

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1

CATEDRÁTICO: ING. WILLIAM ESTUARDO ESCOBAR ARGUETA

TUTOR ACADÉMICO: JOSUÉ RODOLFO MORALES CASTILLO



MANUAL TÉCNICO

EDUARDO SEBASTIAN GUTIERREZ FELIPE

CARNÉ: 202300694

SECCIÓN: B

GUATEMALA, 4 DE SEPTIEMBRE DEL 2,024

ÍNDICE

ÍNDICE	1
INTRODUCCIÓN	2
OBJETIVOS	2
1. GENERAL	2
2. ESPECÍFICOS	2
ALCANCES DEL SISTEMA	3
ESPECIFICACIÓN TÉCNICA	4
• REQUISITOS DE HARDWARE	4
• REQUISITOS DE SOFTWARE	4
DESCRIPCIÓN DE LA SOLUCIÓN	5
LÓGICA DEL PROGRAMA	6

INTRODUCCIÓN

El objetivo de este manual es brindarle al usuario una documentación con información técnica acerca del funcionamiento del programa IPC Quimik, un programa para la gestión y análisis de muestras químicas de laboratorio. Con este manual, el usuario entenderá la lógica detrás del programa y cómo los elementos que lo componen interactúan entre sí para crear un programa consistente y robusto.

OBJETIVOS

1. GENERAL

- 1.1. Brindar al usuario información técnica sobre la lógica que compone el programa

2. ESPECÍFICOS

- 2.1. Describir las clases, métodos y variables más importantes del programa
- 2.2. Indicar las librerías utilizadas para realizar el programa

ALCANCES DEL SISTEMA

Con este manual, el usuario será capaz de entender la lógica que hay detrás del programa que utilizará, aún si no conoce los fundamentos de programación orientada a objetos. Por medio de este manual el usuario conocerá las clases, métodos, funciones, procedimientos y variables declaradas en su programa y la forma en que éstas interactúan entre sí para darle la funcionalidad y robustez al programa. Así mismo, se especificarán los tipos de archivos que el usuario puede cargar en el sistema y el formato que estos deben tener para que el programa funcione de manera correcta. Por último, se le hará saber al usuario qué requisitos debe cumplir su dispositivo para ejecutar el programa de forma eficiente.

ESPECIFICACIÓN TÉCNICA

● REQUISITOS DE HARDWARE

- Configuraciones mínimas de hardware:
 - Procesador: Intel Pentium III a 800 MHz o equivalente
 - Memoria: 512 MB
 - Espacio en disco: 1 GB de espacio libre en disco
- Configuración recomendada de hardware:
 - Procesador: Intel Core i5 9ª generación o posterior
 - Memoria: 4GB o superior
 - Espacio en disco: 2 GB de espacio libre en disco o más

● REQUISITOS DE SOFTWARE

- Requisitos mínimos de software:
 - Windows 7 Professional
 - Ubuntu 9.1
- Requisitos recomendados de software
 - Windows 10 Home o posteriores
 - Ubuntu 15.04 o posteriores

DESCRIPCIÓN DE LA SOLUCIÓN

- La solución propuesta se basa en la serialización de los datos dentro del programa. Se pueden identificar 3 objetos fundamentales en este programa, los Investigadores, las Muestras y los Patrones. Cada uno de estos objetos tiene distintos atributos, por lo que es ideal crear un “molde” (clase) para cada uno de estos objetos y luego serializarlos, es decir, almacenarlos en archivos binarios (extensión .bin) para lograr la persistencia de éstos. Así mismo, fue necesario crear varias interfaces que manipularían los mismos datos, por lo que se crearon tres listas dinámicas estáticas para que pudieran ser accedidas y manipuladas desde cualquier clase. Para el análisis de muestras, se utilizó un algoritmo que convierte el patrón de cualquier muestra en un patrón de 1´s y 0´s para posteriormente compararlo con los patrones almacenados en el sistema. El resultado se muestra en la tabla de resultados dentro de la aplicación a la vez que se despliega un reporte html en el navegador o en cualquier otro visor de este tipo de archivos.

LÓGICA DEL PROGRAMA

❖ Login

```
import java.io.IOException;
import javax.swing.JOptionPane;
```

➤ Librerías

La clase java.io.IOException es utilizada para capturar la excepción al leer el archivo binario.

La clase javax.swing.JOptionPane es utilizada para mostrar ventanas emergentes, principalmente para mostrar mensajes de error.

➤ Variables Globales de la clase _(Login)

```
public class Login extends javax.swing.JFrame {

    //Variables que almacenarán los datos del JTextField
    private String user = "";
    private String password = "";
    public static int indexInvestigador;
```

Variables de tipo String para almacenar el código y contraseña ingresados por el usuario, si estas coinciden con los almacenados en el sistema, se guarda el índice de la posición del investigador en el ArrayList para mostrar sus datos en el módulo de Investigador.

➤ Función Main

En la función Main se leen los archivos binarios que almacenan datos de Investigadores, Muestras y Patrones, además se muestra el JFrame Login();

```

public static void main(String args[]) throws IOException, ClassNotFoundException {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    //Al iniciar el programa, se leen los objetos de tipo Investigador del archivo binario y se agreguen al arraylist
    Escribir_InvestigadorBinario lectorInvestigador = new Escribir_InvestigadorBinario();
    lectorInvestigador.leer_investigadorbin();

    //Al iniciar el programa, se leen los objetos de tipo Muestra del archivo binario y se agreguen al arraylist
    Escribir_muestraBinaria lectorMuestra = new Escribir_muestraBinaria();
    lectorMuestra.leer_muestrabin();

    Escribir_patronBinario lectorPatron = new Escribir_patronBinario();
    lectorPatron.leer_patronbin();

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Login().setVisible(true);
        }
    });
}



```

➤ Procedimientos, métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

-  `public Login()`

Constructor que carga todos los atributos del JFrame Login.

-   `private void label_botonMouseClicked(java.awt.event.MouseEvent evt) {`

Método que captura el evento al hacer clic sobre el botón de iniciar sesión

❖ Administrador

```
import java.awt.Desktop;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.UIManager;
import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

➤ Librerías

La librería javax.swing contiene los elementos de la interfaz gráfica.

La librería java.awt.Desktop sirve para abrir documentos con una ruta especificada, en este caso se usó para abrir archivos html.

La librería java.io.BufferedWriter sirve para escribir en archivos de texto, utilizada para escribir un archivo html.

La librería java.io.File, usada para crear objetos de tipo File.

La librería java.io.FileWriter es usada para usarla en conjunto con un objeto de tipo BufferedWriter.

La librería java.io.IOException es usada para capturar la excepción al leer o escribir un archivo.

➤ Variables Globales de la clase _(Administrador)

```
//Se crea un dtm estático para agregar y actualizar datos de la tabla
public static DefaultTableModel dtml = new DefaultTableModel(); //default table model para investigadores
public static DefaultTableModel dtm_muestras; //default table model para muestras
public static DefaultTableModel dtm_patrones; //default table model para patrones

//Creación del botón VER de la tabla de muestras
public static JButton ver = new JButton("VER");

//Se crean dos enteros para almacenar la fila y columna de la tabla de muestras que seleccionó el usuario
public static int fila, columna;
```

Variables de tipo DefaultTableModel que corresponden a las tablas del módulo de Administrador. También se declaró un JButton que va en las tablas de Muestras y Patrones para ver el html con el patrón


ingresado. Las variables enteras son utilizadas para obtener las coordenadas x, y cuando el usuario hace clic en las tablas de muestras o patrones.

➤ **Función Main**



Esta clase no tiene un main. El constructor de esta clase se invoca desde el Login, cuando el usuario ingresa “admin” como usuario y contraseña.

➤ **Procedimientos, métodos y Funciones utilizadas**

A continuación, se dará una explicación general de lo que hace cada función:



-  `45 public Administrador()`
Constructor que contiene los elementos de esta interfaz.
-  `109 public void tabla_investigadores()`
Procedimiento para crear la tabla de investigadores.
-  `124 public void tabla_muestras()`
Procedimiento para crear la tabla de muestras.
-  `133 public void tabla_patrones()`
Procedimiento para crear la tabla de patrones.
-   `563 //Método que captura el evento del bot
private void crear_btnActionPerformed`

Procedimiento para mostrar el formulario de crear investigador.

-  



```
568 //Método que captura el evento del botón ac
private void actualizar_btnActionPerformed
```

Procedimiento para mostrar el formulario de actualizar Investigador.

-  



```
573 //Método que captura el evento del botón
private void eliminar_btnActionPerformed
```

Procedimiento para mostrar el formulario de eliminar Investigador.

-  



```
577 private void crear_muestra_btnActionPerformed
```

Procedimiento para mostrar el formulario para crear una muestra.

-  


```
586 //Método para capturar el evento del bo
private void asignar_btnActionPerformed
```

Procedimiento para mostrar una muestra a un investigador.

-  



```
653 //Método que captura el evento del botó
private void cargar_btnActionPerformed(
```

Procedimiento para cargar investigadores al sistema.

- 



```
687 private void tabla_muestrasMouseClicked
```

Procedimiento para capturar las coordenadas del clic del mouse sobre las tablas de muestras y patrones.

-  

```
741 private void crear_patron_btnActionPerformed
```

Procedimiento para mostrar el formulario para crear un patrón.

-  

```
745 private void eliminar_patron_btnActionPerformed
```


Procedimiento para mostrar el formulario para eliminar un patrón.

- 750 `private void cerrarS_1ActionPerformed(`


Procedimiento para el botón de cerrar sesión

- 791 `private void cargar_muestra_btnActionPerformed(`

- Procedimiento para cargar muestras en el sistema

- 803  `private void cargar_patron_btnActionPerformed(`

Procedimiento para cargar patrones en el sistema

- 817  `private void tabla_patronesMouseClicked(`

Procedimiento para capturar las coordenadas del clic del mouse en las tablas, si se presionó el botón se muestra el html con el reporte.

❖ Investigador

```
import java.awt.Desktop;  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Arrays;  
import java.util.Date;  
import javax.swing.JButton;  
import javax.swing.JOptionPane;  
import javax.swing.UIManager;  
import javax.swing.table.DefaultTableModel;
```

➤ Librerías

La librería `java.awt.Desktop` se utiliza para abrir los archivos html.

La librería `java.io.File` se utiliza para crear objetos de tipo `File` y poder manipularlos como archivos.

La librería `java.io` es utilizada también para el manejo de excepciones, cuando ocurre un comportamiento inesperado en el programa.

La librería `java.text` fue utilizada para obtener la fecha en la que se crean los reportes.

La librería `java.util.Arrays` sirve para comparar si dos matrices son iguales.

La librería `java.util.Date` también es utilizada para el manejo de fechas.

La librería `javax.swing` se utilizó para importar los elementos de la interfaz gráfica del programa.

➤ Variables Globales de la clase _(Investigador)

```
int indexInvestigador = Login.indexInvestigador;;  
public static DefaultTableModel dtm_resultados;  
  
public static JButton ver = new JButton("VER");  
  
public static int fila, columna;
```

La variable entera indexInvestigador es utilizada para guardar la posición del investigador dentro de la lista dinámica, para poder obtener sus datos.


Se crea una variable de tipo DefaultTableModel para la tabla de resultados, un botón “ver” para mostrar los html en la tabla, y dos variables enteras para almacenar las coordenadas x,y del clic del mouse en la tabla.

➤ Función Main


Esta clase no contiene una función main, su constructor es invocado desde la clase Login, al ingresar un código y contraseña válidos.

➤ Procedimientos, métodos y Funciones utilizadas



A continuación, se dará una explicación general de lo que hace cada función:

-  `public void tabla_resultados()`


Procedimiento para crear la tabla de resultados.

-  `private void cerrarS_btnActionPerformed()`

Procedimiento utilizado para darle funcionalidad al botón de cerrar sesión.

-  279  `private void analizar_btnActionPerformed`

Procedimiento para darle funcionalidad al botón de Analizar muestras

- 453  `private void tabla_resultadosMouseClicked`

Procedimiento para mostrar el html al dar clic en el botón “Ver” de la tabla de resultados.

❖ Crear_Investigador

```
import javax.swing.JOptionPane;
```

➤ Librerías

La librería `javax.swing.JOptionPane` sirve para mostrar ventanas emergentes, con mensajes de alerta o de error.

➤ Variables Globales de la clase `_(Crear_Investigador)`



Sin variables globales.

➤ Función Main

Esta clase no contiene una función main, su constructor es invocado desde la clase `Administrador`.


➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  

```
129 //Método que captura el evento del bot  
private void crear_btnActionPerformed(
```

Procedimiento utilizado para crear (ingresar) un investigador en el sistema

- 

```
192 private void field_generoKeyTyped(
```

Procedimiento para limitar el ingreso de caracteres en el `JTextField` a uno, únicamente los caracteres F o M.

❖ Actualizar_Investigador

```
- import javax.swing.JOptionPane;
```

➤ Librerías

La librería `javax.swing.JOptionPane` es utilizada para mostrar ventanas emergentes con mensajes de error o de alerta.

➤ Variables Globales de la clase `_Actualizar_Investigador`


Sin variables globales.

➤ Función Main

Esta clase no contiene una función main, esta clase es invocada desde la clase `Administrador` a través de su correspondiente constructor.


➤ Procedimientos, métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

-  `private void buscar_btnActionPerformed`

Procedimiento para buscar por medio del código, los datos del investigador

-

-  `private void actualizar_btnActionPerformed`

Procedimiento para actualizar los datos de un investigador

❖ Eliminar_Investigador

```
import java.util.ArrayList;  
import java.util.List;  
import javax.swing.JOptionPane;
```

➤ Librerías

Las librerías `java.util.ArrayList` y `java.util.List` son las librerías que permiten el manejo de las listas dinámicas.

La librería `javax.swing.JOptionPane` permite mostrar ventanas emergentes con mensajes de error o de alerta.

➤ Variables Globales de la clase `_Eliminar_Investigador`


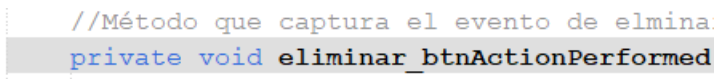
Sin variables globales.

➤ Función Main

Esta clase no contiene una función main, esta clase es invocada desde la clase `Administrador` a través de su correspondiente constructor.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

-  

Procedimiento para eliminar un investigador del sistema.

❖ Crear_muestra

```
import java.io.IOException;  
import javax.swing.JOptionPane;
```

➤ Librerías

La librería `java.io.IOException` es utilizada para capturar la excepción al leer o escribir en el archivo binario.

La librería `javax.swing.JOptionPane` sirve para mostrar ventanas emergentes.

➤ Variables Globales de la clase `_(Crear_muestra)`

```
int[][] patron_muestra = null;
```


Variable global de tipo matriz de enteros, utilizada para guardar temporalmente la matriz cargada desde el csv.

➤ Función Main


Esta clase no contiene una función main, es invocada desde la clase de `Crear_muestra`.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

-  `private void crear_btnActionPerformed()`

Procedimiento para crear una muestra en el sistema.

-  `private void cargar_btnActionPerformed()`

Procedimiento para cargar un patrón desde un archivo csv a una muestra.

❖ Crear_patron

```
import java.io.IOException;
import javax.swing.JOptionPane;
```

➤ Librerías

La librería `java.io.IOException` es utilizada para capturar la excepción al leer o escribir en el archivo binario.

La librería `javax.swing.JOptionPane` sirve para mostrar ventanas emergentes.

➤ Variables Globales de la clase `_(Crear_patron)`

```
int[][] patron = null;
```


Variable global de tipo matriz de enteros, utilizada para guardar temporalmente la matriz cargada desde el csv.

➤ Función Main


Esta clase no contiene una función main, es invocada desde la clase de `Crear_muestra`.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

-  `private void cargar_btnActionPerformed`

Procedimiento para cargar el patrón del csv al patrón del sistema.

-  `private void crear_btnActionPerformed`

Procedimiento para crear un patrón en el sistema.

❖ Actualizar_Tabla

```
import java.util.List;  
import javax.swing.JButton;  
import javax.swing.JTable;  
import javax.swing.table.DefaultTableModel;
```

➤ Librerías

La librería `java.util.List` es usada para el manejo de listas dinámicas.

La librería `javax.swing` es utilizada para importar los elementos de la interfaz gráfica.

➤ Variables Globales de la clase `_(Actualizar_Tabla)`



Sin variables globales.


➤ Función Main

No contiene función main. Esta clase es invocada desde la clase Administrador, al hacer modificaciones en alguna de las tablas del módulo.


➤ Procedimientos, métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:


-  `15` `public static void nuevo_elemento(`
Método para agregar un elemento a la tabla de Investigadores.
-  `27` `public static void actualizar_elemento(`
Método para actualizar un elemento de la tabla de Investigadores.

- 36  `public static void actualizar_muestra{`


Método para actualizar un elemento de la tabla de Muestras.

- 41  `public static void eliminar_elemento{`

Método para eliminar un elemento de la tabla de Investigadores.

- 45  `public static void nueva_muestra{`

Método para agregar un elemento a la tabla de muestras.

- 55  `public static void nuevo_patron`

Método para agregar un elemento a la tabla de patrones.

❖ Carga_csv

```
import java.io.File;
import java.io.IOException;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
```

➤ Librerías

La librería `java.io.File` es utilizada para crear objetos de tipo `File` y poder utilizar los métodos para manipular archivos.

La librería `java.io.IOException` es utilizada para capturar excepciones de entrada/salida.

La librería `javax.swing` se utiliza para importar elementos de la interfaz gráfica. El `JFileChooser` es la interfaz que permite seleccionar archivos.

➤ Variables Globales de la clase `_Cargar_csv`

```
public String ruta;
```


Variable global de tipo `String` que almacena la ruta del archivo csv que seleccione el usuario.

➤ Función Main


Esta clase no tiene función main. Esta clase es invocada desde distintas partes de la clase `Administrador` y también desde las clases `Crear_muestra` y `Crear_patron`.

➤ Procedimientos, métodos y Funciones utilizadas


A continuación, se dará una explicación general de lo que hace cada función:

- 23  `public int cargarCSV()`


Método para seleccionar el archivo desde el `JFileChooser`.

- `38`  `public void cargarAlArray(String rutaArchivo)`

Método para ingresar los investigadores del archivo csv en el ArrayList de investigadores.

- `63`  `public int[][] cargarMuestra(String rutaArchivo)`

Método para cargar el patrón de un csv en una matriz de enteros int [][].

- `86`  `public void cargarcsvMuestra(String rutaArchivo, int`

Método para hacer una carga masiva de muestras desde un archivo csv.

❖ Escribir_InvestigadorBinario

```
import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
- import javax.swing.JOptionPane;
```

➤ Librerías

La librería java.io es utilizada para el manejo de archivos, lectura y escritura de estos. También para capturar excepciones en caso de existir al manipularlas.

La clase javax.swing.JOptionPane se utiliza para mostrar ventanas emergentes.

➤ Variables Globales de la clase _(Escribir_InvestigadorBinario)


Sin variables globales.

➤ Función Main


Esta clase no contiene main. Sus métodos son llamados desde otras clases.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- 20  `public static void escribir_investigadorbin()`

Método para serializar los objetos de tipo Investigador en un archivo llamado investigadores.bin.

- `41`  `public void leer investigadorbin()`

Método para deserializar los objetos de tipo Investigador y cargarlos al ArrayList.

❖ Escribir_investigador

```
import java.util.List;  
import java.util.ArrayList;  
import javax.swing.JComboBox;
```

➤ Librerías

La librería java.util se utiliza para el manejo de las listas dinámicas.

La librería javax.swing.JComboBox es utilizada para el manejo de este tipo de objetos de la interfaz.

➤ Variables Globales de la clase _(Escribir_investigador)

```
public static List<Investigador> investigadores = new ArrayList<Investigador> ();
```



Esta es el ArrayList que almacena y los objetos de tipo Investigador durante la ejecución del programa.

➤ Función Main

No contiene función main. La lista estática es invocada desde muchas otras clases dentro del programa.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  `public static boolean comparar_codigo`
Método para buscar coincidencias de código en el ArrayList de investigadores.
-  `public static void Escribir_investigadorCombo`
Método para añadir los investigadores creados o cargados en el JComboBox de investigadores.

❖ Escribir_muestra

```
import java.util.ArrayList;  
import java.util.List;  
import javax.swing.JComboBox;
```

➤ Librerías

Las librerías java.util se usan para el manejo de listas dinámicas.

La librería javax.swing.JComboBox se utiliza para el manejo de este tipo de objetos de la interfaz.

➤ Variables Globales de la clase _(Escribir_muestra)

```
public static List <Muestra> muestras = new ArrayList<>();
```



Este es el ArrayList estático que se encarga de almacenar y manipular los datos de muestras durante la ejecución del programa.

➤ Función Main



Esta clase no tiene una función main. Sus métodos y atributos son invocados desde varias clases.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-   `public static boolean comparar_codigo(String codigo)`

Método para validar que un código introducido no sea igual a uno que ya exista en el sistema.

-   `public static void Escribir_muestraCombo(JComboBox combo, List<Muestra> muestra)`

Método para agregar las muestras al JComboBox de muestras.

❖ Escribir_muestraBinaria

```
import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import javax.swing.JOptionPane;
```

➤ Librerías

La librería java.io es utilizada, en general, para el manejo de archivos, entrada y salida de datos, y manejo de excepciones.

La librería javax.swing.JOptionPane sirve para mostrar ventanas emergentes.

➤ Variables Globales de la clase _(Escribir_muestraBinaria)


Sin variables globales.

➤ Función Main


Esta clase no contiene un método main. Sus métodos son invocados desde otras clases.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- 19  `public static void escribir muestrabin()`

Método para serializar archivos de tipo Muestra en un archivo llamado muestras.bin.

- 39  `public void leer_muestrabin()`

Método para deserializar los objetos de tipo Muestra.

❖ Escribir_patron

```
import java.util.ArrayList;  
import java.util.List;  
import javax.swing.JComboBox;
```

➤ Librerías

Las librerías java.util se usan para el manejo de listas dinámicas.

La librería javax.swing.JComboBox se utiliza para el manejo de este tipo de objetos de la interfaz.

➤ Variables Globales de la clase _(Escribir_patron)

```
public static List<Patron> patrones = new ArrayList<> ();
```


Este es el ArrayList encargado de almacenar y manipular los datos de tipo Patrón durante la ejecución del programa.

➤ Función Main


Esta clase no contiene un método main. Sus métodos son invocados desde otras clases.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  `public static boolean comparar_codigo(String codigo)`

Método para comparar los códigos existentes en el sistema.

-  `public static void Escribir_patronCombo`

Método para agregar los patrones en el JComboBox de patrones del módulo de Investigador

❖ **Escribir_patronBinario**

```
import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import javax.swing.JOptionPane;
```

➤ **Librerías**

La librería java.io es utilizada para el manejo de archivos, entrada y salida de datos y manejo de excepciones en general.

➤ **Variables Globales de la clase _ (Escribir_patronBinario)**


Sin variables globales.

➤ **Función Main**

Esta clase no tiene una función main. Sus métodos son invocados desde varias clases.

➤ **Procedimientos, métodos y Funciones utilizadas**

A continuación, se dará una explicación general de lo que hace cada función:

- 22  `public static void escribir_patronbin()`

Método para serializar los objetos de tipo Patrón en un archivo llamado patrones.bin.

- 42  `public void leer_patronbin()`

Método para deserializar los objetos de tipo Patrón.

❖ Grafica_investigadores

```
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.util.List;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
```

➤ Librerías

La librería java.awt es utilizada para darle atributos a ciertos elementos de las interfaces gráficas.

La librería java.util.List sirve para manejar listas dinámicas.

La librería org.jfree.chart es utilizada para crear la gráfica top 3 investigadores.

➤ Variables Globales de la clase _(Grafica_investigadores)

Sin variables globales.

➤ Función Main

No contiene función main. Su método es llamado desde la clase de Administrador.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación, se dará una explicación general de lo que hace cada función:

- `public void graficar(int no1, int no2, int no3, String nom1, String nom2, String nom3) {`

Método para crear la gráfica con los investigadores con más número de experimentos.

❖ Investigador

```
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
```

➤ Librerías

La librería java.io.Serializable es utilizada para serializar objetos.

La librería java.util es utilizada para el manejo de listas dinámicas.

➤ Variables Globales de la clase _(Investigador)

```
private String codigo;
private String nombre;
private char genero;
private int numExperimentos = 0;
private List<Muestra> muestra asignada = new ArrayList<>();
private String contrasenia;
private int contadorDeAnalisis = 0;
```

Las variables globales declaradas son los atributos que tienen los objetos de tipo Investigador.

➤ Función Main

Sin función main.

➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

```
● 49  [ ]  public String getCodigo()
63  [ ]  public String getNombre()
77  [ ]  public char getGenero()
91  [ ]  public int getNumExperimentos()
104  [ ]  */
    [ ]  public List getMuestra_asignada()
119  [ ]  public String getContrasenia()
```

```
130 public int getContadorDeAnalisis()
```

Métodos getter, retornan el atributo especificado.

- ```
56 public void setCodigo(String codigo)
70 public void setNombre(String nombre)
84 public void setGenero(char genero)
98 public void setNumExperimentos(int numExperimentos)
112 public void setMuestra_asignada(Muestra muestra_asignada)
126 public void setContrasenia(String contrasenia)
134 public void setContadorDeAnalisis(int contadorDeAnalisis)
```

Métodos setter, establecen un valor a un atributo especificado.

## ❖ Leer\_contrasenias

```
import java.util.List;
```

### ➤ Librerías

La librería `java.util.List` es utilizada para el manejo de listas dinámicas.

### ➤ Variables Globales de la clase `_`(Leer\_contrasenias)


*Sin variables globales.*

### ➤ Función Main

No tiene función main. Sus funciones y métodos son invocados desde otras clases.

### ➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

●  `public static int leer_contrasenias(String user, String password, List<Investigador> investigador)`

Procedimiento para leer las contraseñas de los investigadores y retornar un valor que indica si se encontraron coincidencias.

## ❖ Leer\_csv

```
import java.io.BufferedReader;
import java.io.FileReader;
import javax.swing.JOptionPane;
import interfaces.Administrador;
import static java.lang.Math.sqrt;
import java.util.ArrayList;
import java.util.List;
```

### ➤ Librerías

La librería java.io es utilizada para leer archivos csv.

La librería javax.swing.JOptionPane se utiliza para mostrar ventanas emergentes.

La librería java.lang.Math proporciona ciertos métodos para realizar cálculos matemáticos.

La librería java.util en este caso, se utilizó para el manejo de listas dinámicas.

### ➤ Variables Globales de la clase \_(Leer\_csv)

```
Investigador investigador_temp = new Investigador();

private BufferedReader lector; //Objeto que lea el ar
private String linea; //String para saber si una flia
private String partes[] = null; //vector tipo string
```

Se crea un objeto de tipo Investigador temporal.


Las siguientes variables globales son utilizadas para leer los datos del csv.

### ➤ Función Main


No contiene función main.

## ➤ Procedimientos, métodos y Funciones utilizadas


A continuación se dará una explicación general de lo que hace cada función:

- 25  `public void leer_archivoInvestigadores(String csv)`

Método para leer un archivo csv de investigadores.

- 87  `public int [][] leer_patronNuevaMuestra(String csv)`

Método para leer el patrón de un csv para una muestra.

- 127  `public void leer_cargaMuestra(String csv, int estado)`

Método para leer una carga masiva de muestras o patrones de un csv.

## ❖ Muestra

```
import java.io.Serializable;
```

### ➤ Librerías

La librería `java.io.Serializable` es utilizada para serializar objetos.

### ➤ Variables Globales de la clase `_`(Muestra)

```
private String codigo;
private String descripcion;
private int[][] patron;
private String estado = "Ingreso";
```

Estas variables globales son los atributos de un objeto de Muestra.

### ➤ Función Main

No contiene una función main.

### ➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- ```
17 | public String getCodigo()  
27 | public String getDescripcion()  
37 | public int[][] getPatron()  
47 | public String getEstado()
```

Métodos getter, retornan el atributo especificado.

- ```
22 | public void setCodigo(String codigo)
32 | public void setDescripcion(String descripcion)
42 | public void setPatron(int[][] patron)
52 | public void setEstado(String estado)
```

Métodos setter, modifican el atributo especificado.

## ❖ MuestraDeInvestigador

```
import java.util.ArrayList;
import java.util.List;
```

### ➤ Librerías

La librería java.util es utilizada para el manejo de listas dinámicas.

### ➤ Variables Globales de la clase \_(MuestraDeInvestigador)


*Sin variables globales.*

### ➤ Función Main

Sin función main.

### ➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

●  `public void borrar_MuestraInvestigador(int indexInvestigador, String descripcion_muestra, List<Investigador> investigadores)`

Método para borrar una muestra del ArrayList de un investigador.

## ❖ Patron

```
import java.io.Serializable;
```

### ➤ Librerías

La librería `java.io.Serializable` es usada para serializar objetos.

### ➤ Variables Globales de la clase `_(Patron)`

```
private String codigo;
private String nombre;
private int[][] patron;
```




Estas variables globales son los atributos que tienen todos los objetos de tipo Patrón.

### ➤ Función Main




Sin función main.

### ➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- 19  `public String getCodigo()`
- 27  `public String getNombre()`
- 35  `public int[][] getPatron()`

Métodos getter, retornan el atributo indicado.

- 23  `public void setCodigo(String codigo)`
- 31  `public void setNombre(String nombre)`
- 39  `public void setPatron(int[][] patron)`

Métodos setter, modifican el atributo indicado.



## ❖ RenderTabla

```
import java.awt.Component;
import javax.swing.JButton;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
```

### ➤ Librerías

La librería `java.awt.Component` es utilizada para validar si fue presionado un botón en la tabla.

La librería `javax.swing` es utilizada para importar elementos de la interfaz gráfica.

### ➤ Variables Globales de la clase `_RenderTabla`




*Sin variables globales*

### ➤ Función Main




Sin función main.

### ➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

- 19  `public String getCodigo()`
- 27  `public String getNombre()`
- 35  `public int[][] getPatron()`

Métodos getter, retornan el atributo indicado.

- 23  `public void setCodigo(String codigo)`
- 31  `public void setNombre(String nombre)`
- 39  `public void setPatron(int[][] patron)`

Métodos setter, modifican el atributo indicado.

## ❖ **Tabla\_resultados**

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.JButton;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import interfaces.Investigador;
import java.awt.Desktop;
import java.io.File;
import java.io.IOException;
import javax.swing.JOptionPane;
```

### ➤ **Librerías**

La librería java.io se utiliza para entrada y salida de datos, manejo de archivos y manejo de excepciones.

Las librerías java.text y java.util.Date se utilizan para manipular fechas.

La librería javax.swing se utiliza para agregar los elementos de la interfaz gráfica.

La librería java.awt.Desktop se utilizó para abrir un archivo html en el navegador.

### ➤ **Variables Globales de la clase \_(Tabla\_resultados)**


*Sin variables globales*

### ➤ **Función Main**


Sin función main.

## ➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  `public void escribir_txtResultados(JTable tabla_resultados, int resultado, String path, int contadorDeAnálisis, String nombreMuestra,`

Método para escribir el historial en un archivo de texto.

-  `public void leer_txtResultados(String path)`

Método utilizado para leer el archivo con el historial de análisis del investigador.

## ❖ Top\_3

```
import java.util.List;
```

### ➤ Librerías

La librería `java.util.List` es utilizada para manejar listas dinámicas.

### ➤ Variables Globales de la clase `_ (Top_3)`

```
private String codigo1, codigo2, codigo3, nombre1, nombre2, nombre3;
private int num1, num2, num3;
```


Atributos de los 3 investigadores con más experimentos.

### ➤ Función Main


Sin función main.

### ➤ Procedimientos, métodos y Funciones utilizadas

A continuación se dará una explicación general de lo que hace cada función:

-  `19` `public void obtener_top3(List<Investigador> investigadores)`

Método para obtener los códigos de los 3 investigadores con más experimentos.

-  `81` `public void nombres_experimentos(List<Investigador> investigador, String codigo1, String codigo2, String codigo3)`

Método que ordena por nombre, en orden descendente los nombres y números de experimentos de los investigadores.