Laboratorio #4

Grupo: 13

Integrantes:

- Añez Vladimirovna Leonardo Henry
- Caricari Torrejon Pedro Luis
- Mercado Oudalova Danilo Anatoli †
- Mollinedo Franco Milena
- Oliva Rojas Gerson

Materia: Interacción Hombre-Computador

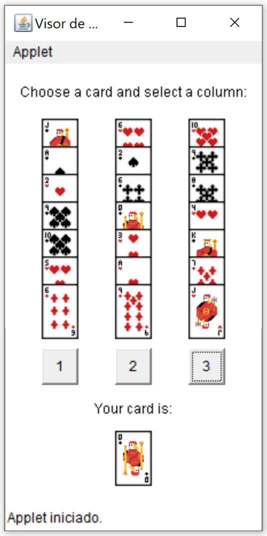
Fecha: 30 de enero de 2020

Porcentaje Completado: 100 %

Comentario(s): En esta práctica utilizamos Componente y Applets, realizamos interfaces graficas con los componentes: `Font`, `Color`, `Label`, `TextField`, `TextArea`, `Button`, `Checkbox`, `Choice`, `List`, `Panel`. Utilizando de manera mezclada `GridLayout`. Algunos con iteracciones, utilizando principios de OOP como la herencia y las interfaces. Tratando de implementar algunos algoritmos de IA así como principios de programacion grafica.

Ejercicio 1:

Este programa es una representación del problema de las 21 Cartas (21 Card Trick), que consiste en escoger una carta e indicar al programa en que columna está, luego de 3 iteraciones el programa adivina la carta del usuario.
Referencia: https://en.wikipedia.org/wiki/Twenty-One_Card_Trick



Choose a card and select a column:

Your card is:

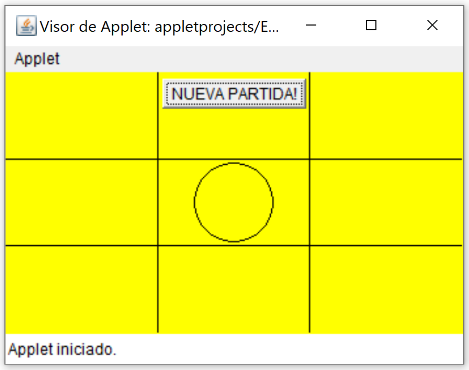
```java
package magictrick;

import java.applet.Applet;
import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Random;
import javax.imageio.ImageIO;
import magictrick.Sprites.SpriteSheetBuilder;
import magictrick.Sprites.SpriteSheet;


public class GuessCard extends Applet implements ActionListener{


    int iter=0,col=-1;
```

```java
private SpriteSheet spriteSheet;
ArrayList<Integer> cards = new ArrayList<>();

ArrayList<Integer> c1 = new ArrayList<>();
ArrayList<Integer> c2 = new ArrayList<>();
ArrayList<Integer> c3 = new ArrayList<>();

Button col1,col2,col3;

public void randomSeed(){

    while(cards.size()<21){
    Random rnd = new Random();
        int seed = rnd.nextInt(52);
        if(!cards.contains(seed)){
             cards.add(seed);
        }

    }

}



@Override
public void init() {
    try{
    //File f = new File("cardsMagic.png");
    BufferedImage sheet =
       ImageIO.read(getClass().getResourceAsStream("/cardsMagic.png"));
    spriteSheet  = new SpriteSheetBuilder().
                withSheet(sheet).
                withColumns(13).
                withRows(4).
                withSpriteCount(52).
                build();
    }catch(Exception e){
    }

    randomSeed();


    col1 = new Button("1");
    col2 = new Button("2");
    col3 = new Button("3");
```

```
        col1.addActionListener(this);
        col2.addActionListener(this);
        col3.addActionListener(this);

        col1.setBounds(32,248,32,32);
        col2.setBounds(96,248,32,32);
        col3.setBounds(160,248,32,32);

        add(col1);
        add(col2);
        add(col3);

        setLayout(new BorderLayout());
        resize(224,384);

    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(iter<3){

            //Borramos toda la lista
            cards.removeAll(cards);

            if(e.getSource()==col1){

                cards.addAll(c2);
                cards.addAll(c1);
                cards.addAll(c3);
                col=0;
                iter++;
            }else
            if(e.getSource()==col2){

                cards.addAll(c1);
                cards.addAll(c2);
                cards.addAll(c3);
                col=1;
                iter++;
             }else
            if(e.getSource()==col3){

                cards.addAll(c1);
                cards.addAll(c3);
                cards.addAll(c2);
                col=2;
                iter++;
            }
            c1.removeAll(c1);
            c2.removeAll(c2);
            c3.removeAll(c3);
            repaint();
            System.out.println(col);
        }

    }

    @Override
    public void paint(Graphics g){
        drawCenteredString("Choose a card and select a column:", 224, 48, g);

        int j=0;
        for(int i=0;i<cards.size();++i){
            int Row = i / 3;
            int Column = i % 3;

            switch(Column){
                case 0:
                    c1.add(cards.get(i));
                    break;
                case 1:
                    c2.add(cards.get(i));
                    break;
                case 2:
                    c3.add(cards.get(i));
                    break;
            }
            g.drawImage(spriteSheet.getSprite(cards.get(i)), 32 + Column*64 ,
                48+24*Row , this);

        }

        if(iter==3){
             drawCenteredString("Your card is:", 224, 600, g);
             g.drawImage(spriteSheet.getSprite(cards.get(10)), 96 , 320 , this);


        }
    }

   public void drawCenteredString(String s, int w, int h, Graphics g) {
    FontMetrics fm = g.getFontMetrics();
    int x = (w - fm.stringWidth(s)) / 2;
    int y = (fm.getAscent() + (h - (fm.getAscent() + fm.getDescent())) / 2);
    g.drawString(s, x, y);
  }

}
```

Ejercicio 2:

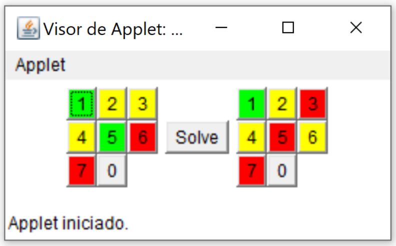
Clasico juego de tres en raya. (TicTacToe)



```java
package Repasando;

import java.applet.Applet;
import java.awt.*;

public class JuegoPuzzle extends Applet {
    @Override
    public void init() {
        add(new Button("NUEVA PARTIDA!"));
        this.setBackground(Color.yellow);
    }

    @Override
    public void paint(Graphics g) {
        int width = this.getWidth() / 3;
        int height = this.getHeight() / 3;
        g.drawOval(width * 3 / 2 - 30, height * 3 / 2 - 30, 60, 60);

        g.drawLine(width, 0, width, height * 3);
        g.drawLine(width * 2, 0, width * 2, height * 3);
        g.drawLine(0, height, width * 3, height);
        g.drawLine(0, height * 2, width * 3, height * 2);
    }
}
```

Ejercicio 3:

Juego del Puzzle

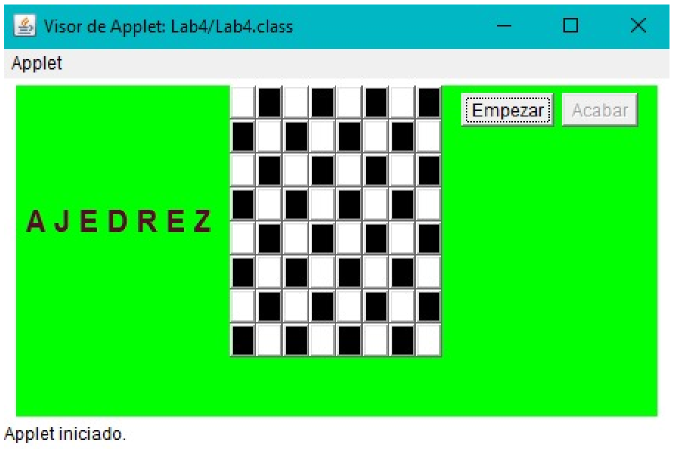


```java

package UtilPanel;

import java.awt.Button;
import java.awt.Color;
import java.awt.Panel;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Puzzle extends Panel{
      private int n,m;
      List<Color> Colores = new ArrayList<>();

      public Puzzle(int n,int m){
          this.m = m;
          this.n = n;
          Colores.add(Color.red);
          Colores.add(Color.yellow);
          Colores.add(Color.green);
      }

      public void generarInicial(){
          for(int i=1;i<n*m-1;++i){
              Button x = new Button(Integer.toString(i));
              Random rnd = new Random();
              x.setBackground(Colores.get(rnd.nextInt(Colores.size())));
              this.add(x);
          }
```

```
            add(new Button("0"));
        }

        public void generarFinal(){
            for(int i=1;i<n*m-1;++i){
                Button x = new Button(Integer.toString(i));
                Random rnd = new Random();
                x.setBackground(Colores.get(rnd.nextInt(Colores.size())));
                this.add(x);
            }
            add(new Button("0"));
        }
}
```

Ejercicio 4:

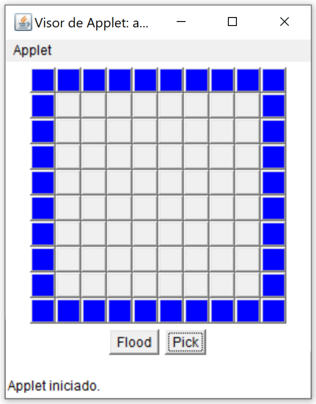
Ajedrez.



```java
package Lab4;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.*;
import java.awt.event.*;

public class Tablero extends Panel{
    public Tablero() {

        setBackground(Color.GREEN);

        Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
        int ancho=300, alto=350;
        setSize(ancho, alto);
        setLocation(d.width/2-ancho/2,d.height/2-alto/2);

        setLayout(new BorderLayout(20,20));

        Font fuente = new Font("Arial", Font.BOLD, 20);
        Label etiq = new Label("A J E D R E Z ", Label.CENTER);
        etiq.setFont(fuente);
        etiq.setForeground(new Color(100,0,50));

        add(etiq, BorderLayout.NORTH);

        Panel tablero = new Panel();
        tablero.setLayout(new GridLayout(8,8));
        for (int i=1; i<=8; i++)
           for (int j=1; j<=8; j++)
                if ((i+j) % 2 == 0) {
                    Button blanca = new Button(" ");
                    blanca.setBackground(Color.white);
                    blanca.setEnabled(false);
                    tablero.add(blanca);
                    }
                else {
                    Button negra = new Button(" ");
                    negra.setBackground(Color.black);
                    negra.setEnabled(false);
                    tablero.add(negra);
                }


        add(tablero,BorderLayout.CENTER);


        Panel botones = new Panel();
        Button empezar = new Button("Empezar");
        Button acabar = new Button("Acabar");

        acabar.setEnabled(false);

        botones.add(empezar);
        botones.add(acabar);

        add(botones, BorderLayout.SOUTH);


        Panel izq = new Panel();
        Panel der = new Panel();
        add(izq,BorderLayout.EAST);
        add(der,BorderLayout.WEST);

    }
}
class ParaAcabar extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
       System.exit(0);
     }
}
```

Ejercicio 5:

Representación interactiva del algoritmo de Flood Fill.
Referencia: https://en.wikipedia.org/wiki/Flood_fill

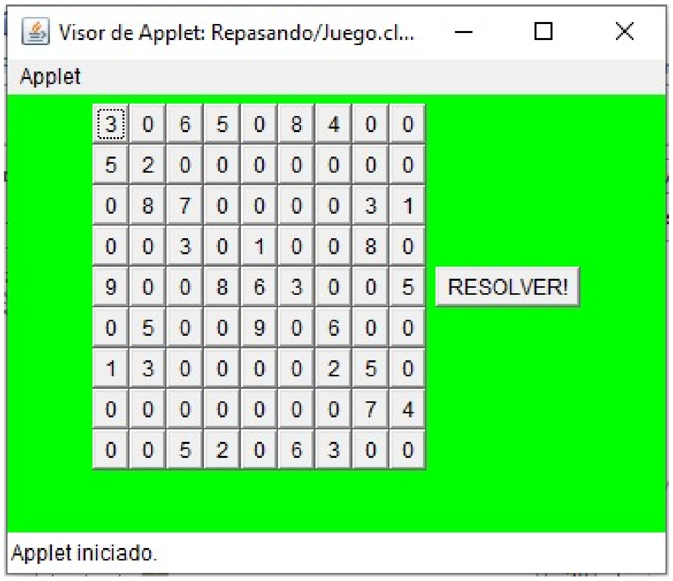


```java
public class Puzzle extends Panel{
      private int n,m;
      List<Color> Colores = new ArrayList<>();

      public Puzzle(int n,int m){
          this.m = m;
          this.n = n;
          Colores.add(Color.red);
          Colores.add(Color.yellow);
          Colores.add(Color.green);
      }

      public void generarInicial(){
          for(int i=1;i<=n;++i){
              for(int j=1;j<=m;++j){

                Button x = new Button("   ");

                if(j==1 || j==m || i==1 || i ==n){
                    x.setBackground(Color.blue);
                }


                this.add(x);
              }

          }
      }
}
```

```java
package appletprojects;

import UtilPanel.Puzzle;
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.GridLayout;


public class Ejercicio extends Applet {

    private Puzzle p1,p2;
    Button b1,b2;

    @Override
    public void init() {

        b1 = new Button("Flood");
        b2 = new Button("Pick");

        p1 = new Puzzle(10, 10);
        p1.setLayout(new GridLayout(10,10));

        p1.generarInicial();


        add(p1);
        add(b1);
        add(b2);
    }


    @Override
    public void paint(Graphics g){

    }
}
```

Ejercicio 6:

Sudoku.



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

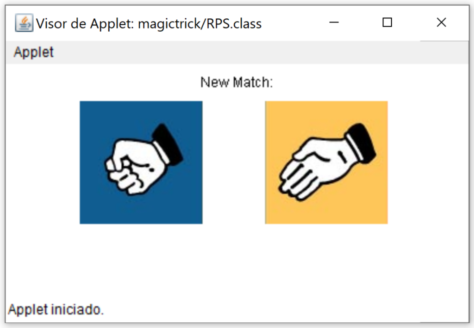
Applet iniciado.

```java
package Repasando;

import java.awt.*;

public class Sudoku extends Panel {

    public static void main(String args[]) {
        new Sudoku(new int[][]{
            {3, 0, 6, 5, 0, 8, 4, 0, 0},
            {5, 2, 0, 0, 0, 0, 0, 0, 0},
            {0, 8, 7, 0, 0, 0, 0, 3, 1},
            {0, 0, 3, 0, 1, 0, 0, 8, 0},
            {9, 0, 0, 8, 6, 3, 0, 0, 5},
            {0, 5, 0, 0, 9, 0, 6, 0, 0},
            {1, 3, 0, 0, 0, 0, 2, 5, 0},
            {0, 0, 0, 0, 0, 0, 0, 7, 4},
            {0, 0, 5, 2, 0, 6, 3, 0, 0}
        }).solve();
    }

    private int sudoku[][];
    private int n = 9;

    public Sudoku(int sudoku[][]) {
        this.sudoku = sudoku;
    }

    public void cargar(){
        for (int i = 0; i < this.sudoku.length; i++) {
            for (int j = 0; j < this.sudoku[i].length; j++) {
                this.add(new Button(Integer.toString(sudoku[i][j])));
            }
        }
    }

    public void solve() {

        if (!backtrackSolve()) {
            System.out.println("This sudoku can't be solved.");
        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(sudoku[i][j] + " ");
            }
            System.out.println();
        }
    }

    public boolean isSuitableToPutXThere(int i, int j, int x) {

        for (int jj = 0; jj < n; jj++) {
            if (sudoku[i][jj] == x) {
                return false;
            }
        }

        for (int ii = 0; ii < n; ii++) {
            if (sudoku[ii][j] == x) {
                return false;
            }
        }

        int boxRow = i - i % 3;
        int boxColumn = j - j % 3;

        for (int ii = 0; ii < 3; ii++) {
            for (int jj = 0; jj < 3; jj++) {
                if (sudoku[boxRow + ii][boxColumn + jj] == x) {
                    return false;
                }
            }
        }

        return true;
    }

    public boolean backtrackSolve() {
        int i = 0, j = 0;
        boolean isThereEmptyCell = false;

        for (int ii = 0; ii < n && !isThereEmptyCell; ii++) {
            for (int jj = 0; jj < n && !isThereEmptyCell; jj++) {
                if (sudoku[ii][jj] == 0) {
                    isThereEmptyCell = true;
                    i = ii;
                    j = jj;
                }
            }
        }

        if (!isThereEmptyCell) {
            return true;
        }

        for (int x = 1; x < 10; x++) {

            if (isSuitableToPutXThere(i, j, x)) {
                sudoku[i][j] = x;

                if (backtrackSolve()) {
                    return true;
                }

                sudoku[i][j] = 0;
            }

        }

        return false;
    }
}


package Repasando;

import java.applet.Applet;
```

```java
import Repasando.Sudoku;
import java.awt.*;

public class Juego extends Applet {

    private Sudoku sudoku;
    private Sudoku sol;

    @Override
    public void init() {
        this.sudoku = new Sudoku(new int[][]{
            {3, 0, 6, 5, 0, 8, 4, 0, 0},
            {5, 2, 0, 0, 0, 0, 0, 0, 0},
            {0, 8, 7, 0, 0, 0, 0, 3, 1},
            {0, 0, 3, 0, 1, 0, 0, 8, 0},
            {9, 0, 0, 8, 6, 3, 0, 0, 5},
            {0, 5, 0, 0, 9, 0, 6, 0, 0},
            {1, 3, 0, 0, 0, 0, 2, 5, 0},
            {0, 0, 0, 0, 0, 0, 0, 7, 4},
            {0, 0, 5, 2, 0, 6, 3, 0, 0}
        });
//          this.add(new Label("RESOLVER SUDOKU"));
        this.sol = sudoku;
        this.sudoku.setLayout(new GridLayout(9, 9));
        sudoku.cargar();
        add(sudoku);
        add(new Button("RESOLVER!"));
        this.setBackground(Color.green);
    }
}   //  END CLASS
```

Ejercicio 7:

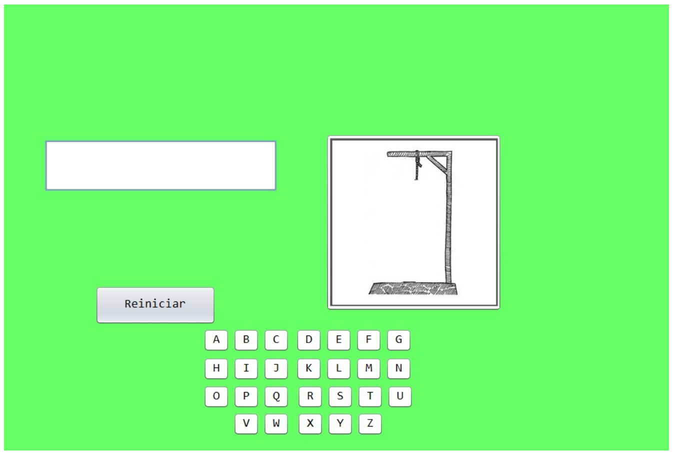
Programa interactivo para jugar Piedra-Papel-Tijeras.



```java
package magictrick;

import java.applet.Applet;
import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Random;
import javax.imageio.ImageIO;
import magictrick.Sprites.SpriteSheetBuilder;
import magictrick.Sprites.SpriteSheet;


public class RPS extends Applet implements ActionListener{

    private SpriteSheet spriteSheet;
    ArrayList<Integer> cards = new ArrayList<>();



    @Override
    public void init() {
        try{
        //File f = new File("cardsMagic.png");
        BufferedImage sheet =
           ImageIO.read(getClass().getResourceAsStream("/rps.jpg"));
        spriteSheet  = new SpriteSheetBuilder().
                        withSheet(sheet).
                        withColumns(3).
                        withRows(1).
                        withSpriteCount(3).
                        build();
        }catch(Exception e){
        }


        setLayout(new BorderLayout());
        resize(400,200);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

    }

    @Override
    public void paint(Graphics g){
         drawCenteredString("New Match:", 400,30, g);
           g.drawImage(spriteSheet.getSprite(0), 64 , 32 , this);
           g.drawImage(spriteSheet.getSprite(1), 192+32 , 32 , this);
    }

      public void drawCenteredString(String s, int w, int h, Graphics g) {
    FontMetrics fm = g.getFontMetrics();
    int x = (w - fm.stringWidth(s)) / 2;
    int y = (fm.getAscent() + (h - (fm.getAscent() + fm.getDescent())) / 2);
    g.drawString(s, x, y);
  }




}
```

Ejercicio 8:

El clasico juego de ahorcado. (Sin interaccion aplicada)

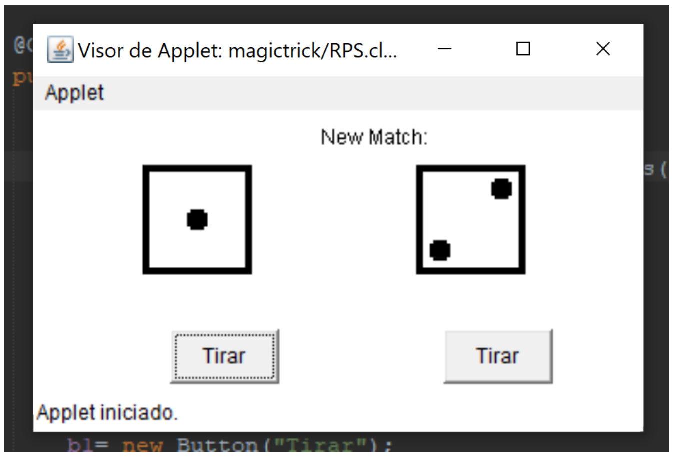


```
package milena;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JOptionPane;


public class Main extends javax.swing.JFrame {


        public ImageIcon imgs[];
        public JButton btns[];
        public String msgs[];
        public int ran;
        public int err;
        public String res[];

        public Main() {
            initComponents();
            imgs = new ImageIcon[6];
            btns = new JButton[27];
            msgs = new String[20];


            imgs[0] = new
                ImageIcon(getClass().getResource("/MexicanHangedPerson/im1.jpg"));
            imgs[1] = new
                ImageIcon(getClass().getResource("/MexicanHangedPerson/im2.jpg"));
            imgs[2] = new
                ImageIcon(getClass().getResource("/MexicanHangedPerson/im3.jpg"));
            imgs[3] = new
                ImageIcon(getClass().getResource("/MexicanHangedPerson/im4.jpg"));
            imgs[4] = new
                ImageIcon(getClass().getResource("/MexicanHangedPerson/im5.jpg"));
            imgs[5] = new
                ImageIcon(getClass().getResource("/MexicanHangedPerson/im6.jpg"));

            //botones para las letras
            btns[1] = jButton2;
            btns[2] = jButton3;
            btns[3] = jButton4;
            btns[4] = jButton5;
            btns[5] = jButton6;
            btns[6] = jButton7;
            btns[7] = jButton8;
            btns[8] = jButton9;
            btns[9] = jButton10;
            btns[10] = jButton11;
            btns[11] = jButton12;
            btns[12] = jButton13;
            btns[13] = jButton14;
            btns[14] = jButton15;
            btns[15] = jButton16;
            btns[16] = jButton17;
            btns[17] = jButton18;
            btns[18] = jButton19;
            btns[19] = jButton20;
            btns[20] = jButton21;
            btns[21] = jButton22;
            btns[22] = jButton23;
            btns[23] = jButton24;
            btns[24] = jButton25;
            btns[25] = jButton26;
            btns[26] = jButton27;


            for (int i = 1; i < 27; i++) {
                btns[i].addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        //checarLetra(e);
                    }
                });
            }
```

```
        iniciar();
    }


    public void iniciar() {

        err = 0;
        jButton1.setIcon(imgs[0]);
        jTextPane1.setText("");

        for (int i = 1; i < 27; i++) {
            btns[i].setEnabled(true);
        }

    }


    private void jButton28ActionPerformed(java.awt.event.ActionEvent evt) {
        iniciar();

        private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        }

        public static void main(String args[]) {

            try {
                for (javax.swing.UIManager.LookAndFeelInfo info:
                    javax.swing.UIManager.getInstalledLookAndFeels()) {
                     if ("Nimbus".equals(info.getName())) {
                         javax.swing.UIManager.setLookAndFeel(info.getClassName());
                         break;
                     }
                }
            } catch (ClassNotFoundException ex) {
                java.util.logging.Logger.getLogger(Main.class.getName()).
                                    log(java.util.logging.Level.SEVERE, null,
                                        ex);
            } catch (InstantiationException ex) {
                java.util.logging.Logger.getLogger(Main.class.getName()).
                                    log(java.util.logging.Level.SEVERE, null,
                                        ex);
            } catch (IllegalAccessException ex) {
                java.util.logging.Logger.getLogger(Main.class.getName()).
                                    log(java.util.logging.Level.SEVERE, null,
                                        ex);
            } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                java.util.logging.Logger.getLogger(Main.class.getName()).
                                    log(java.util.logging.Level.SEVERE, null,
                                        ex);
            }

            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    new Main().setVisible(true);
                }
            });
        }

        private javax.swing.JButton jButton1;
        private javax.swing.JButton jButton10;
        private javax.swing.JButton jButton11;
        private javax.swing.JButton jButton12;
        private javax.swing.JButton jButton13;
        private javax.swing.JButton jButton14;
        private javax.swing.JButton jButton15;
        private javax.swing.JButton jButton16;
        private javax.swing.JButton jButton17;
        private javax.swing.JButton jButton18;
        private javax.swing.JButton jButton19;
        private javax.swing.JButton jButton2;
        private javax.swing.JButton jButton20;
        private javax.swing.JButton jButton21;
        private javax.swing.JButton jButton22;
        private javax.swing.JButton jButton23;
        private javax.swing.JButton jButton24;
        private javax.swing.JButton jButton25;
        private javax.swing.JButton jButton26;
        private javax.swing.JButton jButton27;
        private javax.swing.JButton jButton28;
        private javax.swing.JButton jButton3;
        private javax.swing.JButton jButton4;
        private javax.swing.JButton jButton5;
        private javax.swing.JButton jButton6;
        private javax.swing.JButton jButton7;
        private javax.swing.JButton jButton8;
        private javax.swing.JButton jButton9;
        private javax.swing.JLabel jLabel3;
        private javax.swing.JPanel jPanel1;
        private javax.swing.JScrollPane jScrollPane1;
        private javax.swing.JTextPane jTextPane1;
    }
```

Ejercicio 9:

Juego de lanzamiento de dados.

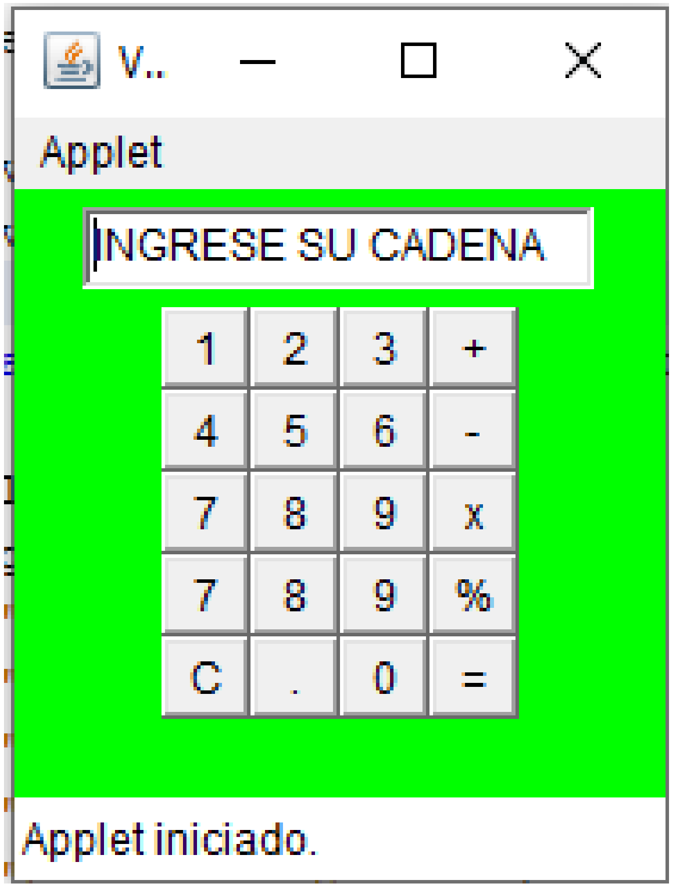


```java
package magictrick;

import java.applet.Applet;
import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Random;
import javax.imageio.ImageIO;
import magictrick.Sprites.SpriteSheetBuilder;
import magictrick.Sprites.SpriteSheet;


public class RPS extends Applet implements ActionListener{

    private SpriteSheet spriteSheet;
    ArrayList<Integer> cards = new ArrayList<>();
    Button b1,b2;


    @Override
    public void init() {
        try{
        //File f = new File("cardsMagic.png");
        BufferedImage sheet =
            ImageIO.read(getClass().getResourceAsStream("/dice.png"));
        spriteSheet  = new SpriteSheetBuilder().
                        withSheet(sheet).
                        withColumns(6).
                        withRows(2).
                        withSpriteCount(12).
                        build();
        }catch(Exception e){
        }
        b1= new Button("Tirar");
        b2= new Button("Tirar");

        b1.setBounds(80,128,64,32);
        b2.setBounds(240,128,64,32);
            add(b1);
            add(b2);
        setLayout(new BorderLayout());
        resize(400,200);

    }

    @Override
    public void actionPerformed(ActionEvent e) {

    }

    @Override
    public void paint(Graphics g){
         drawCenteredString("New Match:", 400,30, g);
           g.drawImage(spriteSheet.getSprite(0), 64 , 32 , this);
           g.drawImage(spriteSheet.getSprite(1), 192+32 , 32 , this);
    }

      public void drawCenteredString(String s, int w, int h, Graphics g) {
    FontMetrics fm = g.getFontMetrics();
    int x = (w - fm.stringWidth(s)) / 2;
    int y = (fm.getAscent() + (h - (fm.getAscent() + fm.getDescent())) / 2);
    g.drawString(s, x, y);
  }
```

```
}
```

Ejercicio 10:

Calculadora de enteros.



INGRESE SU CADENA

Applet iniciado.

```
public class Calculadora extends Panel {
    private String m[][] = {
        {"1", "2", "3", "+"},
        {"4", "5", "6", "-"},
        {"7", "8", "9", "x"},
        {"7", "8", "9", "%"},
        {"C", ".", "0", "="},
    };

    public Calculadora(String m[][]){
        this.m = m;
    }

    public void cargar(){
        for (int i = 0; i < m.length; i++) {
            for (int j = 0; j < m[i].length; j++) {
                this.add(new Button(m[i][j]));
            }
        }
```

```java
    }

    public double  sum(double a, double b){
        return a + b;
    }

    public double  mul(double a, double b){
        return a * b;
    }

}

package Repasando;

import java.applet.Applet;
import java.awt.*;

public class Juego extends Applet {

    Calculadora cal;
    String m[][] = {
        {"1", "2", "3", "+"},
        {"4", "5", "6", "-"},
        {"7", "8", "9", "x"},
        {"7", "8", "9", "%"},
        {"C", ".", "0", "="}
    };
    @Override
    public void init() {
        this.add(new TextField("INGRESE SU CADENA"));
        this.cal = new Calculadora(m);
        this.cal.setLayout(new GridLayout(5, 4));
        this.cal.cargar();
        this.add(cal);
        this.setBackground(Color.green);
    }
}   //  END CLASS
```