Módulo: PROGRAMACIÓN EN LENGUAJES ESTRUCTURADOS
I.E.S.: AGUADULCE **C.F.G.S. :** DESARROLLO DE APLICACIONES INFORMÁTICAS
Fecha: **Hora de Comienzo:**
Examen: SEPTIEMBRE – PARTE PRÁCTICA **Duración:** 3 horas 30 minutos
Nombre: **D.N.I.:**

INSTRUCCIONES:

Estructura del examen de PLE Septiembre 2007.

- El primer y segundo ejercicio corresponden al primer cuatrimestre, el tercero y cuarto al segundo cuatrimestre. Cada cuatrimestre se considerará un examen distinto y se calificará sobre 10 puntos.
- Los ejercicios del primer cuatrimestre no es necesario que los realices en el entorno NetBeans, simplemente se pide que los hagas en papel, aunque si quieres o te resulta más fácil, puedes hacer uso de dicha herramienta.
- Se podrá hacer uso del material bibliográfico que se estime oportuno, así como de apuntes. No obstante, se advierte del peligro de pérdida de tiempo que conlleva ponerse a consultarlo durante el examen, pudiendo consumirse el tiempo disponible en la consulta, y quedándose sin tiempo para las respuestas.
- Aunque el alumnado puede traer al examen todo el código que crea necesario (en CD o en pen-drive USB), nosotros le suministraremos aquél que consideremos de utilidad para la realización de los ejercicios.
- La puntuación de cada apartado va al final del mismo.

EXAMEN PRIMER CUATRIMESTRE:

1.- Sabiendo que 0 es par, es decir,

$$esPar(0) = true$$

y que la paridad de cualquier otro entero positivo es la opuesta que la del entero anterior, desarrolla la función lógica esPar en Java, que nos permita averiguar de forma RECURSIVA, la paridad de un entero positivo. No puedes usar la función módulo. **(4 puntos)**

ejercicio1.java

```
package ejercicio1;

/**
 *
 * @author José Ramón Jiménez Reyes
 */
public class ejercicio1 {

    public static void main(String[] args){
        int numero = ES.leeNº("Introduce el número del que quieres saber su paridad: ", 0);
        if (esPar(numero))
            System.out.println("El número: " + numero + " es par");
        else
            System.out.println("El número: " + numero + " NO es par");
    }
    public static boolean esPar(int numero){
        if (numero == 0)
            return true;
        else
            return !esPar(numero - 1);
    }
}
```

2.- Diseñar una función RECURSIVA en Java para calcular multiplicaciones de enteros positivos según el siguiente método conocido como el del campesino egipcio: **(4 puntos)**

$$\text{mult}(x, y) = \begin{cases} 0 & \text{si } y=0 \\ x & \text{si } y=1 \\ \text{mult}(2 \cdot x, y \text{ div } 2) & \text{si } y \geq 2,\ y \bmod 2 = 0 \\ \text{mult}(2 \cdot x, y \text{ div } 2) + x & \text{si } y \geq 2,\ y \bmod 2 = 1 \end{cases}$$

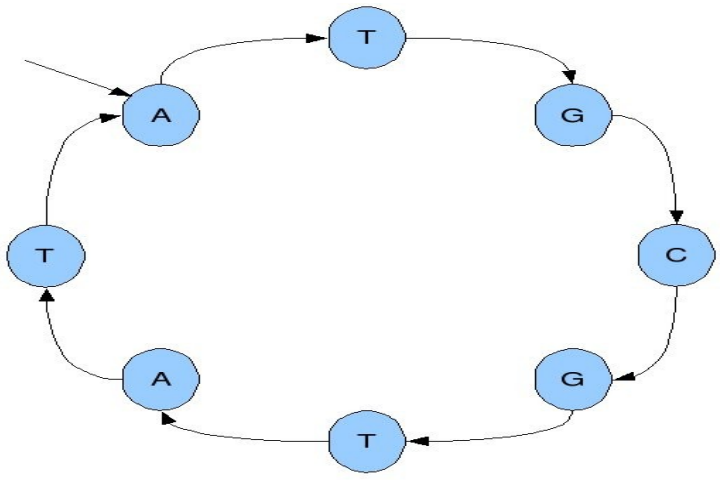
- Realiza la traza para el caso mult(3,5). **(2 puntos)**

ejercicio2.java

```
package ejercicio2;

/**
 *
 * @author José Ramón Jiménez Reyes
 */
public class ejercicio2 {

    public static void main(String[] args){
          System.out.println("Programa que multiplica números utilizando el método de los campesinos
egipcios");
        int x = ES.leeNº("Introduce el primer operando: ", 0);
        int y = ES.leeNº("Introduce el segundo opernado: ", 0);
        System.out.println("El resultado de la multiplicación es: " + mult(x, y));

    }
    public static int mult(int x, int y){
        if (y == 0) return 0;
        if (y == 1) return x;
        if ((y % 2) == 0) return mult(2*x, y / 2);
        else return mult(2*x, y / 2) + x;

    }
}
```

EXAMEN SEGUNDO CUATRIMESTRE:

3.- Una cadena de ADN se representa como una secuencia circular de bases (adenina, timina, citosina y guanina) que es única para cada ser vivo, por ejemplo:



Dicha cadena se puede representar como una lista circular que podría listarse recorriéndola en sentido horario desde su primer nodo como:

A T G C G T A T

Se pide, que partiendo del proyecto que te entregamos (ya utilizado en la teoría) GestionListaCircular:

- Modifiques la clase "Nodo" para reflejar que los datos de la lista serán caracteres indicando la base (A, T, G o C) que contiene cada nodo de la cadena de ADN. **(0.5 puntos)**
- Añadas un método constructor a la clase "ListaEnlazadaCircular" el cuál recibirá como parámetro un String indicando las bases que formarán nuestra cadena de ADN y cree la lista circular asociada. **(1 punto)**
- Debes crear una excepción que será lanzada en el método constructor cuando la cadena de caracteres que se le pase contenga algún carácter distinto a las bases posibles (A, T, G o C). **(1 punto)**

- Añadas otro método a la clase "ListaEnlazadaCircular" nombrado como "equals" que nos devuelva true si las dos cadenas de ADN coinciden y false en caso contrario. El método compararía la secuencia de ADN representada en la clase con otra secuencia de ADN que se pasa como argumento (**el argumento sería otra lista enlazada y NO un String**). **MUY IMPORTANTE**: La secuencia de ADN es cíclica, por lo que puede comenzar en cualquier posición. Por ejemplo, las dos secuencias siguientes coinciden, simplemente que una tendría su primer nodo en una posición distinta a la otra: **(2 puntos)**

  A T G C G T A T
  A T A T G C G T

- Sustituyas la clase "GestionListaCircular" por otra nombrada como "Ejercicio3" que se encargará de pedirnos las bases que compondrán nuestra primera cadena de ADN y las bases de la cadena que queremos comparar y nos dirá si dichas cadenas de ADN son iguales o no**. (0.5 puntos)**

Nodo.java

```
package ejercicio3;

/**
 * @author José Ramón Jiménez Reyes
 */

/**
 * Clase Nodo copiada de la documentación proporcionada para el examen, sustituyendo
 * el miembro nodo para que sea un char y eliminando los comentarios innecesarios y
 * dejando aquellos en los que hemos modificado algo.
 */
public class Nodo {
   /* Ahora el campo Dato es de tipo char*/
    public char dato;
    public Nodo siguienteNodo;

    // El constructor ahora recibe como parámetro un objeto Integer.
    public Nodo(char dato) {

        this.dato = dato;
        siguienteNodo = null;
    }

    public String toString(){
        String imprime= "" + this.dato;
        return imprime;
    }

    public boolean equals(Nodo nodo){
        boolean mismoDato=false;
        if (this.dato == nodo.dato){
          mismoDato=true;
        }
        return mismoDato;
    }
}
```

ListaEnlazadaCircular.java

```
package ejercicio3;

/**
 * @author José Ramón Jiménez Reyes
 */

class BaseNoValida extends Exception {}

public class ListaEnlazadaCircular
```

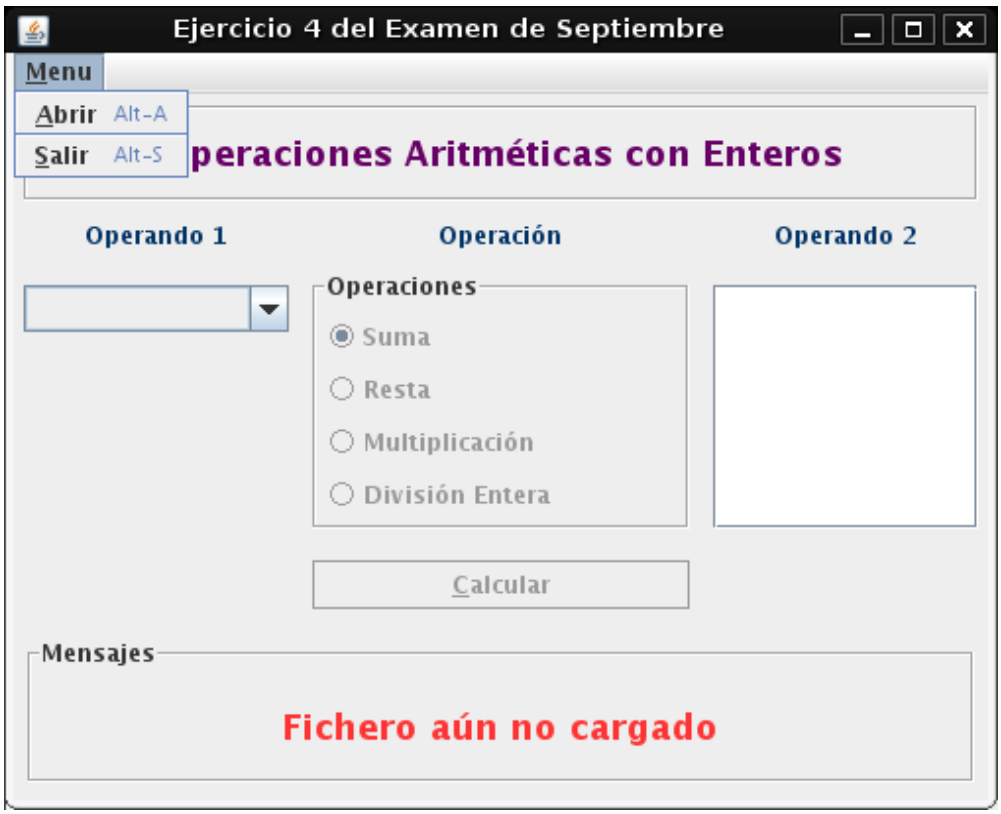
```
    public Nodo nodoEntradaActual=null;
    public int totalNodos=0;

    //Métodos necesarios para la realización del ejercicio copiados de la clase
    //ListaEnlazadaCircular proporcionada en el examen (los comentarios han sido omitidos)
    public ListaEnlazadaCircular() {
        nodoEntradaActual=null;
        totalNodos=0;
    }

    public ListaEnlazadaCircular insertarNodo(Nodo nodo){
        if (nodoEntradaActual==null){
            nodoEntradaActual=nodo;
            nodo.siguienteNodo=nodo;

        }else{
            Nodo nodoAux = nodoEntradaActual;
            while(nodoAux.siguienteNodo!=nodoEntradaActual){
                nodoAux=nodoAux.siguienteNodo;
            }
            nodo.siguienteNodo=nodoEntradaActual;
            nodoAux.siguienteNodo=nodo;

        }
        totalNodos++;
        return this;
    }

    public void listarNodos(String cabecera){
        if(nodoEntradaActual==null){
            System.out.println("La lista está vacía. No hay datos que listar");
        }else{
            Nodo nodoAux;
            System.out.println("");
            System.out.println(cabecera);
            for (int i =0;i<cabecera.length();i++){
                System.out.print("=");
            }
            System.out.println("");
            System.out.println(nodoEntradaActual);
            nodoAux=nodoEntradaActual.siguienteNodo;
            while (nodoAux!=nodoEntradaActual){
                System.out.println(nodoAux);
                nodoAux=nodoAux.siguienteNodo;
            }
        }
        System.out.println("");

    }

    public int consultaTotalNodos(){
        return totalNodos;
    }

    //Nuevos métodos implementados para la realización del ejercico del examen

    /**
     * Método constructor que recibe un String y crea una lista circular que
     * representa nuestra cadena de ADN
     */
    public ListaEnlazadaCircular(String cadenaBases) throws BaseNoValida {
        //Pasamos a mayúsculas la cadena que se pasa como parámetro
        cadenaBases = cadenaBases.toUpperCase();
        //Recorremos la cadena y si la base que representa es correcta la insertamos en la lista y en
caso contrario
        //Vacíamos nuestra lista y lanzamos una excepción, indicando que se ha cometido un error
        for (int i=0; i<cadenaBases.length(); i++){
            if (cadenaBases.charAt(i) != 'A' && cadenaBases.charAt(i) != 'T' && cadenaBases.charAt(i)
!= 'G' && cadenaBases.charAt(i) != 'C') {
                this.nodoEntradaActual = null;
```

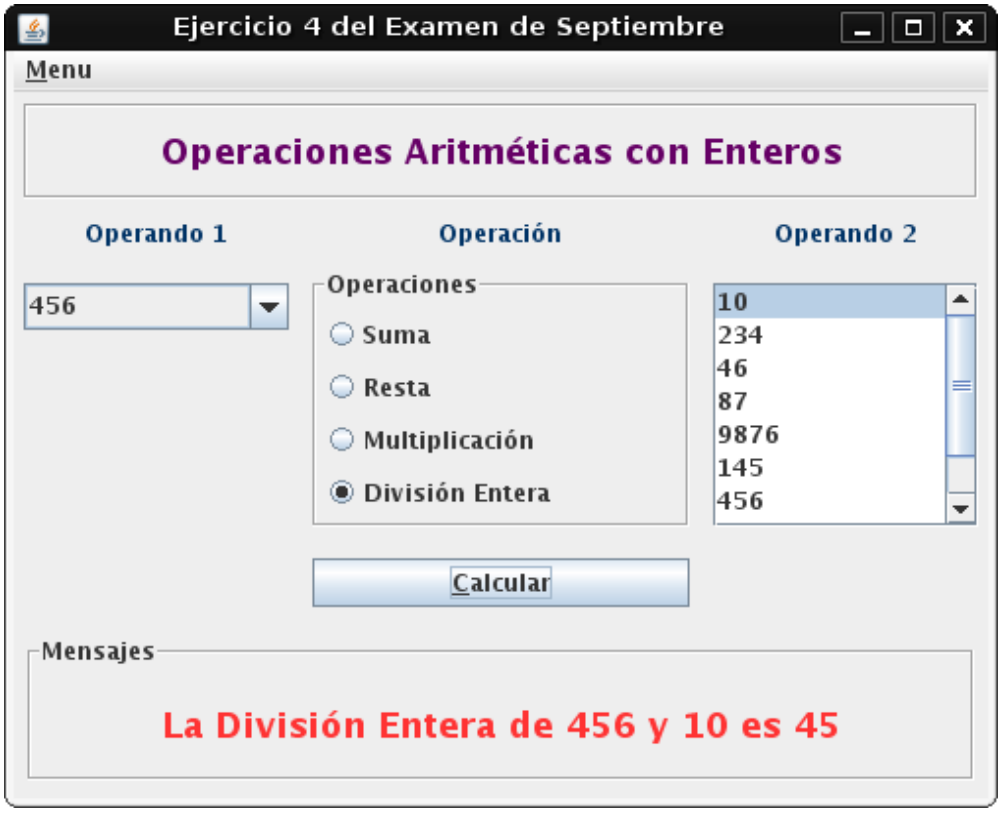
```
                this.totalNodos = 0;
                throw new BaseNoValida();
            }
            else
                this.insertarNodo(new Nodo(cadenaBases.charAt(i)));
        }
    }

    /**
     * Método que comprueba si la lista actual y la que se pasa por parámetro son iguales.
     * Para ello iremos rotando la segunda lista (avanzando desde su nodoEntradaActual) y en cada
     * rotación, comprobaremos si todos los datos de los nodos son iguales, recorriendo ambas listas
     * desde el principio.
     */
    public boolean equals(ListaEnlazadaCircular secuenciaADN) {
        //Nos curamos en salud y comprobamos si la lista a comparar es null (no está creada aún)
        if (secuenciaADN == null)
            return false;
        //Si ambas listas están vacías, las listas son iguales
        if (this.nodoEntradaActual == null && secuenciaADN.nodoEntradaActual == null)
            return true;
        //Si las listas no tienen el mismo número de nodos, estas no serán iguales
        if (this.totalNodos != secuenciaADN.totalNodos)
            return false;
        //Nodos auxiliares con los que recorreremos las listas
        Nodo nodoAux1 = null, nodoAux2 = null;
        //Definimos los nodos inciales de cada una de las listas circulares
        Nodo nodoInicial1 = this.nodoEntradaActual, nodoInicial2 = secuenciaADN.nodoEntradaActual;
         //Variable que nos permitirá salir del bucle cuando estemos seguros de que ambas listas son
iguales
        boolean iguales = false;
          //Recorremos la lista actual hasta que le demos la vuelta o estemos seguros de que ambas
listas son iguales
        do {
             //Empezamos con los nodos iniciales de cada lista (en el caso de la lista a comparar irá
avanzando en cada iteración)
            nodoAux1 = nodoInicial1;
            nodoAux2 = nodoInicial2;
              //Variable que nos permitirá salir del bulce cuando sepamos que en esta vuelta no son
iguales
            boolean seguir = true;
            //Recorremos la lista pasada por parámetro hasata que le demos la vuelta o sepamos que ya
no van a ser iguales
            do {
                 //Si uno de los datos de los nodos es diferente saldremos de este bucle para empezar
otra iteración
                  //del bucle exterior, para rotar la lista a comparar y empezar a comparar todas los
nodos
                if (nodoAux1.dato != nodoAux2.dato)
                    seguir = false;
                else {
                    nodoAux1 = nodoAux1.siguienteNodo;
                    nodoAux2 = nodoAux2.siguienteNodo;
                }
            } while (nodoAux2!=nodoInicial2 && seguir);
              //Si hemos llegado al final del bucle anterior (seguir es true) querrá decir que todos
los datos eran iguales
            if (seguir)
                iguales = true;
            //Si no avanzamos el nodo Inicial de la lista actual para seguir comparando
            else
                nodoInicial1 = nodoInicial1.siguienteNodo;
        } while (nodoInicial1!=this.nodoEntradaActual && !iguales);
        return iguales;
    }
}
```

Ejercicio3.java

```
package ejercicio3;

/**
 * @author José Ramón Jiménez Reyes
 */

public class Ejercicio3 {

    public Ejercicio3() {
    }

    public static void main(String[] args) {
        // Creamos las listas circulares
        ListaEnlazadaCircular secuenciaADN1 = null, secuenciaADN2 = null;
        boolean secuenciaValida = true;
        do {
            try {
                String cadena1 = ES.leeDeTeclado("Introduce la secuencia de bases para la primera
cadena de ADN (A, T, G, C): ");
                secuenciaADN1 = new ListaEnlazadaCircular(cadena1);
                secuenciaValida = true;
            } catch (BaseNoValida bnv) {
                System.out.println("ERROR: Recuerda que la cadena de ADN sólo puede contener las
bases A (Adenina), T (Timina), C (Citosina) y G (Guanina).");
                secuenciaValida = false;
            }
        } while (!secuenciaValida);

        do {
            try {
                String cadena2 = ES.leeDeTeclado("Introduce la secuencia de bases para la segunda
cadena de ADN (A, T, G, C): ");
                secuenciaADN2 = new ListaEnlazadaCircular(cadena2);
                secuenciaValida = true;
            } catch (BaseNoValida bnv) {
                System.out.println("ERROR: Recuerda que la cadena de ADN sólo puede contener las
bases A (Adenina), T (Timina), C (Citosina) y G (Guanina).");
                secuenciaValida = false;
            }
        } while (!secuenciaValida);

        if (secuenciaADN1.equals(secuenciaADN2))
            System.out.println("Cadenas de ADN IGUALES");
        else
            System.out.println("Cadenas de ADN DIFERENTES");

    }
}
```

4.- Se quiere dotar de un interfaz gráfico a una aplicación bastante sencilla, cuyo único cometido es realizar operaciones aritméticas con números enteros. En un fichero de texto tenemos almacenados en cada línea un número que consideraremos como un operando. La interfaz tendrá el siguiente aspecto:


Ejercicio 4 del Examen de Septiembre
Menu
Abrir Alt-A
Salir Alt-S
peraciones Aritméticas con Enteros
Operando 1
Operación
Operando 2
Operaciones
Suma
Resta
Multiplicación
División Entera
Calcular
Mensajes
Fichero aún no cargado


Ejercicio 4 del Examen de Septiembre
Menu
Operaciones Aritméticas con Enteros
Operando 1
Operación
Operando 2
456
Operaciones
Suma
Resta
Multiplicación
División Entera
10
234
46
87
9876
145
456
Calcular
Mensajes
La División Entera de 456 y 10 es 45

Se pide lo siguiente:

- Diseñes dicha interfaz utilizando para ello un Layaout null. **(0.5 puntos)**
- En el menú debe haber una opción nombrada como "Abrir" que nos permitirá abrir un fichero, leerlo comprobando que dicho fichero está bien formado, es decir, en cada línea hay un número entero, indicando el resultado de la operación en el panel de mensajes (todo bien, fichero mal formado, ...). Cuando el fichero se lee correctamente, el botón, los radio botones, el combo y la lista se habilitarán. Podremos abrir tantos ficheros como queramos pero siempre que abramos uno perderemos los contenidos del anterior, y si ha habido algún error también. **(1.5 puntos)**
- El resultado de la lectura correcta del fichero deberá rellenar el modelo utilizado por el combo y la lista. (Si por nervios o tiempo no eres capaz de rellenar los modelos con los datos del fichero, te recomendamos que rellenes a mano los modelos con datos inventados para poder continuar con el ejercicio, con su consiguiente penalización). **(1.5 puntos)**
- El botón "Calcular" nos mostrará en el panel "Mensajes" el resultado de realizar la operación seleccionada en los radio botones entre el operando 1 y el operando 2. Si no tenemos seleccionado alguno de los operandos, se nos deberá informar del error en el panel de mensajes. **(1 punto)**
- El menú tendrá una entrada "Abrir", un separador y otra entrada "Salir". De la primera ya hemos hablado. Y la última como su nombre indica nos permitirá abandonar la aplicación. Debes dotar a los menús de nemónicos y aceleradores. **(0.5 puntos)**

jF_Ejercicio4.java

```
package ejercicio4;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.Vector;
import javax.swing.JFileChooser;

/**
 *
 * @author  José Ramón Jiménez Reyes
 */
public class jF_Ejercicio4 extends javax.swing.JFrame {

    /** Creates new form jF_Ejercicio4 */
    public jF_Ejercicio4() {
        initComponents();
        modeloLista.add("      ");
        jL_Operando.setListData(modeloLista);
        cambiaEstado(false);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        bG_Operaciones = new javax.swing.ButtonGroup();
        jCB_Operando = new javax.swing.JComboBox();
        jP_Operaciones = new javax.swing.JPanel();
        jR_Suma = new javax.swing.JRadioButton();
        jR_Resta = new javax.swing.JRadioButton();
        jR_Multiplicacion = new javax.swing.JRadioButton();
```

```
        jR_Division = new javax.swing.JRadioButton();
        jSP_Operando = new javax.swing.JScrollPane();
        jL_Operando = new javax.swing.JList();
        jL_Info = new javax.swing.JLabel();
        jL_Mensajes = new javax.swing.JLabel();
        jB_Calcular = new javax.swing.JButton();
        jL_Operando1 = new javax.swing.JLabel();
        jL_Operando2 = new javax.swing.JLabel();
        jL_Operacion = new javax.swing.JLabel();
        jMB_Principal = new javax.swing.JMenuBar();
        jM_Fichero = new javax.swing.JMenu();
        jM_Abrir = new javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        jM_Salir = new javax.swing.JMenuItem();

        getContentPane().setLayout(new java.awt.GridBagLayout());

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Ejercicio 4 del Examen de Septiembre");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(13, 5, 5, 5);
        getContentPane().add(jCB_Operando, gridBagConstraints);

        jP_Operaciones.setLayout(new java.awt.GridBagLayout());

        jP_Operaciones.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFacto
ry.createEtchedBorder(), "Operaciones"));
        bG_Operaciones.add(jR_Suma);
        jR_Suma.setSelected(true);
        jR_Suma.setText("Suma");
        jR_Suma.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        jR_Suma.setMargin(new java.awt.Insets(0, 0, 0, 0));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        jP_Operaciones.add(jR_Suma, gridBagConstraints);

        bG_Operaciones.add(jR_Resta);
        jR_Resta.setText("Resta");
        jR_Resta.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        jR_Resta.setMargin(new java.awt.Insets(0, 0, 0, 0));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        jP_Operaciones.add(jR_Resta, gridBagConstraints);

        bG_Operaciones.add(jR_Multiplicacion);
        jR_Multiplicacion.setText("Multiplicaci\u00f3n");
        jR_Multiplicacion.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        jR_Multiplicacion.setMargin(new java.awt.Insets(0, 0, 0, 0));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
```

```
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        jP_Operaciones.add(jR_Multiplicacion, gridBagConstraints);

        bG_Operaciones.add(jR_Division);
        jR_Division.setText("Divisi\u00f3n Entera");
        jR_Division.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        jR_Division.setMargin(new java.awt.Insets(0, 0, 0, 0));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        jP_Operaciones.add(jR_Division, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        getContentPane().add(jP_Operaciones, gridBagConstraints);

        jL_Operando.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        jSP_Operando.setViewportView(jL_Operando);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(13, 5, 7, 5);
        getContentPane().add(jSP_Operando, gridBagConstraints);

        jL_Info.setFont(new java.awt.Font("Dialog", 1, 18));
        jL_Info.setForeground(new java.awt.Color(102, 0, 102));
        jL_Info.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jL_Info.setText("Operaciones Aritm\u00e9ticas con Enteros");
        jL_Info.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.ipady = 15;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        getContentPane().add(jL_Info, gridBagConstraints);

        jL_Mensajes.setFont(new java.awt.Font("Dialog", 1, 18));
        jL_Mensajes.setForeground(new java.awt.Color(255, 51, 51));
        jL_Mensajes.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jL_Mensajes.setText("Fichero a\u00fan no cargado");
        jL_Mensajes.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.
createEtchedBorder(), "Mensajes"));
        jL_Mensajes.setPreferredSize(new java.awt.Dimension(157, 70));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.gridwidth = 3;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.ipady = 20;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        getContentPane().add(jL_Mensajes, gridBagConstraints);
```

```
        jB_Calcular.setMnemonic('C');
        jB_Calcular.setText("Calcular");
        jB_Calcular.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jB_CalcularActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
        gridBagConstraints.insets = new java.awt.Insets(5, 7, 5, 7);
        getContentPane().add(jB_Calcular, gridBagConstraints);

        jL_Operando1.setForeground(new java.awt.Color(0, 51, 102));
        jL_Operando1.setText("Operando 1");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        getContentPane().add(jL_Operando1, gridBagConstraints);

        jL_Operando2.setForeground(new java.awt.Color(0, 51, 102));
        jL_Operando2.setText("Operando 2");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        getContentPane().add(jL_Operando2, gridBagConstraints);

        jL_Operacion.setForeground(new java.awt.Color(0, 51, 102));
        jL_Operacion.setText("Operaci\u00f3n");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.insets = new java.awt.Insets(5, 5, 5, 5);
        getContentPane().add(jL_Operacion, gridBagConstraints);

        jM_Fichero.setMnemonic('M');
        jM_Fichero.setText("Menu");
             jM_Abrir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_A,
java.awt.event.InputEvent.ALT_MASK));
        jM_Abrir.setMnemonic('A');
        jM_Abrir.setText("Abrir");
        jM_Abrir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jM_AbrirActionPerformed(evt);
            }
        });

        jM_Fichero.add(jM_Abrir);

        jM_Fichero.add(jSeparator1);

             jM_Salir.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_S,
java.awt.event.InputEvent.ALT_MASK));
        jM_Salir.setMnemonic('S');
        jM_Salir.setText("Salir");
        jM_Salir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jM_SalirActionPerformed(evt);
            }
        });

        jM_Fichero.add(jM_Salir);
```

```
        jMB_Principal.add(jM_Fichero);

        setJMenuBar(jMB_Principal);

        pack();
    }// </editor-fold>

    private void jB_CalcularActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        if (jCB_Operando.getSelectedIndex() == -1)
            jL_Mensajes.setText("Debes seleccionar el Operando 1");
        else if (jL_Operando.getSelectedIndex() == -1)
            jL_Mensajes.setText("Debes seleccionar el Operando 2");
        else {
                                            String    operando1=(String)jCB_Operando.getSelectedItem(),
operando2=(String)jL_Operando.getSelectedValue();
            int opInt1=0, opInt2=0;
            try {
                opInt1 = Integer.parseInt(operando1);
                opInt2 = Integer.parseInt(operando2);
            } catch (NumberFormatException nfe) {}
            if (jR_Suma.isSelected())
                 jL_Mensajes.setText("La Suma de " + operando1 + " y " + operando2 + " es " +
(opInt1+opInt2));
            else if (jR_Resta.isSelected())
                jL_Mensajes.setText("La Resta de " + operando1 + " y " + operando2 + " es " +
(opInt1-opInt2));
            else if (jR_Multiplicacion.isSelected())
                 jL_Mensajes.setText("La Multiplicación de " + operando1 + " y " + operando2 + " es "
+ (opInt1*opInt2));
            else
                jL_Mensajes.setText("La División Entera de " + operando1 + " y " + operando2 + " es "
+ (opInt1/opInt2));

        }

    }

    private void jM_SalirActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        System.exit(0);
    }

    /**
     * Método que cambia de Estado nuestros componentes.
     */
    private void cambiaEstado(boolean estado) {
        jB_Calcular.setEnabled(estado);
        jR_Suma.setEnabled(estado);
        jR_Resta.setEnabled(estado);
        jR_Multiplicacion.setEnabled(estado);
        jR_Division.setEnabled(estado);
    }

    private void jM_AbrirActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        JFileChooser jFC_Abrir = new JFileChooser();
        int seleccion = jFC_Abrir.showOpenDialog(this);
        if (seleccion == jFC_Abrir.APPROVE_OPTION) {
            File fichero = jFC_Abrir.getSelectedFile();
            modeloLista.clear();
            jCB_Operando.removeAllItems();
            boolean error = false;
            try {
                BufferedReader in = new BufferedReader(new FileReader(fichero));
                String operando;
                while ((operando = in.readLine()) != null && !error) {
                    //Si el operando no es entero el fichero está mal formado
                    try {
                        int opInt = Integer.parseInt(operando);
```

```
                    } catch (NumberFormatException e) {
                        jL_Mensajes.setText("Fichero Mal Formado");
                        error = true;
                    }
                    modeloLista.add(operando);
                    jCB_Operando.addItem(operando);
                }
                in.close();
                jL_Operando.setListData(modeloLista);
                jCB_Operando.setSelectedIndex(-1);
            } catch (IOException ioe) {
                jL_Mensajes.setText("Error en la lectura del Fichero");
                error = true;
            }
            if (error) {
                modeloLista.clear();
                modeloLista.add("      ");
                jCB_Operando.removeAllItems();
            }
            else {
                jL_Mensajes.setText("Fichero leído correctamente");
                cambiaEstado(true);
            }
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new jF_Ejercicio4().setVisible(true);
            }
        });
    }

    private Vector<String> modeloLista = new Vector();
    // Variables declaration - do not modify
    private javax.swing.ButtonGroup bG_Operaciones;
    private javax.swing.JButton jB_Calcular;
    private javax.swing.JComboBox jCB_Operando;
    private javax.swing.JLabel jL_Info;
    private javax.swing.JLabel jL_Mensajes;
    private javax.swing.JLabel jL_Operacion;
    private javax.swing.JList jL_Operando;
    private javax.swing.JLabel jL_Operando1;
    private javax.swing.JLabel jL_Operando2;
    private javax.swing.JMenuBar jMB_Principal;
    private javax.swing.JMenuItem jM_Abrir;
    private javax.swing.JMenu jM_Fichero;
    private javax.swing.JMenuItem jM_Salir;
    private javax.swing.JPanel jP_Operaciones;
    private javax.swing.JRadioButton jR_Division;
    private javax.swing.JRadioButton jR_Multiplicacion;
    private javax.swing.JRadioButton jR_Resta;
    private javax.swing.JRadioButton jR_Suma;
    private javax.swing.JScrollPane jSP_Operando;
    private javax.swing.JSeparator jSeparator1;
    // End of variables declaration

}
```