Laboratorio #2

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

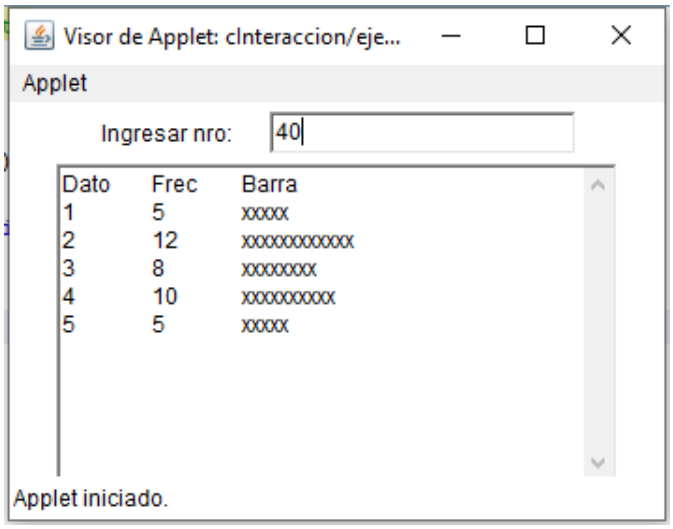
Fecha: 24 de enero de 2020

Porcentaje Completado: 100 %

Comentario(s): En esta práctica aprendimos a solucionar los problemas asignados previamente de una manera más eficiente usando los applets con interacción utilizando Event y Object, aprendimos a usar la función Action y poder realizar los ejercicios aún más rápido, trabajamos en conjunto y nos organizamos como grupo. Realizamos programas mas interactivos con entrada por teclado mostrando estos resultados en el applet.

Ejercicio 1:

Implementar una applet para generar aleatoriamente n-elementos iniciales que contienen respuestas de 1 a 5. Encontrar la cantidad total de cada opción elegida y mostrar el número de opción, cantidad y barra de frecuencias. Utilizar un arreglo contador.

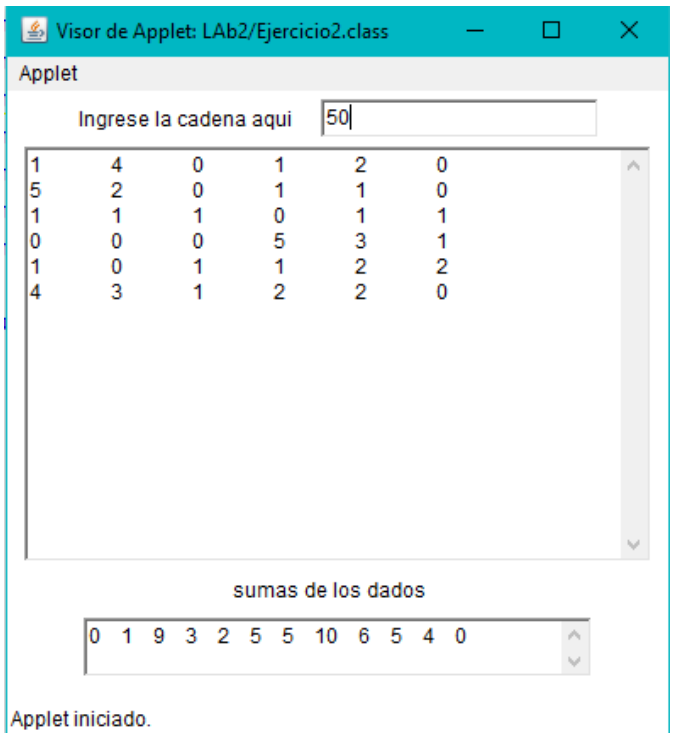



```java
package cInteraccion;

import java.applet.Applet;
import java.awt.Event;
import java.awt.Label;
import java.awt.TextArea;
import java.awt.TextField;
import java.util.Random;

public class ejercicio1 extends Applet {

    //  VAR
    private int n;
    private int a[];
    private int f[];
    private Label ls;
    private TextField tf;
    private TextArea ta;

    //  METHODS
    @Override
    public void init() {
        this.n = 0;
        this.f = new int[6];
        this.tf = new TextField(20);
        this.ls = new Label("Ingresar nro: ");
```

```
        this.ta = new TextArea(10, 20);
        this.add(ls);
        this.add(tf);
        this.add(ta);
    }

    @Override
    public boolean action(Event e, Object o) {
        if (e.target == tf) {
            String number = tf.getText();
            this.n = Integer.valueOf(number);
            randomPool(n, f);
            ta.append("Dato\tFrec\tBarra");
            for (int i = 1; i < f.length; i++) {
                ta.append(fila(i, f[i]));
            }
        }
        return true;
    }

    public String cadena(char ch, int n) {
        String s = "";
        for (int i = 0; i < n; i++) {
            s += ch;
        }
        return s;
    }

    public String fila(int dato, int frec) {
        return "\t" + dato + "\t" + frec + "\t" + cadena('x', frec) + "\n";
    }

    public void randomPool(int n, int f[]) {
        Random random = new Random();
        for (int i = 0; i < n; i++) {
            int index = random.nextInt(5) + 1;
            f[index]++;
        }
    }

}
```

Ejercicio 2:

Implementar una applet para simular n-lanzamientos de un par de datos. a) Encontrar una matriz de contingencia. Las filas representan a un dado y las columnas al otro dado, es decir; la cantidad de veces de resultados similares en los lanzamientos de dados. (Utilizar una matriz de contadores), b) encontrar la frecuencia del total de la suma de ambos dados en los n-lanzamientos, mostrar cantidades presentadas para cada suma.



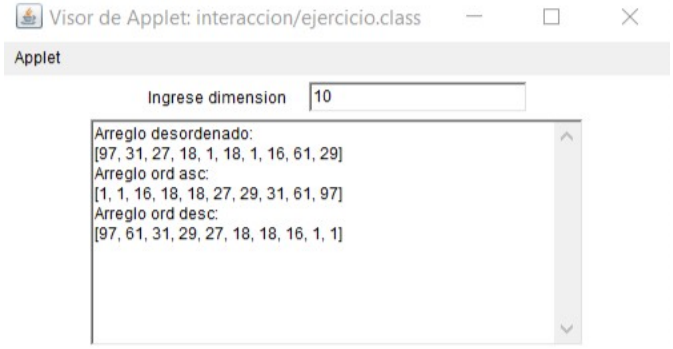

```
package LAb2;

import java.applet.Applet;
import java.applet.Applet;
import java.awt.Event;
import java.awt.Graphics;
import java.awt.Label;
import java.awt.TextArea;
import java.awt.TextField;
import java.util.Random;

public class Ejercicio2 extends Applet {


    private Label ls;
    private Label ls2;
    private TextField tf;
    private TextArea ta;
    private TextArea ta1;
```

```
    static int dim = 60;
    static int m[][] = new int[7][7];
    static int f[] = new int[13];


    public void init() {
        ls = new Label("Ingrese la cadena aqui");
        tf = new TextField(20);
        ta = new TextArea(15,50);
        ta1 = new TextArea(1,40);
        this.add(ls);
        this.add(tf);
        this.add(ta);
        ls2 = new Label("sumas de los dados");
        this.add(ls2);
        this.add(ta1);
    }

    public void mostrarRespuesta() {
        for (int i = 1; i < 7; i++) {
            for (int j = 1; j < 7; j++) {
                ta.append("" + m[i][j] + "\t");
            }
            ta.append("\n"+"");
        }

        for (int i = 1; i < 13; i++) {
            ta1.append("" + f[i] +"    ");
        }
    }

//    public void adjust() {
//        this.setSize(dim * 10, dim * 7);
//    }

    public void randomPool(int a) {
        Random random = new Random();
        for (int i = 0; i < a; i++) {
            int dice1 = random.nextInt(6) + 1;
            int dice2 = random.nextInt(6) + 1;
            f[dice1 + dice2]++;
            m[dice1][dice2]++;
        }
    }
```

```
    public boolean action( Event e, Object o){
        if (e.target == tf) {
            int a = Integer.valueOf(tf.getText());
            randomPool(a);
            mostrarRespuesta();
        }
        return true;
    }
}
```

Ejercicio 3:

Implementar una applet, para mostrar un arreglo de n-elementos y volver a mostrar el mismo arreglo ordenado en forma ascendente y ordenado en forma descendente.

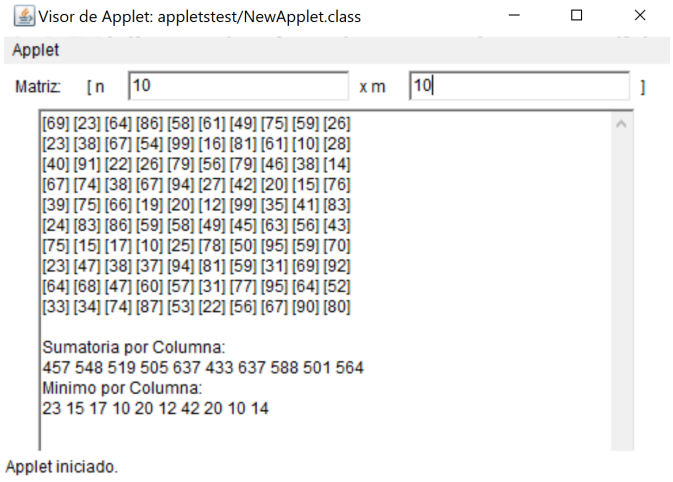



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package interaccion;

import java.applet.Applet;
import java.awt.Event;
import java.awt.Label;
import java.awt.TextArea;
import java.awt.TextField;
import java.util.Arrays;
import java.util.Random;

/**
 *
 * @author Milena
 */
public class ejercicio extends Applet {

    int dim;
    int a[];
    private Label ls;
    private TextField tf;
    private TextArea ta;

    @Override
    public void init() {
        this.setSize(500, 500);
```

```
        dim = 0;
        a = new int[dim];
        ls = new Label("Ingrese dimension");
        tf = new TextField(20);
        ta = new TextArea(10, 50);
        this.add(ls);
        this.add(tf);
        this.add(ta);
    }

    public void cargarVector(int vector[]) {
        Random random = new Random();
        for (int i = 0; i < vector.length; i++) {
            vector[i] = random.nextInt(100);
        }
    }

    public int[] invertir(int a[]) {
        int[] invertido = new int[a.length];
        for (int i = 0; i < a.length; i++) {
            invertido[i] = a[a.length - 1 - i];
        }
        return invertido;
    }

    @Override
    public boolean action(Event e, Object o) {
        if (e.target == tf) {
            String S1 = tf.getText();
            dim = Integer.valueOf(S1);
            a = new int[dim];
            cargarVector(a);
            ta.append("Arreglo desordenado: \n"+Arrays.toString(a) + "\n");
            Arrays.sort(a);
            ta.append("Arreglo ord asc: \n"+Arrays.toString(a) + "\n");

            ta.append("Arreglo ord desc: \n"+Arrays.toString(invertir(a)) + "\n");


        }
        return true;
    }

}
```

Ejercicio 4:

Implementar una applet para mostrar una matriz rectangular de n x m (Generar aleatoriamente los elementos), y mostrar la suma de los elementos de las columnas y los elementos menores de cada columna.




```
package appletstest;


import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;

public class NewApplet extends Applet implements KeyListener{

    private String s1;
    private Label ls,ls1,ls2,ls3;
    private TextField tf1;
    private TextField tf2;
    private TextArea ta;
@Override
public void keyPressed( KeyEvent e ) { }
@Override
public void keyReleased( KeyEvent e ) { }
@Override
public void keyTyped( KeyEvent e ) {
      char c = e.getKeyChar();
      if ( c != KeyEvent.CHAR_UNDEFINED ) {

         repaint();
         e.consume();
      }
   }
```

```
    public boolean action(Event e,Object o){


        if(e.target==tf1 || e.target==tf2){

                fila = Integer.valueOf(tf1.getText());
                columna = Integer.valueOf(tf2.getText());
                matrix = new int[fila][columna];
                randomMatrix();



        }
        return true;
    }



    int matrix[][],fila,columna;
    int sumcol[]= new int[1000];
    int mincol[]= new int[1000];

    public void randomMatrix(){
        Random random = new Random();

        for(int i=0;i<columna;++i){
            mincol[i]=Integer.MAX_VALUE;
        }

        for(int i=0;i<fila;++i){
            for(int j=0;j<columna;++j){
                int rnd = 10+random.nextInt(90);

                matrix[i][j]=rnd;
                sumcol[j]+=rnd;
                mincol[j] = Math.min(rnd, mincol[j]);

            }
        }
```

```
        for(int i=0;i<fila;++i){
            for(int j=0;j<columna;++j){
                ta.append("["+matrix[i][j]+"] ");
            }
            ta.append("\n");
        }
        ta.append("\n");
        ta.append("Sumatoria por Columna: \n");
        for(int i=0;i<columna;++i){
            ta.append(""+sumcol[i]+" ");
        }
        ta.append("\n");
        ta.append("Minimo por Columna: \n");
        for(int i=0;i<columna;++i){
            ta.append(""+mincol[i]+" ");
        }

    }

    public void init() {

      addKeyListener( this );
        ls = new Label("Matriz:");
        ls1 = new Label("[ n");
        tf1 = new TextField(20);
        ls2 = new Label("x m");
        tf2 = new TextField(20);
        ls3 = new Label("]");
        ta = new TextArea(20,60);

        this.add(ls);
        this.add(ls1);
        this.add(tf1);
        this.add(ls2);
        this.add(tf2);
        this.add(ls3);
        this.add(ta);



    }

    // TODO overwrite start(), stop() and destroy() methods
```

```
}
```

Ejercicio 5:

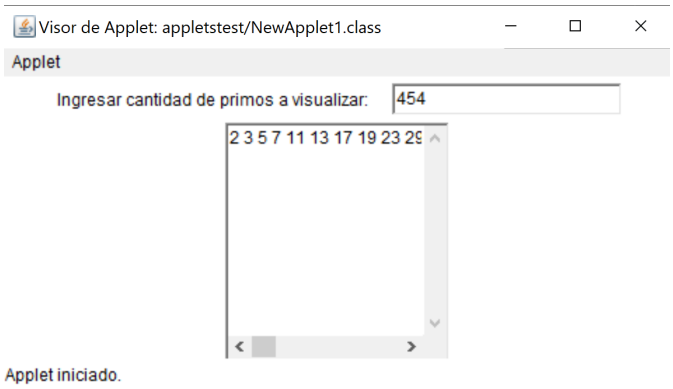
Implementar una applet, para mostrar todas las subcadenas de una cadena de entrada. Es decir; mostrar todas las subcadenas de la cadena leida.


```
package appletstest;

import java.awt.Label;
import java.awt.TextArea;
import java.awt.TextField;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javafx.event.Event;

public class Cadena extends java.applet.Applet {

    private String s1;
```

```
    private Label ls;
    private TextField tf;
    private TextArea ta;


    public void init() {
        ls = new Label("Ingresar cadena: ");
        tf = new TextField(20);
        ta = new TextArea(10,20);
        this.add(ls);
        this.add(tf);
        this.add(ta);
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }


    public boolean action(Event e,Object o){

        if(e.getTarget()==tf){
            s1 = tf.getText();
            for(int i=0;i<s1.length();++i){
                for(int j=i+1;j<= s1.length();++j){
                    ta.append(s1.substring(i,j)+"\n");
                }
            }
        }
        return true;
    }

    private void initComponents() {

        setLayout(new java.awt.BorderLayout());
    }
}
```

Ejercicio 6:

Muestra la cantidad de primos ingresados.

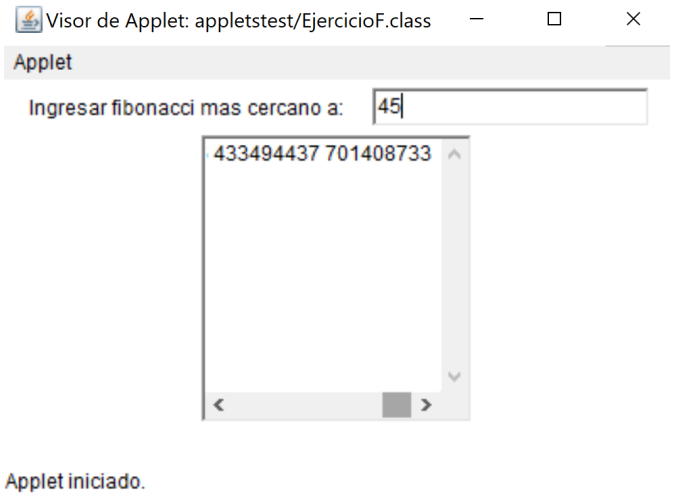



```
package appletstest;

import java.applet.*;
import java.awt.*;
import java.awt.event.*;


public class NewApplet1 extends Applet{

    private String s1;
    private Label ls;
    private TextField tf;
    private TextArea ta;


    @Override
    public void init() {
        ls = new Label("Ingresar cantidad de primos a visualizar:");
        tf = new TextField(20);
        ta = new TextArea(10, 20);
        this.add(ls);
        this.add(tf);
        this.add(ta);

    }


    @Override
    public boolean action(Event e, Object o) {
        System.out.println("entra");
        if (e.target == tf) {
```

```
            s1 = tf.getText();
            int n = Integer.valueOf(s1);
            System.out.println(n);
            int c = 1;
            int p = 2;
            int d = 2;
            while (c <= n) {
                if (p % d == 0) {
                    if (p == d) {
                        ta.append(String.valueOf(p)+" ");
                        c++;
                    }
                    d = 2;
                    p++;
                } else {
                    d++;
                }
            }
        }
        return true;
    }
}
```

Ejercicio 7:

N-Fibonacci

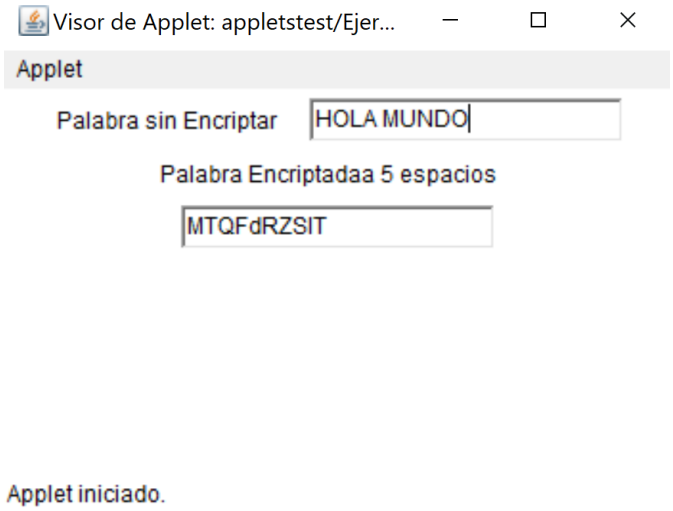



```
package appletstest;

import java.awt.Event;
import java.awt.Label;
import java.awt.TextArea;
import java.awt.TextField;


public class EjercicioF extends java.applet.Applet {

    private String s1;
    private Label ls;
    private TextField tf;
    private TextArea ta;

    @Override
    public void init() {
        ls = new Label("Ingresar fibonacci mas cercano a:");
        tf = new TextField(20);
        ta = new TextArea(10, 20);
        this.add(ls);
        this.add(tf);
        this.add(ta);

    }

    @Override
    public boolean action(Event e, Object o) {
```

```
        System.out.println("entra");
        if (e.target == tf) {
            s1 = tf.getText();
            int n = Integer.valueOf(s1);
                    int numero = 10, fibo1, fibo2, i;

        fibo1 = 1;
        fibo2 = 1;
        ta.append(String.valueOf(fibo1) + " ");
        for (i = 2; i <= n; i++) {
            ta.append(String.valueOf(fibo1) + " ");
            fibo2 = fibo1 + fibo2;
            fibo1 = fibo2 - fibo1;
        }
        }
        return true;
    }

    private void initComponents() {

        setLayout(new java.awt.BorderLayout());
    }
}
```

Ejercicio 8:

Encriptar Cesar con 5 Espacios



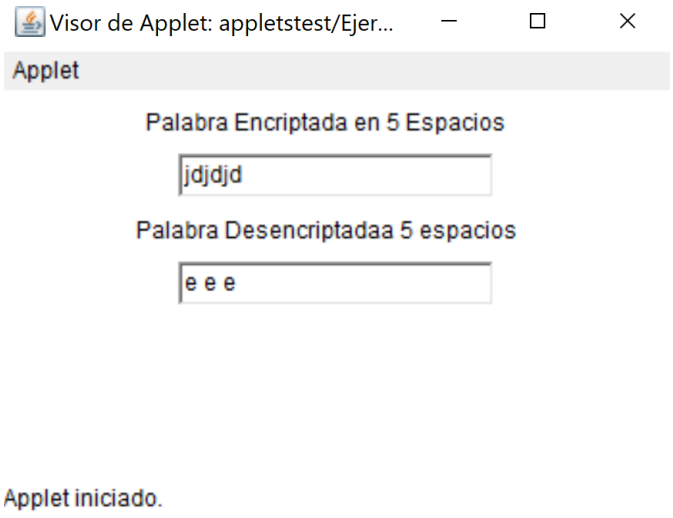

```
package appletstest;

import java.awt.Event;
import java.awt.Label;
import java.awt.TextField;


public class Ejercicio8 extends java.applet.Applet {
    private String s1;
    private Label ls;
    private TextField tf;
    private TextField tf2;
    private Label ls2;

    public void init() {
        ls = new Label("Palabra sin Encriptar");
        tf = new TextField(20);
        ls2 = new Label("Palabra Encriptadaa 5 espacios");
        tf2 = new TextField(20);
        this.add(ls);
        this.add(tf);
        this.add(ls2);
        this.add(tf2);

    }


    @Override
```

```
    public boolean action(Event e, Object o) {
        System.out.println("entra");
        if (e.target == tf) {
            s1 = tf.getText();
            tf2.setText(encrypt(s1, 5));
        }
        return true;
    }

    static char[] chars = {
        'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
        'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
        'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
        'y', 'z', '0', '1', '2', '3', '4', '5',
        '6', '7', '8', '9', 'A', 'B', 'C', 'D',
        'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
        'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
        'U', 'V', 'W', 'X', 'Y', 'Z', '!', '@',
        '#', '$', '%', '^', '&', '(', ')', '+',
        '-', '*', '/', '[', ']', '{', '}', '=',
        '<', '>', '?', '_', '"', '.', ',', ' '
    };
    static String encrypt(String text, int offset)
    {
        char[] plain = text.toCharArray();

        for (int i = 0; i < plain.length; i++) {
            for (int j = 0; j < chars.length; j++) {
                if (j <= chars.length - offset) {
                    if (plain[i] == chars[j]) {
                        plain[i] = chars[j + offset];
                        break;
                    }
                }
                else if (plain[i] == chars[j]) {
                    plain[i] = chars[j - (chars.length - offset + 1)];
                }
            }
        }
        return String.valueOf(plain);
    }


    private void initComponents() {
```

```
        setLayout(new java.awt.BorderLayout());
    }
}
```

Ejercicio 9:

Desencriptar Cesar con 5 espacios

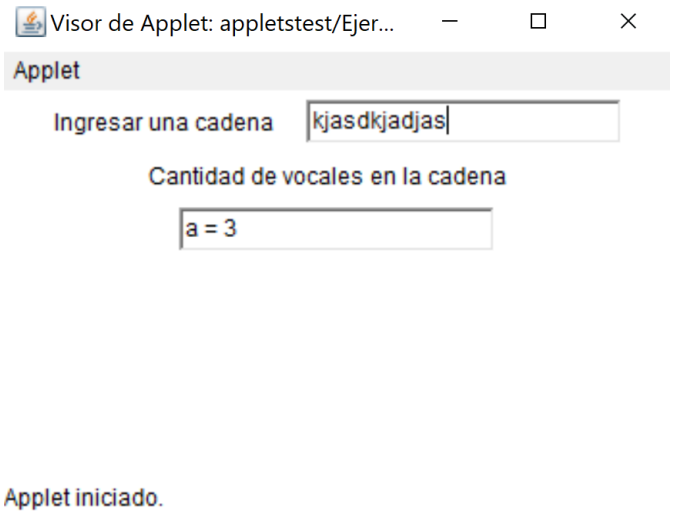



```
package appletstest;

import java.awt.Event;
import java.awt.Label;
import java.awt.TextField;

public class Ejercicio9 extends java.applet.Applet {
private String s1;
    private Label ls;
    private TextField tf;
    private TextField tf2;
    private Label ls2;

        @Override
    public void init() {
        ls = new Label("Palabra Encriptada en 5 Espacios");
        tf = new TextField(20);
        ls2 = new Label("Palabra Desencriptadaa 5 espacios");
        tf2 = new TextField(20);
        this.add(ls);
        this.add(tf);
        this.add(ls2);
        this.add(tf2);

    }


        @Override
```

```
    public boolean action(Event e, Object o) {
        System.out.println("entra");
        if (e.target == tf) {
            s1 = tf.getText();
            tf2.setText(decrypt(s1, 5));
        }
        return true;
    }
    static char[] chars = {
        'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
        'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
        'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
        'y', 'z', '0', '1', '2', '3', '4', '5',
        '6', '7', '8', '9', 'A', 'B', 'C', 'D',
        'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
        'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
        'U', 'V', 'W', 'X', 'Y', 'Z', '!', '@',
        '#', '$', '%', '^', '&', '(', ')', '+',
        '-', '*', '/', '[', ']', '{', '}', '=',
        '<', '>', '?', '_', '"', '.', ',', ' '
    };
    static String decrypt(String cip, int offset)
    {
        char[] cipher = cip.toCharArray();
        for (int i = 0; i < cipher.length; i++) {
            for (int j = 0; j < chars.length; j++) {
                if (j >= offset && cipher[i] == chars[j]) {
                    cipher[i] = chars[j - offset];
                    break;
                }
                if (cipher[i] == chars[j] && j < offset) {
                    cipher[i] = chars[(chars.length - offset +1) + j];
                    break;
                }
            }
        }
        return String.valueOf(cipher);
    }
    private void initComponents() {

        setLayout(new java.awt.BorderLayout());
    }
}
```

Ejercicio 10:

Cantidad de Vocales en una Cadena




```
package appletstest;

import java.applet.Applet;
import java.awt.Event;
import java.awt.Label;
import java.awt.TextField;

public class Ejercicio9 extends Applet {

    private String s1;
    private Label ls;
    private TextField tf;
    private TextField tf2;
    private Label ls2;

    @Override
    public void init() {
        ls = new Label("Ingresar una cadena");
        tf = new TextField(20);
        ls2 = new Label("Cantidad de vocales en la cadena");
        tf2 = new TextField(20);
        this.add(ls);
        this.add(tf);
        this.add(ls2);
        this.add(tf2);

    }
    private String cad = "En la materia de interaccion hombre computador";
```

```
    private int v[] = {0, 0, 0, 0, 0};
    private String vs[] = {"a", "e", "i", "o", "u"};

    @Override
    public boolean action(Event e, Object o) {
        System.out.println("entra");
        if (e.target == tf) {
            s1 = tf.getText();
            cantidadVocales(s1, v);
            String aux = "";
            for (int i = 0, fila = 30; i < v.length; i++, fila += 30) {

                aux += vs[i]+" = "+v[i]+"\n";
            }
            tf2.setText(aux);
        }
        return true;
    }

    public void cantidadVocales(String cad, int v[]) {
        cad = cad.toLowerCase();
        for (int i = 0; i < cad.length(); i++) {
            char c = cad.charAt(i);
            switch (c) {
                case 97:
                    v[0]++;
                    break;
                case 101:
                    v[1]++;
                    break;
                case 105:
                    v[2]++;
                    break;
                case 111:
                    v[3]++;
                    break;
                case 117:
                    v[4]++;
                    break;

            }
        }

    }
}
```