LABORATORIO #1

Grupo 13

Integrantes:

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

Materia: Interacción Hombre Computador

Fecha: 20/01/2020

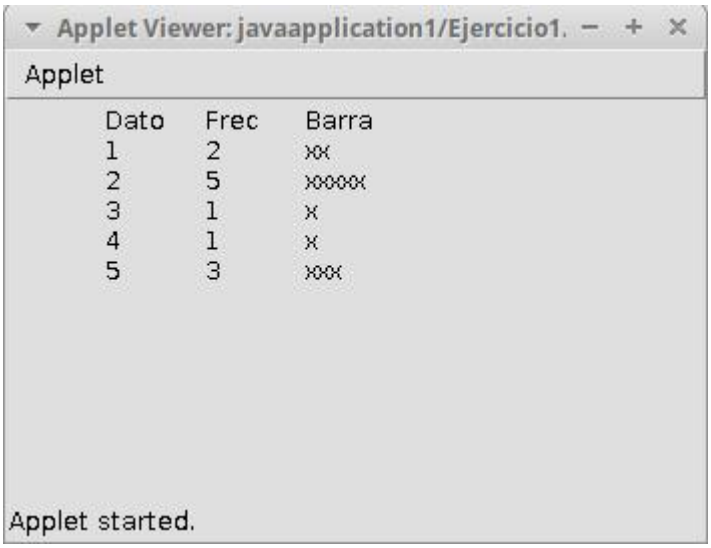
Porcentaje completado: 100%

Comentario:

Con respecto a los ejercicios asignados hemos tenido un satisfecho aprendizaje en cuanto a las diferentes clasificaciones en Java, pudimos enriquecernos aún más sobre las diferentes utilidades en la clase Applet, tener un trabajo mucho mejor realizado en cuanto a las gráficas y la lógica de los ejercicios previamente propuestos. Tuvimos la oportunidad de plantear nuestros propios programas a realizar y eso hizo que nuestra creatividad se explaye, además de mejorar nuestra lógica con esta nueva clase. En cuanto a trabajo en grupo, pudimos organizarnos mejor y coincidir en muchas opiniones.

Ejercicio 1:

Implementar una applet (sin interacción), para procesar un arreglo con elementos iniciales que contienen respuestas de 1 a 5, de selección. Encontrar la cantidad total de cada opción elegida y mostrar el número de opción, cantidad y barra de frecuencias. Utilizar un arreglo contador.

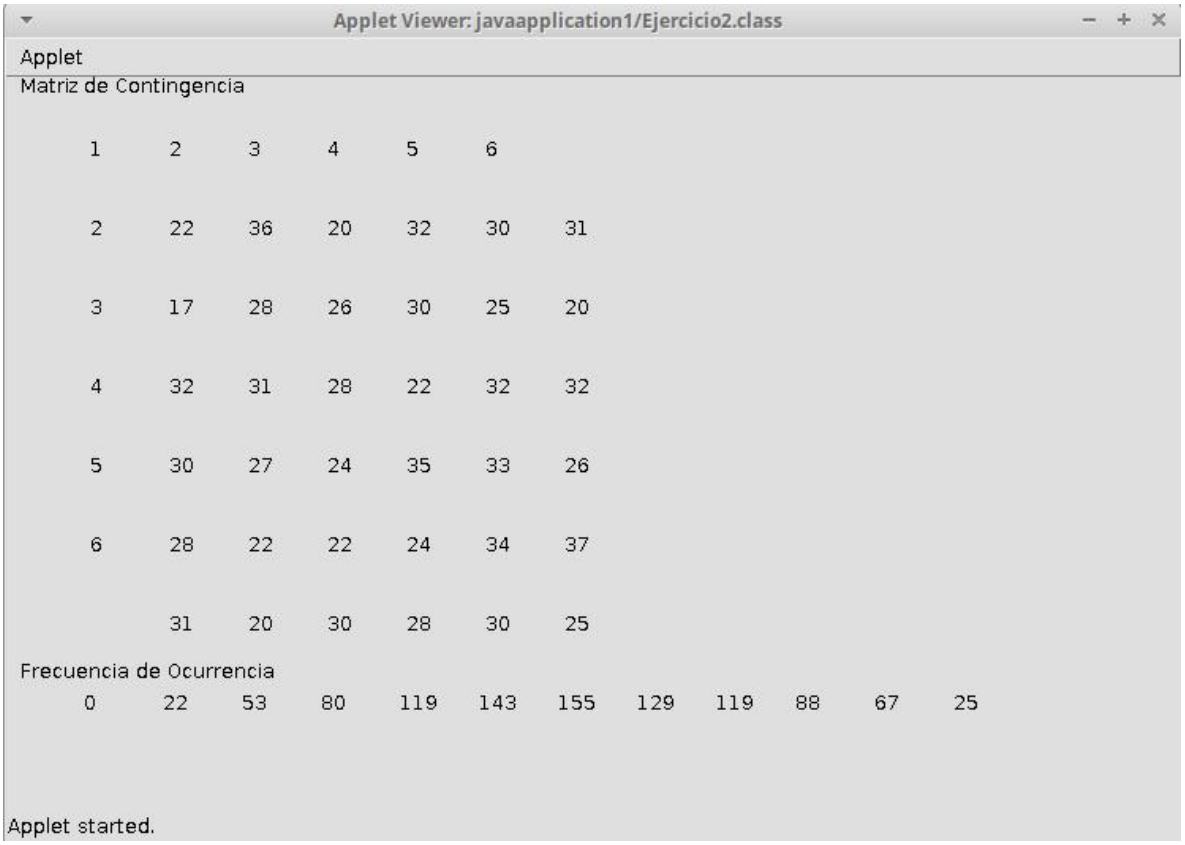



```
package interaccion;
import java.applet.Applet;
import java.awt.Graphics;

public class ejercicio1 extends Applet {

    int a[] = {2, 1, 5, 2, 3, 1, 4, 5, 5, 2, 2, 2};
    int f[] = {0, 0, 0, 0, 0, 0};

    @Override
    public void init() {

    }

    @Override
    public void paint(Graphics g) {
        g.drawString("Dato", 50, 15);
        g.drawString("Frec", 100, 15);
        g.drawString("Barra", 150, 15);
```

```
        encontrarFrec(a, f);
        for (int i = 1, fila= 30; i < f.length; i++, fila+=15) {
            g.drawString(""+i, 50, fila);
            g.drawString(""+f[i]/2, 100, fila);
            g.drawString(cadena('x', f[i]/2), 150, fila);
        }
    }

    public void encontrarFrec(int a[], int b[]){
        for (int i = 0; i < a.length; i++) {
            f[a[i]]++;
        }
    }

    public String cadena(char ch, int n){
        String s1 = "";
        for (int i = 1; i <= n; i++) {
            s1 = s1 + ch;
        }
        return s1;
    }
}
```

Ejercicio 2:

Implementar una applet (sin interacción), para simular mil lanzamientos de un par de datos. a) b) 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), c) encontrar la frecuencia del total de la suma de ambos dados en los mil lanzamientos, mostrar cantidades presentadas para cada suma.

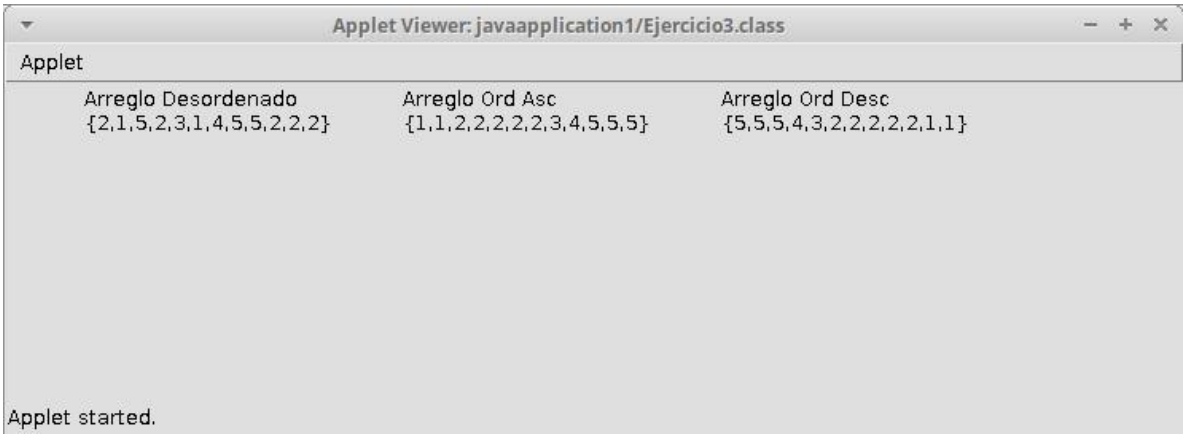



```java
package interaccion;
import java.applet.Applet;
import java.awt.Graphics;
import java.util.Random;
public class ejercicio2 extends Applet {

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

```
    @Override
    public void init() {
    }

    @Override
    public void paint(Graphics g){
        adjust();
        randomPool();
        g.drawString("Matriz de Contingencia", 10, 10);

        for (int i = 1; i < 7; i++) {
            g.drawString("\t"+i, i*dim, dim);
            g.drawString("\t"+i, dim, i*dim);
        }

        for (int i = 1; i < 7; i++) {
            for (int j = 1; j < 7; j++) {
                g.drawString(""+m[i][j], dim +i*dim, dim +j*dim);
            }
        }
        g.drawString("Frecuencia de ocurrencia", 10, 380);
        for (int i = 1; i < 13; i++) {
            g.drawString(""+f[i], dim*i, 400);
        }
    }

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

    public void randomPool(){
        Random random = new Random();
        for (int i = 0; i < 1000; i++) {
            int dice1 = random.nextInt(6) +1;
            int dice2 = random.nextInt(6) +1;

            f[dice1 + dice2]++;
            m[dice1][dice2]++;
        }
    }
}
```

Ejercicio 3:

Implementar una applet (sin interacción) para mostrar una arreglo de elementos y volver a mostrar el mismo arreglo ordenado en forma ascendente y ordenado en forma descendente.

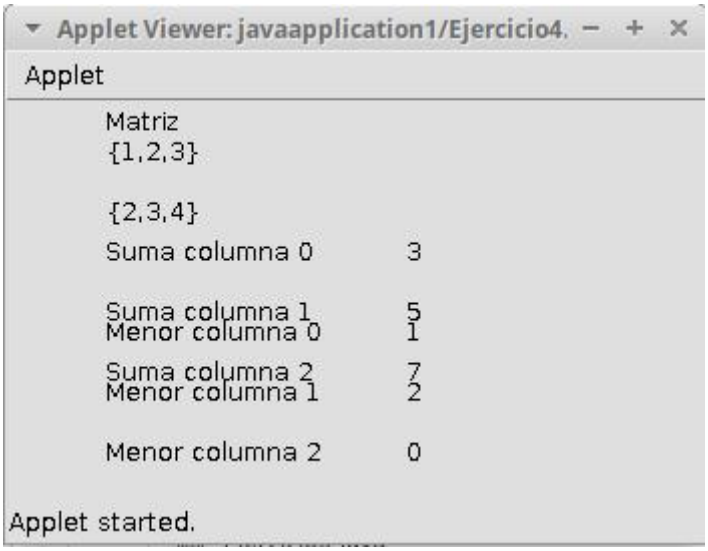



```
package interaccion;
import java.applet.Applet;
import java.awt.Graphics;
import java.util.Arrays;
public class ejercicio3 extends Applet {

    static int dim = 50;
    int a[] = { 2, 1, 5, 2, 3, 1, 4, 5, 5, 2, 2, 2 };

    @Override
    public void init() {
    }

    @Override
    public void paint(Graphics g){
        adjust();
        g.drawString("Arreglo Desordenado", 50, 15);
```

```
        g.drawString("Arreglo OrdenadoAsc", 250, 15);
        g.drawString("Arreglo OrdenadoDes", 450, 15);

        g.drawString(cadena(a), 50, 30);
        Arrays.sort(a);
        g.drawString(cadena(a), 250, 30);
        a = invertir(a);
        g.drawString(cadena(a), 450, 30);
    }

    public void adjust(){
        this.setSize(dim*15, dim*4);
    }

    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;
    }

    public String cadena(int a[]){
        return Arrays.toString(a);
    }
}
```

Ejercicio 4:

Implementar una applet (sin interacción) para mostrar una matriz rectangular y mostrar la suma de los elementos de las columnas y los elementos menores de cada columna.




```
package javaapplication1;
import java.awt.Graphics;
import java.util.Arrays;
public class Ejercicio4 extends java.applet.Applet {

    int a[][] = {
        {1, 2, 3},
        {2, 3, 4}
    };
    int f = 2; int c = 3;

    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
```

```
public void paint(Graphics g) {
    g.drawString("Matriz", 50, 15);
    for (int i = 0, fila = 30; i < f; i++, fila += 30) {
        g.drawString(cadena(a[i]), 50, fila);
    }
    int ss[] = sum(a);
    for (int i = 0, fila = 80; i < c; i++, fila += 30) {
        g.drawString("Suma columna " + i, 50, fila);
        g.drawString(String.valueOf(ss[i]), 200, fila);
    }
    int mcc[] = minc(a);
    for (int i = 0, fila = 120; i < c; i++, fila += 30) {
        g.drawString("Menor columna " + i, 50, fila);
        g.drawString(String.valueOf(mcc[i]), 200, fila);
    }

}

public String cadena(int a[]) {
    String s1 = "{";

    for (int i = 0; i < a.length - 1; i++) {
        s1 = s1 + a[i] + ",";
    }
    s1 = s1 + a[a.length - 1] + "}";
    return s1;
}

public static int[] sum(int[][] mat) {
    int[] myArray = new int[mat[0].length];

    for (int i = 0; i < mat[0].length; i++) {
        int suma = 0;
        for (int j = 0; j < mat.length; j++) {
            suma += mat[j][i];
        }
        myArray[i] = suma;

    }
    return myArray;
```

```
    }

    public int[] minc(int[][] mat) {
        int[] menorColumna = new int[c];
        for (int i = 0; i < f; i++) {
            menorColumna[i] = mat[i][0];
            for (int j = 0; j < c; j++) {
                if (mat[i][j] < menorColumna[i]) {
                    menorColumna[i] = mat[i][j];
                }
            }
        }
        return menorColumna;
    }

    private void initComponents() {

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

Ejercicio 5:

Implementar una applet (sin interacción) para mostrar todas las subcadenas de una cadena determinada. Es decir; mostrar la cadena, luego todas las subcadenas de la cadena inicial.

Applet

Cadena: ABC
Subcadenas:
1- A
2- B
3- C
4- AB
5- BC
6- ABC

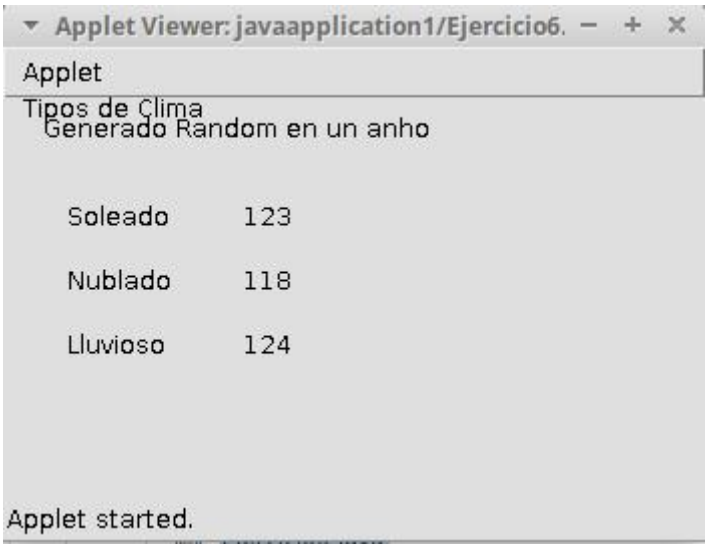
Applet iniciado.

```
package appletstest;
import java.awt.Graphics;import java.util.ArrayList;
public class Ejercicio2 extends java.applet.Applet {

    String s = "ABC";

    public static ArrayList<String> permutaciones = new
ArrayList<String>();

    public String swap(String a, int i, int j)
    {
        char temp;
        char[] charArray = a.toCharArray();
        temp = charArray[i] ;
        charArray[i] = charArray[j];
        charArray[j] = temp;
        return String.valueOf(charArray);
    }

   static void subString(char str[], int n) {
        // Pick starting point
        for (int len = 1; len <= n; len++) {
            // Pick ending point
```

```
            for (int i = 0; i <= n - len; i++) {
                //  Print characters from current
                // starting point to current ending
                // point.
                int j = i + len - 1;
                String x ="";
                for (int k = i; k <= j; k++) {
                    x += str[k];
                    System.out.print(str[k]);
                }
                permutaciones.add(x);
                //System.out.println();
            }
        }
    }

    public void init() {


        char str[] = new char[s.length()];

        for(int i=0;i<s.length();++i){
            str[i]=s.charAt(i);
        }

        subString(str, s.length());
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    @Override
    public void paint(Graphics g){
        g.drawString("Cadena: "+s, 16, 16);
        g.drawString("Subcadenas:", 16, 32);
```

```
        for(int i=0;i<permutaciones.size();++i){
            g.drawString(""+(i+1)+"- "+permutaciones.get(i),16 ,48+ i*20);
        }
    }

    private void initComponents() {

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

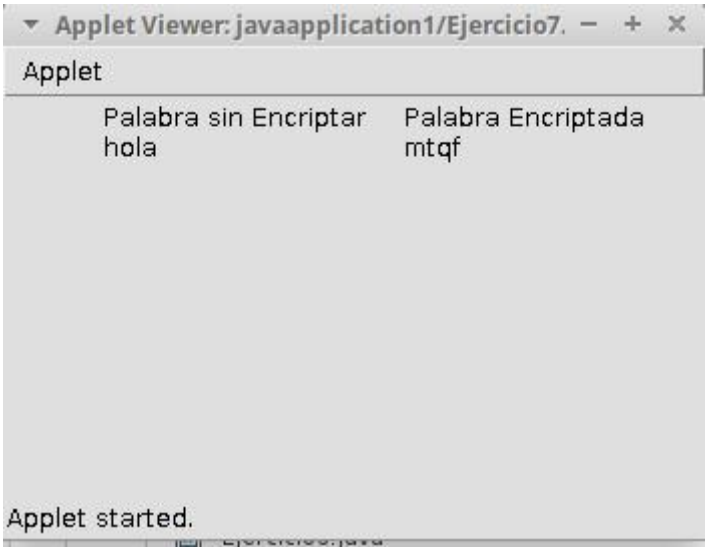
Ejercicio 6: Generación al azar de climas en un año




```
package javaapplication1;
import java.awt.Graphics;
import java.util.Random;

public class Ejercicio6 extends java.applet.Applet {
    static int generacion[] = new int[3];
    public void init() {
        randomGenerate();
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
```

```
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    static void randomGenerate() {
        Random random = new Random();
        for (int i = 0; i < 365; ++i) {
            int rand = random.nextInt(3);
            generacion[rand]++;
        }
    }
    @Override
    public void paint(Graphics g) {

        g.drawString("Tipos de Clima ", 10, 10);
        g.drawString("Generado Random en un anho", 20, 20);

        g.drawString("Soleado  ", 32, 64);
        g.drawString("" + generacion[0], 120, 64);
        g.drawString("Nublado  ", 32, 96);
        g.drawString("" + generacion[1], 120, 96);
        g.drawString("Lluvioso ", 32, 128);
        g.drawString("" + generacion[2], 120, 128);
    }
    private void initComponents() {

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

Ejercicio 7: Desencriptado de cadenas

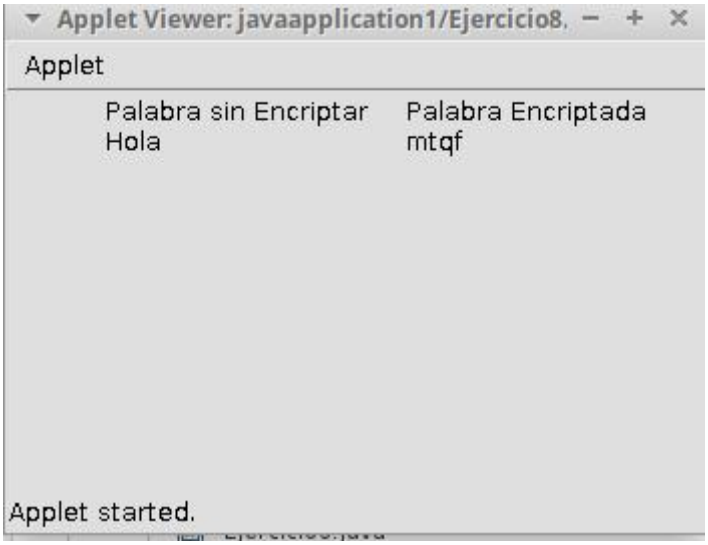



```
package javaapplication1;
import java.awt.Graphics;
import java.util.Random;
public class Ejercicio7 extends java.applet.Applet {
    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    @Override
    public void paint(Graphics g) {
        g.drawString("Palabra sin Encriptar", 50, 15);
        g.drawString("Palabra Encriptada", 200, 15);
```

```
        g.drawString("mtqf",200,30);
        g.drawString(decrypt("mtqf", 5),50,30);
    }
    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());
    }}

    private void initComponents() {

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

```
    }
}
```

Ejercicio 8: desenncriptado de cadenas



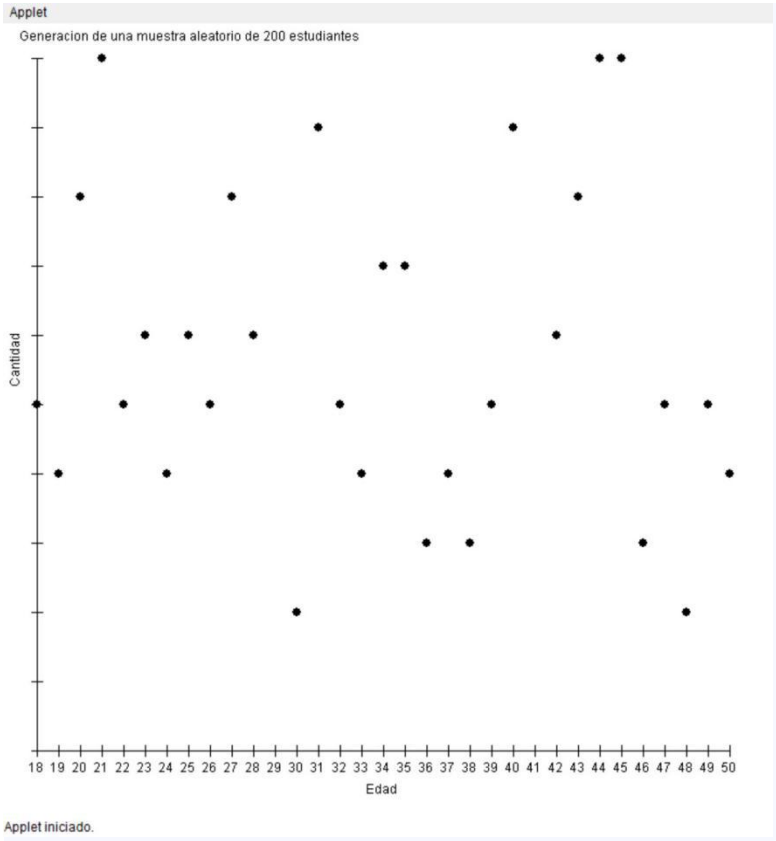

```
package javaapplication1;
import java.awt.Graphics;import java.util.Arrays;public class Ejercicio8
extends java.applet.Applet {
    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    @Override
    public void paint(Graphics g) {
        g.drawString("Palabra sin Encriptar", 50, 15);
        g.drawString("Palabra Encriptada", 200, 15);
        g.drawString("Hola",50,30);
        g.drawString(encrypt("hola", 5),200,30);
```

```
    }
    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());
    }}
.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());
    }
}
  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: Muestreo aleatorio en rango determinado



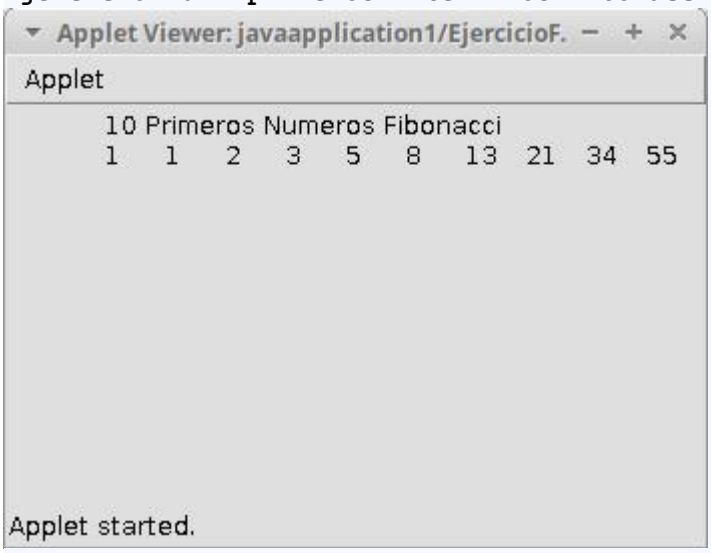

```
package appletstest;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import java.util.Random;
public class Ejercicio1 extends java.applet.Applet {
    public void init() {
        randomSeed();
```

```
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    int horizontalSize = 640;
    int verticalSize = 640;
    int xOffset=32,yOffset=32;
    int minAge = 18, maxAge =50;
    int ageRange = maxAge-minAge;
    int quantity = 10;
    int xSpace = horizontalSize/ageRange;
    int ySpace = verticalSize/quantity;
    int verticalUnit = verticalSize/quantity;



    int numberStudents = 200;

    int randomSample[] = new int[100];
    int width;

    public void randomSeed(){

        width = horizontalSize/32;

        Random random = new Random();
        for(int i=0;i<numberStudents;++i){
            int age = 18+random.nextInt(33);
            randomSample[age]++;
        }
    }

    public static void drawRotate(Graphics2D g2d, double x, double y, int
angle, String text) {
```

```
    g2d.translate((float)x,(float)y);
    g2d.rotate(Math.toRadians(angle));
    g2d.drawString(text,0,0);
    g2d.rotate(-Math.toRadians(angle));
    g2d.translate(-(float)x,-(float)y);}


    @Override
    public void paint(Graphics g){

        g.drawString("Generacion de una muestra aleatorio de
"+numberStudents + " estudiantes", 16, 16);

        g.drawLine(xOffset, yOffset, xOffset, yOffset+verticalSize);
        g.drawLine(xOffset, yOffset+verticalSize, xOffset+horizontalSize,
yOffset+verticalSize);

        for(int i=0;i<=horizontalSize/xSpace;++i){
            g.drawLine(xOffset+i*xSpace, yOffset+verticalSize+5,
xOffset+i*xSpace, yOffset+verticalSize-5);
        }

        for(int i=0;i<=verticalSize/ySpace;++i){
            g.drawLine(xOffset-5,
yOffset+i*ySpace ,xOffset+5,yOffset+i*ySpace );
        }

        for(int i=18,j=0;i<=50;++i,j++){
            g.fillOval(xOffset+j*xSpace-4, (yOffset+verticalSize)-
(randomSample[i]*verticalUnit)-4  , 8, 8);
            g.drawString(""+i, xOffset+j*xSpace-8,
yOffset+verticalSize+20);
        }

        g.drawString("Edad", (xOffset+horizontalSize)/2,
yOffset+verticalSize+40);

         Graphics2D g2 = (Graphics2D) g;
       drawRotate(g2, xOffset-16, (yOffset+verticalSize)/2, -90,
"Cantidad");
```

```
    }

    private void initComponents() {

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

Ejercicio 10: primeros n términos fibonacci




```
package javaapplication1;
import java.awt.Graphics;
public class EjercicioF extends java.applet.Applet {
    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    public void paint(Graphics g) {
        int numero = 10, fibo1, fibo2, i;
        g.drawString("10 Primeros Numeros Fibonacci", 50, 15);
        int filas = 50;
```

```
        fibo1 = 1;
        fibo2 = 1;

        g.drawString(String.valueOf(fibo1), filas, 30);
        filas += 30;
        for (i = 2; i <= numero; i++) {
            g.drawString(String.valueOf(fibo2), filas, 30);
            filas += 30;
            fibo2 = fibo1 + fibo2;
            fibo1 = fibo2 - fibo1;
        }

    }

    private void initComponents() {

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