JLabel l1 = generarEtiquetaValor(tablero[i][j]);
        //crear boton-casilla para añadir al panel con su etiqueta
        GrafCasilla gc = new GrafCasilla(tablero[i][j], l1);
        // poner a la casilla en observador
        gc.getCasilla().addObserver(this);
        pCasillas.add(gc);
        //gc.setText(String.valueOf(gc.getCasilla().getValor())); //QUITAAR COMENTARIO Para probar ganar
    }
}
//contentPane.setLayout(new GridLayout(fila, columna, 0, 0));
add(pCasillas);
contentPane.setLayout(new BoxLayout(getContentPane(),BoxLayout.Y_AXIS));
setLocationRelativeTo(null); //centra el JFrame
}

public void generarMinasTablero(int f, int c){
    int numMinas = 0;
    int vf = 0;
    int vc = 0;

    numMinas = c*nivel; //numero de minas es el producto de columnas del panel * nivel de juego
    Random r = new Random(); //funcion random() para generar posiciones de minas

    PosiMina [] pm = new PosiMina[numMinas];
    for (int s = 0; s < numMinas; s++){
        pm[s]= new PosiMina(0,0);
    }
    int ipm = 0;
    for (int k = 0; k < numMinas; k++){
        vf = r.nextInt(f);
        vc = r.nextInt(c);
        System.out.println (vf+" "+vc);
        int l = 0;
        while (l < numMinas){ //controlar que random no ha dado una posicion de mina ya existente
            PosiMina psm = new PosiMina(vf,vc); // se genera una poscion de mina

            if ((pm [l].getFbomb() == psm.getFbomb()) && //ha dado la misma posicion
                (pm [l].getCbomb() == psm.getCbomb())){ //buscar otra posicion
                vf = r.nextInt(f);
                vc = r.nextInt(c);
                l = 0;
                System.out.println ("aqui"+vf+" "+vc);
            }
            else
                l = l + 1;
        }
        String s = getClass().getClassLoader().getResource("9.png").toString();
        tablero [vf][vc] = new Casilla(-9,s.substring(6),vf,vc);
        pm [ipm] = new PosiMina(vf,vc); //Guardar posicion de mina generada
        ipm = ipm + 1;
    }
}

public void generarValoresTablero(int f, int c){
    for (int i = 0; i < f; i++){
        for (int j = 0; j < c; j++){
            if (tablero[i][j].getValor() == -9){ //si hay una mina sumar uno a sus casillas vecinas
                if (j+1 < c && tablero[i][j+1].getValor() != -9)
                    //siguiente

```

```

        tablero[i][j+1].setValor(tablero[i][j+1].getValor()+1);
        if (j-1 >= 0 && tablero[i][j-1].getValor() != -9)
            //anterior
            tablero[i][j-1].setValor(tablero[i][j-1].getValor()+1);
        if (i-1 >= 0 && tablero[i-1][j].getValor() != -9)
            //arriba
            tablero[i-1][j].setValor(tablero[i-1][j].getValor()+1);
        if (i+1 < f && tablero[i+1][j].getValor() != -9)
            //abajo
            tablero[i+1][j].setValor(tablero[i+1][j].getValor()+1);
        if (i-1 >= 0 && j+1 < c && tablero[i-1][j+1].getValor() != -9)
            //superior derecha
            tablero[i-1][j+1].setValor(tablero[i-1][j+1].getValor()+1);
        if (i-1 >= 0 && j-1 >= 0 && tablero[i-1][j-1].getValor() != -9)
            //superior izquierda
            tablero[i-1][j-1].setValor(tablero[i-1][j-1].getValor()+1);
        if (i+1 < f && j+1 < c && tablero[i+1][j+1].getValor() != -9)
            //inferior derecha
            tablero[i+1][j+1].setValor(tablero[i+1][j+1].getValor()+1);
        if (i+1 < f && j-1 >= 0 && tablero[i+1][j-1].getValor() != -9)
            //inferior izquierda
            tablero[i+1][j-1].setValor(tablero[i+1][j-1].getValor()+1);
    }

}

}

// Gestion del tiempo de la partida
// creacion de un timer que actualiza un contador cada segundo
// poniendo dicho valor en el panel de juego
public void gestionDeTiempo(){
    cont = 0;
    timer = new Timer();
    TimerTask timerTask = new TimerTask(){
        public void run (){
            sumarContador();
            System.out.println("TIEMPO...."+getContador());
            tiempo.setText(String.valueOf(getContador()));
        }
    };
    timer.scheduleAtFixedRate(timerTask, 0, 1000);
}

public JLabel generarEtiquetaValor(Casilla c){
    String img = "0.png";
    if (c.getValor() == 0)
        img = "0.png";
    if (c.getValor() == 1)
        img = "1.png";
    if (c.getValor() == 2)
        img = "2.png";
    if (c.getValor() == 3)
        img = "3.png";
    if (c.getValor() == 4)
        img = "4.png";
    if (c.getValor() == 5)
        img = "5.png";
    if (c.getValor() == 6)
        img = "6.png";
    if (c.getValor() == 7)
        img = "7.png";
    if (c.getValor() == 8)
        img = "8.png";
    if (c.getValor() == -9)
        img = "9.png";
    String sl = getClass().getClassLoader().getResource(img).toString();
    JLabel l = new JLabel();
    l.setIcon(new ImageIcon(sl.substring(6)));
    return l;
}

//Suma uno al contador de tiempo cada segundo
public void sumarContador(){

```

```

        cont = cont + 1;
    }

    //Devuelve el contador de tiempo
    public int getContador(){
        return cont;
    }

    //restar numero de minas
    public void restarMinas(){
        int m = Integer.valueOf(minas.getText());
        m = m - 1;
        minas.setText(String.valueOf(m));
    }

    //sumar numero de minas
    public void sumarMinas(){
        int m = Integer.valueOf(minas.getText());
        m = m + 1;
        minas.setText(String.valueOf(m));
    }

    //Genera la barra de menu
    public void generarMenu(){
        barraMenu = new JMenuBar();
        juego=new JMenu("Juego");
        ayuda=new JMenu("Ayuda");
        barraMenu.add(juego);
        barraMenu.add(ayuda);
        //FALTARIA AÑADIR SUBMENUS....
        //??????????? EVENTOS ??????
        setJMenuBar(barraMenu);
    }

    public void update(Observable o, Object arg) {
        // TODO Auto-generated method stub

        if (arg.equals("iniciar")){
            if (getContador()==0){
                gestionDeTiempo();
                System.out.println("Juego iniciado...");
            }
        }
        if (arg.equals("pararPierdo")){
            timer.cancel();
            String sreinic = getClass().getClassLoader().getResource("perdio.png").toString();
            reinicio.setIcon(new ImageIcon(sreinic.substring(6)));
            JOptionPane.showMessageDialog(this, "¡PERDISTE...!");
            //evaluarJuego();
        }
        if (arg.equals("marcar")){
            restarMinas();
            casActivas = casActivas - 1;
            if (casActivas == Integer.parseInt(minas.getText())){
                timer.cancel();
                String sreinic = getClass().getClassLoader().getResource("gano.png").toString();
                reinicio.setIcon(new ImageIcon(sreinic.substring(6)));
                JOptionPane.showMessageDialog(this, "¡GANASTE...!");
            }
        }
        if (arg.equals("desmarcar")){
            sumarMinas();
            casActivas = casActivas + 1;
            if (casActivas == Integer.parseInt(minas.getText())){
                timer.cancel();
                String sreinic = getClass().getClassLoader().getResource("gano.png").toString();
                reinicio.setIcon(new ImageIcon(sreinic.substring(6)));
                JOptionPane.showMessageDialog(this, "¡GANASTE...!");
            }
        }
        if (arg.equals("destapar")){
            casActivas = casActivas - 1;
            if (casActivas == Integer.parseInt(minas.getText())){
                timer.cancel();
                String sreinic = getClass().getClassLoader().getResource("gano.png").toString();
            }
        }
    }

```



```

                reinicio.setIcon(new ImageIcon(sreinic.substring(6)));
                JOptionPane.showMessageDialog(this, "¡GANASTE...!");
            }
        }
    }
}

package Grafica;
import Model.Casilla;
import java.awt.event.*;
import java.awt.*;
import java.util.*;

import javax.swing.*;

public class GrafCasilla extends JButton{
    private Casilla cas;
    public static final int TAMX=25; //Anchura de la casilla
    public static final int TAMY=25; //Altura de la casilla
    private JLabel etqVal; //Etiqueta con su valor

    //
    // Constructor
    //
    public GrafCasilla (Casilla c, JLabel etq){
        cas = c;
        etqVal = etq;
        setVisible(true);
        setEnabled(true);
        setPreferredSize(new Dimension(TAMX,TAMY));
        //
        //Evento action listener al pulsar una casilla
        //
        addActionListener(new ActionListener(){ // al pulsar boton-casilla se inicia evento

            public void actionPerformed(ActionEvent e) {
                String sv = "";
                sv = Integer.toString(cas.getValor());
                setFocusable(false);
                Container cter = ((GrafCasilla)e.getSource()).getParent();
                //casilla pulsada--> el juego comienza, se da la orden iniciar al observador
                cas.iniciar();
                if (cas.getEstado() == 0){ //si no marcada se acepta el evento
                    if (cas.getValor() == -9){ // Casilla mina, el juego ha terminado
                        setOpaque(false);
                        setIcon(new ImageIcon(cas.getImagen())); //imagen de mina
                        setBackground(Color.RED);
                        setEnabled(false);
                        String smr = getClass().getClassLoader().getResource("9R.png").toString();
                        ((GrafCasilla)e.getSource()).setEtiqueta(smr.substring(6));
                        int pm1 = cter.getComponentZOrder((GrafCasilla)e.getSource());
                        cter.add(((GrafCasilla)e.getSource()).getEtiqueta(), pm1);
                        cter.remove(pm1+1);

                        //
                        // El juego ha terminado (se ha sacado una mina) --> descubrir resto de casillas
                        //
                        Component listCas [] = cter.getComponents();
                        for (int i = 0; i < listCas.length; i++){
                            if (!(listCas[i] instanceof JLabel)){
                                if (((GrafCasilla)listCas[i]).getCasilla().getValor() == -9){
                                    ((GrafCasilla)listCas[i]).getCasilla().setEstado(1);//marcar a destapada
                                }
                                else{
                                    //cambiar a etiqueta de tapada
                                    String se =
                                        getClass().getClassLoader().getResource("0.png").toString();
                                    ((GrafCasilla)listCas[i]).setEtiqueta(se.substring(6));
                                }
                                //***** Sustituir botones por etiquetas al descubrir casilla
                                int p1 = cter.getComponentZOrder((GrafCasilla)listCas[i]);
                                cter.add(((GrafCasilla)listCas[i]).getEtiqueta(), p1);
                            }
                        }
                    }
                }
            }
        });
    }
}

```

```

        cter.remove(p1+1);
        //***** Sustituir botones por etiquetas al descubrir casilla
    }
}
//dar orden de parar el juego al modelo Casilla
cas.parar(-9);
}
else{ // casilla no es mina, por tanto sacar su valor
    setOpaque(false);

    if (sv.equals("0")){ //si es cero es una casilla vacia - SPACES
        sv = "";
        setText(sv);
        setEnabled(false);

        //***** Sustituir botones por etiquetas al descubrir casilla
        //filas del JPanel que contiene la casilla
        int fil = ((GridLayout)cter.getLayout()).getRows();
        //columnas del JPanel que contiene la casilla
        int col = ((GridLayout)cter.getLayout()).getColumns();
        int p1 = cter.getComponentZOrder((GrafCasilla)e.getSource());
        int x = cas.getI();
        int y = cas.getJ();
        cter.add(etqVal, p1);
        cter.remove(p1+1);
        //***** Sustituir botones por etiquetas al descubrir casilla

        //pulsar casillas vacias vecinas con doClick() recursivamente
        pulsarCasillasVacias(fil,col,p1,cter,x,y);
    }
    else{ // si no es cero descubrir su valor
        setText(sv);
        setEnabled(false);
        // setVisible(false);

        //***** Sustituir botones por etiquetas al descubrir casilla
        int p1 = cter.getComponentZOrder((GrafCasilla)e.getSource());
        cter.add(etqVal, p1);
        cter.remove(p1+1);
        //***** Sustituir botones por etiquetas al descubrir casilla
    }
    cas.setEstado(1); //marcar casilla como destapada ESTADO=1
}
}
});

//
//Evento Mouse listener para controlar el raton al pasar por un boton-casilla
//
addMouseListener(new MouseListener(){
    public void mousePressed(MouseEvent me) { //si boton derecho marcar con bandera
        if (isEnabled()){
            if ((me.getModifiers() &
                InputEvent.BUTTON3_MASK) == InputEvent.BUTTON3_MASK) {
                if (cas.getEstado() != 2){ // si boton derecho y no marcada
                    cas.setIante((Imagelcon)getIcon());
                    String s = getClass().getClassLoader().
                        getResource("b1.png").toString();
                    setIcon(new Imagelcon(s.substring(6)));
                    cas.setEstado(2); // ESTADO = 2 (Marcada)
                }
            }
            else{ //si marcada y boton derecho --> desmarcar y dejar icono inicial
                setIcon((Imagelcon)cas.getIcante());
                cas.setEstado(0); //casilla oculta
            }
        }
    }

    }

}

}

public void mouseReleased(MouseEvent me) {
}

public void mouseExited(MouseEvent me) {

```

```

    }
    public void mouseEntered(MouseEvent me) {
    }
    public void mouseClicked(MouseEvent me) {
    }
};

}
//
// Al encontrar una casilla vacia se pulsan todas sus vecinas recursivamente con
// doClick() hasta encontrar una casilla vecina que contenga una mina.
// @param fil - filas del contenedor de la casilla pulsada
// col - columnas del contenedor de la casilla pulsada
// p - posicion absoluta de la casilla pulsada
// ctainer - contenedor (JPanel) de la casilla pulsada
// i - posicion relativa x de la casilla pulsada dentro de la matriz
// j - posicion relativa j de la casilla pulsada dentro de la matriz
//
public void pulsarCasillasVacias(int fil, int col, int p, Container ctainer, int i, int j){

    // casilla siguiente
    if ( p+1<ctainer.getComponentCount() &&
        !(ctainer.getComponent(p+1) instanceof JLabel))

        if (((GrafCasilla)(ctainer.getComponent(p+1))).getCasilla().getEstado() == 0 &&
            ((GrafCasilla)(ctainer.getComponent(p+1))).getCasilla().getValor() != -9 &&
            ((GrafCasilla)(ctainer.getComponent(p+1))).isEnabled() &&
            ((GrafCasilla)(ctainer.getComponent(p+1))).getCasilla().getI() < col &&
            (i == ((GrafCasilla)(ctainer.getComponent(p+1))).getCasilla().getI()) &&
            (j+1 == ((GrafCasilla)(ctainer.getComponent(p+1))).getCasilla().getJ()
            )
            )
            ((GrafCasilla)(ctainer.getComponent(p+1))).doClick();

    //casilla anterior
    if ( p-1 >= 0 && !(ctainer.getComponent(p-1) instanceof JLabel))
        if (((GrafCasilla)(ctainer.getComponent(p-1))).getCasilla().getEstado() == 0 &&
            ((GrafCasilla)(ctainer.getComponent(p-1))).getCasilla().getValor() != -9 &&
            ((GrafCasilla)(ctainer.getComponent(p-1))).isEnabled() &&
            ((GrafCasilla)(ctainer.getComponent(p-1))).getCasilla().getI() >= 0 &&
            (i == ((GrafCasilla)(ctainer.getComponent(p-1))).getCasilla().getI()) &&
            (j-1 == ((GrafCasilla)(ctainer.getComponent(p-1))).getCasilla().getJ()
            )
            )
            ((GrafCasilla)(ctainer.getComponent(p-1))).doClick();

    //casilla arriba
    if ( p-col >= 0 && !(ctainer.getComponent(p-col) instanceof JLabel))
        if (((GrafCasilla)(ctainer.getComponent(p-col))).getCasilla().getEstado() == 0 &&
            ((GrafCasilla)(ctainer.getComponent(p-col))).getCasilla().getValor() != -9 &&
            ((GrafCasilla)(ctainer.getComponent(p-col))).isEnabled() &&
            ((GrafCasilla)(ctainer.getComponent(p-col))).getCasilla().getI() >= 0 &&
            (i-1 == ((GrafCasilla)(ctainer.getComponent(p-col))).getCasilla().getI()) &&
            (j == ((GrafCasilla)(ctainer.getComponent(p-col))).getCasilla().getJ()
            )
            )
            ((GrafCasilla)(ctainer.getComponent(p-col))).doClick();

    //casilla abajo
    if (p+col<ctainer.getComponentCount() && !(ctainer.getComponent(p+col) instanceof JLabel))
        if ( ((GrafCasilla)(ctainer.getComponent(p+col))).getCasilla().getEstado() == 0 &&
            ((GrafCasilla)(ctainer.getComponent(p+col))).getCasilla().getValor() != -9 &&
            ((GrafCasilla)(ctainer.getComponent(p+col))).isEnabled() &&
            ((GrafCasilla)(ctainer.getComponent(p+col))).getCasilla().getI() < fil &&
            (i+1 == ((GrafCasilla)(ctainer.getComponent(p+col))).getCasilla().getI()) &&
            (j == ((GrafCasilla)(ctainer.getComponent(p+col))).getCasilla().getJ()
            )
            )
            ((GrafCasilla)(ctainer.getComponent(p+col))).doClick();

    //casilla superior izquierda
    if (p-col-1 >= 0 && !(ctainer.getComponent(p-col-1) instanceof JLabel))
        if (((GrafCasilla)(ctainer.getComponent(p-col-1))).getCasilla().getEstado() == 0 &&

```

```

        ((GrafCasilla)(ctainer.getComponent(p-col-1))).getCasilla().getValor() != -9 &&
        ((GrafCasilla)(ctainer.getComponent(p-col-1))).isEnabled() &&
        ((GrafCasilla)(ctainer.getComponent(p-col-1))).getCasilla().getI() >= 0 &&
        ((GrafCasilla)(ctainer.getComponent(p-col-1))).getCasilla().getJ() >= 0 &&
        (i-1 == ((GrafCasilla)(ctainer.getComponent(p-col-1))).getCasilla().getI()) &&
        (j-1 == ((GrafCasilla)(ctainer.getComponent(p-col-1))).getCasilla().getJ()
    )
    )

        ((GrafCasilla)(ctainer.getComponent(p-col-1))).doClick();

        //casilla superior derecha
        if (p-col+1 >= 0 && !(ctainer.getComponent(p-col+1) instanceof JLabel))
            if ( ((GrafCasilla)(ctainer.getComponent(p-col+1))).getCasilla().getEstado() == 0 &&
                ((GrafCasilla)(ctainer.getComponent(p-col+1))).getCasilla().getValor() != -9 &&
                ((GrafCasilla)(ctainer.getComponent(p-col+1))).isEnabled() &&
                ((GrafCasilla)(ctainer.getComponent(p-col+1))).getCasilla().getI() >= 0 &&
                ((GrafCasilla)(ctainer.getComponent(p-col+1))).getCasilla().getJ() < col &&
                (i-1 == ((GrafCasilla)(ctainer.getComponent(p-col+1))).getCasilla().getI()) &&
                (j+1 == ((GrafCasilla)(ctainer.getComponent(p-col+1))).getCasilla().getJ()
            )
        )
        )

        ((GrafCasilla)(ctainer.getComponent(p-col+1))).doClick();

        //casilla inferior izquierda
        if (p-col-1 < ctainer.getComponentCount() && !(ctainer.getComponent(p+col-1) instanceof JLabel))
            if (((GrafCasilla)(ctainer.getComponent(p+col-1))).getCasilla().getEstado() == 0 &&
                ((GrafCasilla)(ctainer.getComponent(p+col-1))).getCasilla().getValor() != -9 &&
                ((GrafCasilla)(ctainer.getComponent(p+col-1))).isEnabled() &&
                ((GrafCasilla)(ctainer.getComponent(p+col-1))).getCasilla().getI() < fil &&
                ((GrafCasilla)(ctainer.getComponent(p+col-1))).getCasilla().getJ() >= 0 &&
                (i+1 == ((GrafCasilla)(ctainer.getComponent(p+col-1))).getCasilla().getI()) &&
                (j-1 == ((GrafCasilla)(ctainer.getComponent(p+col-1))).getCasilla().getJ()
            )
        )
        )

        ((GrafCasilla)(ctainer.getComponent(p+col-1))).doClick();

        //casilla inferior derecha
        if (p+col+1 < ctainer.getComponentCount() && !(ctainer.getComponent(p+col+1) instanceof JLabel))
            if (((GrafCasilla)(ctainer.getComponent(p+col+1))).getCasilla().getEstado() == 0 &&
                ((GrafCasilla)(ctainer.getComponent(p+col+1))).getCasilla().getValor() != -9 &&
                ((GrafCasilla)(ctainer.getComponent(p+col+1))).isEnabled() &&
                ((GrafCasilla)(ctainer.getComponent(p+col+1))).getCasilla().getI() < fil &&
                ((GrafCasilla)(ctainer.getComponent(p+col+1))).getCasilla().getJ() < col &&
                (i+1 == ((GrafCasilla)(ctainer.getComponent(p+col+1))).getCasilla().getI()) &&
                (j+1 == ((GrafCasilla)(ctainer.getComponent(p+col+1))).getCasilla().getJ()
            )
        )
        )

        ((GrafCasilla)(ctainer.getComponent(p+col+1))).doClick();

    }

    //
    // Devolver el valor de la casilla
    // @return valor de la casilla
    //
    public Casilla getCasilla(){
        return cas;
    }

    //
    // Devolver el valor de la etiqueta
    // @return etiqueta de la casilla
    //
    public JLabel getEtiqueta(){
        return etqVal;
    }

    //
    // Cambiar el valor de la etiqueta
    // @param nueva etiqueta para la casilla
    //
    public void setEtiqueta(String newEtq){
        JLabel nl = new JLabel();
        nl.setIcon(new ImageIcon(newEtq));
        etqVal = nl;
    }

```

```

    }

}
package Model;

public class PosiMina { //guardar mina por si el ramdon da la misma posicion y hay que buscar otra poscion

    private int fBomb;
    private int cBomb;
    public PosiMina(int f, int c){
        fBomb = f;
        cBomb = c;
    }
    public int getFbomb(){
        return fBomb;
    }
    public int getCbomb(){
        return cBomb;
    }
}
}

```

DOCUMENTACIÓN DE TAREAS.

SPRINT1					
Tarea	SubTarea	Responsable	Plan	Real	Comentarios
Diseño	Diseño UML	Miguel Olga Susana	5	5	
	Diseño Diagrama de Secuencia	Miguel Olga Susana	3	3	
Diseño de la InterFaz	Creacion Ventana de Logueo	Miguel Olga Susana	4	4	
	Creación Tablero2	Miguel Olga Susana	3	3	
Prototipo	Implementación prototipo juego	Miguel Olga Susana	8	8	
Implementacion	Generación proyecto Maven	Miguel Olga Susana	5	5	
	Implementación clases iniciales: <ul style="list-style-type: none"> Clase Logueo Clase Tablero2 Clase Casilla 	Miguel Olga Susana	8	6	
	Implementación procedimientos para generar valores en las casillas y sacar panel inicial: <ul style="list-style-type: none"> Clase GrafCasilla Clase PosiMina generarMinasTablero() generarValoresTablero() Añadir dos atributos JPanel a Tablero2 para dividir los campos correspondientes a marcador (minas, reinicio, 	Miguel Olga Susana	8	8	

	cronometro) y la matriz de casillas. <ul style="list-style-type: none"> • generarMenu() 				
	Implementación metodos para cronometrar tiempo de la partida: <ul style="list-style-type: none"> • gestionDeTiempo(), • sumarContador() • getContador(). 	Miguel Olga Susana	4	2	
	Implementacion de procedimientos relacionados con pulsar una casilla y su visualización cuando se pulsa. <ul style="list-style-type: none"> • pulsarCasillasVacias(), • generarEtiquetaValor() • Tratamiento de imágenes 	Miguel Olga Susana	8	6,5	
	Implementación del metodo update del observador(Tablero2) para controlar a todas las notificaciones del objeto observable (Casilla): <ul style="list-style-type: none"> • Respuesta a iniciar. • Respuesta a parar perdiendo. • Respuesta a destapar. 	Miguel Olga Susana	4	4	
Pruebas Unitarias JUnit	JUnit Casilla y PosiMina	Miguel Olga Susana	1	1	Resultado Correcto Runs 14/14 Error 0 Failures 0
Pruebas Generales Integración modelo	Probar paneles esperados según nivel, pulsación de casillas con valor, pulsación casillas com bomba, pulsación casillas vacias, imágenes ganadoras, imágenes perdedoras, ganar/perder según casillas que queden por destapar.	Miguel Olga Susana	8	6	Resultados correctos según imágenes en documentación de pruebas
Corrección de errores en Implementación	Corregir errores en algunos metodos y clases después de pruebas	Miguel Olga Susana	8	3	
Finalizar Documentación	Finalizar Documentación después de corregir errores	Miguel Olga Susana	8	3	

DOCUMENTACIÓN DE PRUEBAS.

*** Con JUnit solo se realizan las pruebas a los modelos sin atributos graficos.
Las pruebas de vistas se realizan de modo visual, sin utilizar JUnit.**

Id.	Objetivo	Entrada	Resultado Esperado	Resultado Obtenido	Coment.
P1	JUnit Casilla	new Casilla(true)	assertNotNull	assertNotNull	Correc.
		new Casilla(-9,"bomba.jpg",0,0))	assertNotNull	assertNotNull	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(4, c1.getValor())	assertEquals(4, c1.getValor())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(0, c1.getEstado())	assertEquals(0, c1.getEstado())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertFalse(c1.getMina())	assertFalse(c1.getMina())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(0, c1.getI())	assertEquals(0, c1.getI())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(1, c1.getJ())	assertEquals(1, c1.getJ())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals("4.PNG", c1.getImagen())	assertEquals("4.PNG", c1.getImagen())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setImg("9.PNG");	assertEquals("9.PNG", c1.getImagen())	assertEquals("9.PNG", c1.getImagen())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setValor(9);	assertEquals(9, c1.getValor())	assertEquals(9, c1.getValor())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setEstado(1);	assertEquals(1, c1.getEstado())	assertEquals(1, c1.getEstado())	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setMina(true);	assertTrue(c1.getMina());	assertTrue(c1.getMina());	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1) c1.setI(2);	assertEquals(2, c1.getI());	assertEquals(2, c1.getI());	Correc.
		Casilla c1 = new Casilla(4,"4.PNG", 0, 1)	assertEquals(3, c1.getJ())	assertEquals(3, c1.getJ())	Correc.
P2	JUnit PosiMina	new PosiMina(1,2)	assertNotNull	assertNotNull	Correc.
		PosiMina pm1 = new PosiMina(1,3)	assertEquals(1,pm1.getFbomb())	assertEquals(1,pm1.getFbomb())	Correc.
		PosiMina pm1 = new PosiMina(1,3)	assertEquals(3,pm1.getCbomb())	assertEquals(3,pm1.getCbomb())	Correc.
P3	Panel nivel 1	Logueo FIG1., datos (Us1, Facil)	Panel-Frame Titulo = "Buscaminas Nv1 – Us1" Minas = 10, cara=nueva, tiempo=0 Filas = 7 casillas Columnas =10 casillas Total casillas = 70	FIG2.	Correc.
	Panel nivel 2	Logueo FIG1., datos (USER1, Medio)	Panel-Frame Titulo = "Buscaminas Nv2 – USER1" Minas = 30, cara=nueva, tiempo=0 Filas = 10 casillas Columnas = 15 casillas Total casillas = 150	FIG3.	Correc.
	Panel nivel 3	Logueo FIG1., datos (USER1, Dificil)	Panel-Frame Titulo = "Buscaminas Nv3 – USER1" Minas = 75, cara=nueva, tiempo=0 Filas = 12 casillas Columnas = 25 casillas Total casillas = 300	FIG4.	Correc.
P4	Conocer valor de una casilla	Panel nivel 1 y conocimiento de la posición de las minas mediante println() a consola. FIG5. Pulsar casillas (2,0) – (5,4)	Panel con casillas (2,0) y (5,4) destapadas	FIG6.	Correc.
P5	Al pulsar mina que se	Panel nivel 1 y conocimiento de la posición de las minas mediante println() a consola FIG7.	Panel con casillas minas descubiertas (la primera en rojo) y resto de casillas tapadas , cara	FIG8	Correc.

	descubran todas las minas y las demas casillas que queden tapadas	Pulsar casilla conocida como mina	de perdedor y juego iniciado (cronometro <>0)		
P6	Al pulsar casilla sin valor descubrir todas sus vecinas hasta casillas vecinas con mina	Panel nivel 1 y conocimiento de la posición de las minas mediante println() a consola FIG9. Pulsar casilla sin valor (sin minas vecinas)	Panel con casilla pulsada descubierta y ademas sus vecinas tambien descubiertas hasta toparse con minas	FIG10	Correc.
P7	Probar opcion ganadora	Panel nivel 1 y conocimiento de la posición de las minas mediante println() a consola FIG11. Jugar con normalidad teniendo la ventaja de que conocemos la posición de las minas ya que las hemos sacado a consola mediante println()	Panel con casillas descubiertas menos las casillas bomba, cara ganadora, numero de bombas igual al numero de casillas no despatadas y cronometro con el tiempo de juego que llevemos, y mensaje de dialogo "GANASTE..."	FIG12	Correc.
P8	Probar opcion perdedora	Panel nivel 1 y conocimiento de la posición de las minas mediante println() a consola FIG13. Jugar con normalidad teniendo la ventaja de que conocemos la posición de las minas ya que las hemos sacado a consola mediante println()	Panel con casillas pulsadas descubiertas, las casillas bombas descubiertas (primera en rojo) y casillas no pulsadas durante el juego cubiertas, cara infeliz y mensaje de dialogo "PERDISTE..."	FIG14	Correc.

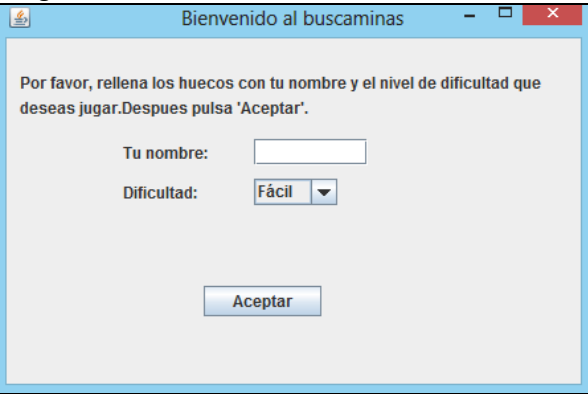
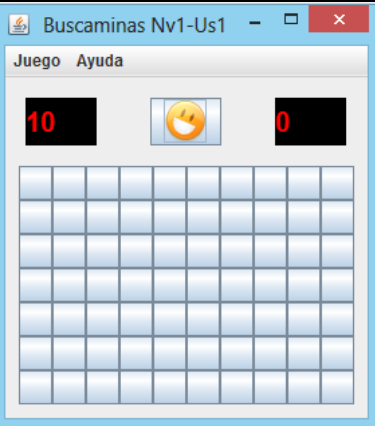
Figuras	Imágenes
FIG1	
FIG2	

FIG3

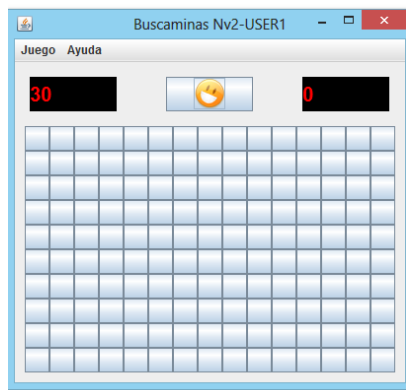


FIG4

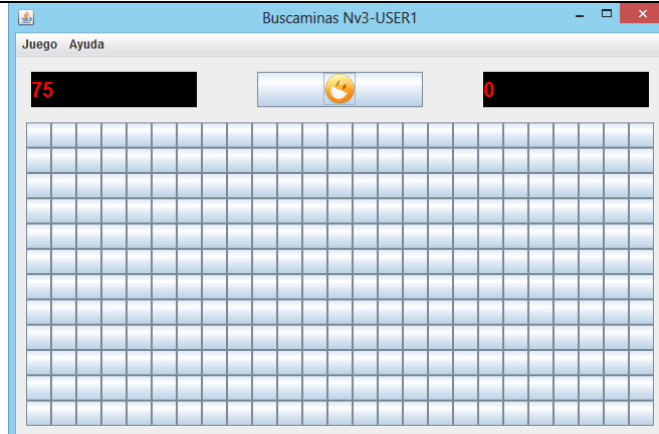


FIG5

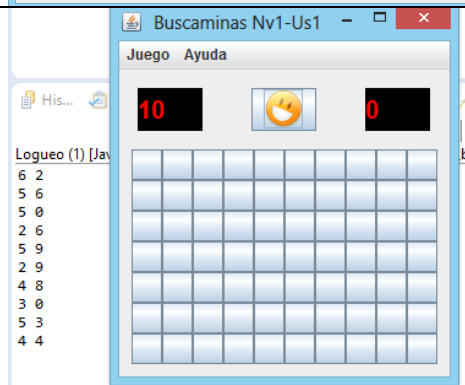
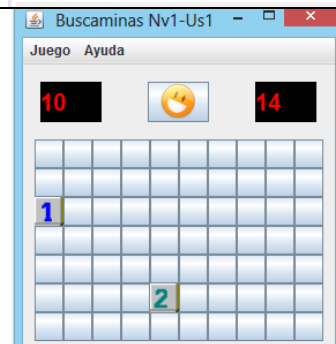


FIG6



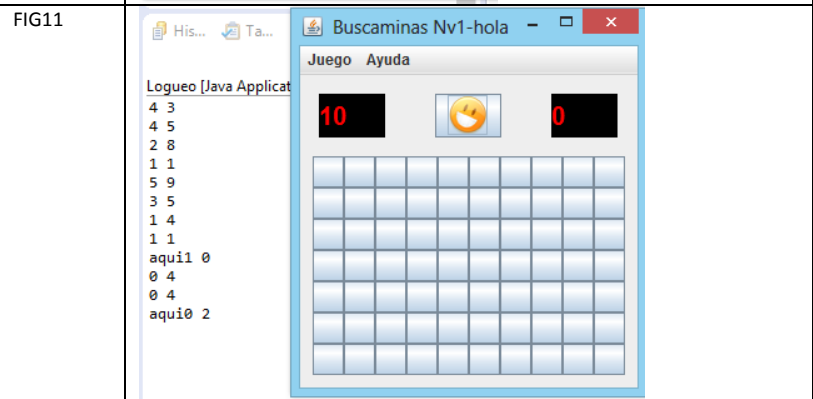
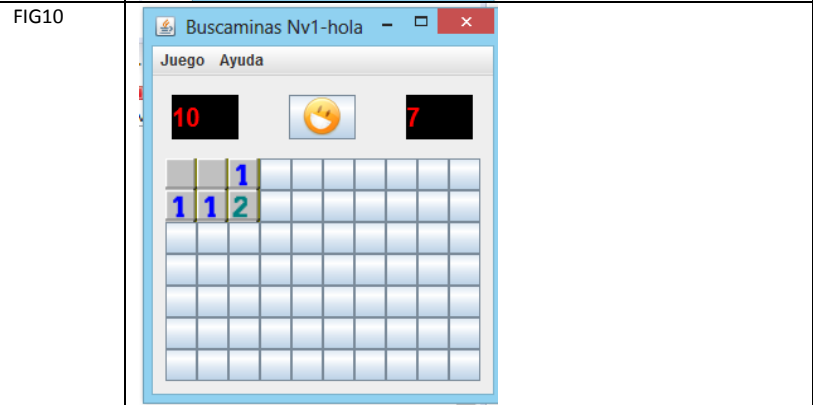
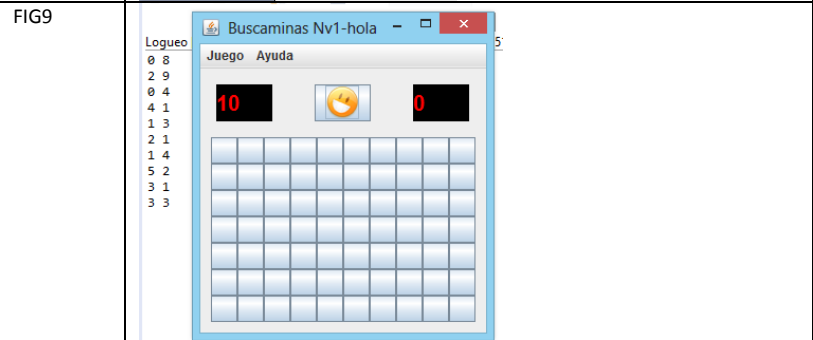
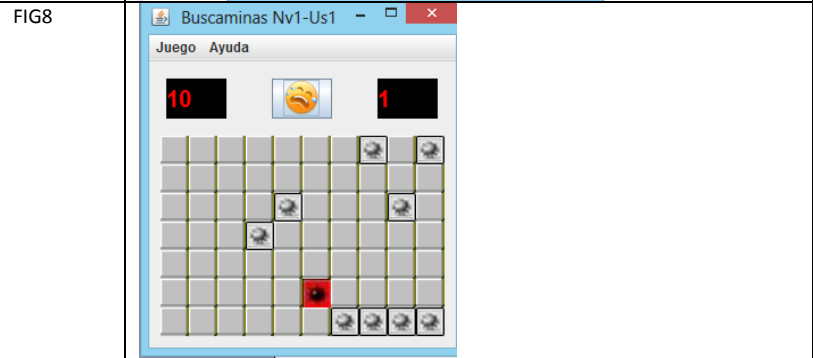
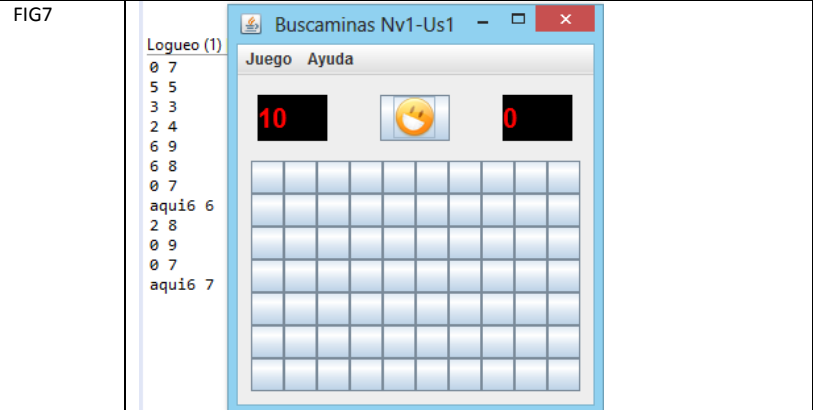


FIG12

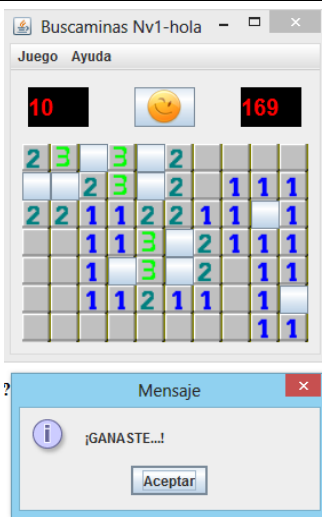


FIG13

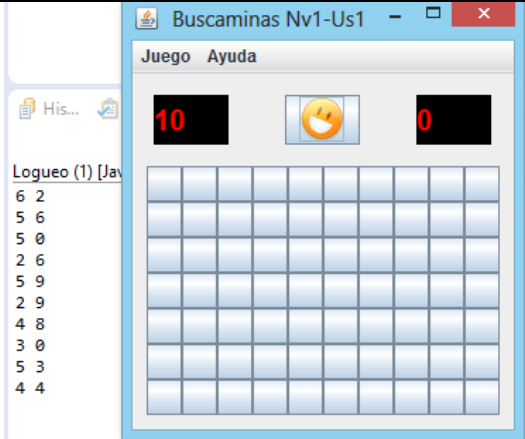


FIG14

