Universidad de San Carlos de Guatemala
FACULTAD DE INGENIERÍA

PRIMER SEMESTRE 2024

Auxiliar: Rodrigo Alejandro Hernández de León

Curso: Laboratorio de Introducción a la Programación y Computación 1

Sección: A

Tema:

Manual Técnico practica 2

Estudiante: Carlos Heraldo Quiná Corona

Reg: 202300645

INTRODUCCIÓN

El siguiente manual técnico proporciona descripciones sobre el funcionamiento y cierta lógica de diferentes clases, métodos y paquetes, esto puede ayudar a cualquier persona a entender cada parte del código.

- ## Objetivo

El motivo de la realización de este manual técnico es dar a conocer, y explicar cómo es que funciona internamente el programa requerido para la practica 2, esto con el objetivo de que cualquiera que vea el código del programa sepa que se realizar en cada parte y ayude a una futura modificación.

Especificaciones Técnicas

- ## Requisitos de hardware
  - Procesador: Se requiere un Intel Pentium 4 o un procesador AMD con soporte para instrucciones SSE2.
  - Memoria RAM: Debes tener al menos 2 GB de memoria RAM.
  - Espacio en disco duro: Asegúrate de tener al menos 1 GB de espacio disponible en el disco duro para la instalación del JDK.

- ## Requisitos de software
  - Sistema Operativo Windows 7 o superior.
  - Java Development Kit (JDK) versión 15 o superior.
  - Lenguaje de Programación JAVA.
  - NetBeans IDE 14 o superior.
  - Librerías mínimas ya integradas

LOGICA DEL PROGRAMA

NOMBRE DEL PAQUETE	DESCRIPCIÓN	CLASES CONTENIDAS EN LOS PAQUETES
Clases	Este paquete contiene las clases necesarias para el funcionamiento del programa, pues aquí se crean los objetos que serán añadidos a una linkedlist	• Rutaelegida • Rutas • Vehiculo
Gui	Este paquete contiene las clases que además de tener diversos métodos, contienen interfaces graficas	• Cargaruta • Editardist • paradatos
Hilos	Este paquete contiene el único hilo responsable de mover los vehículos necesarios	• movimiento
imágenes	Este paquete contiene las imágenes que se usaron para cada vehiculo	
Main	Este paquete contiene la clase Main que es la primera en ejecutarse	• Main
		Cantidad total de clases: 8

- # Clases Utilizadas

Paquete Clases

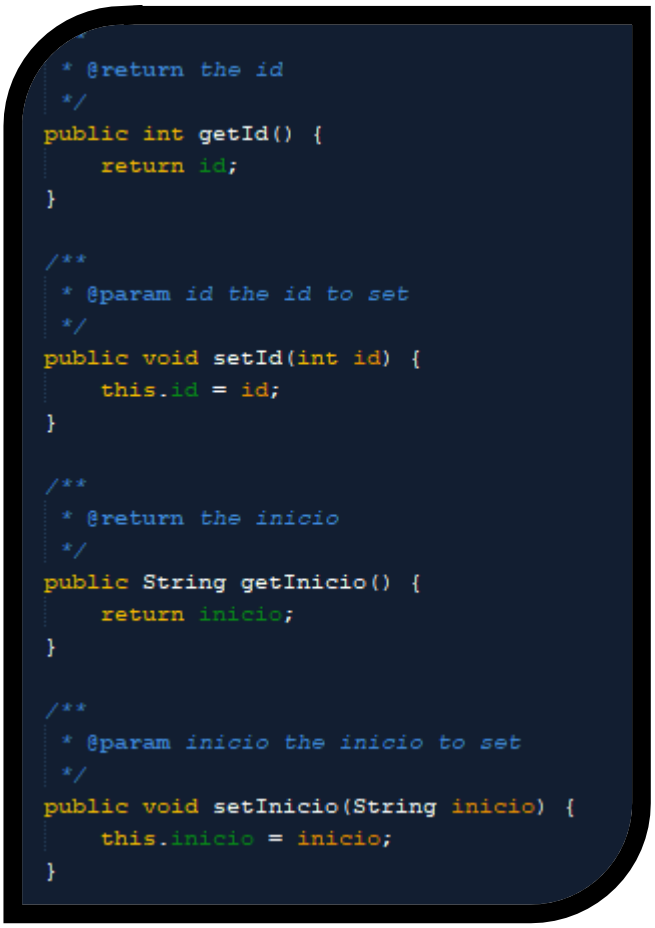
Clase rutaelegida

Esta clase se usa para crear los datos del historial

Atributo o variable	Tipo
Id	int
Inicio	String
Fin	String
Distancia	Int
Vehiculo	String
Gasol	double

Encapsulamiento

Encargado de obtener o establecer valores ya en un objeto de la clase, se muestran ahora, pero no se volverán a mostrar pues seria redundante.

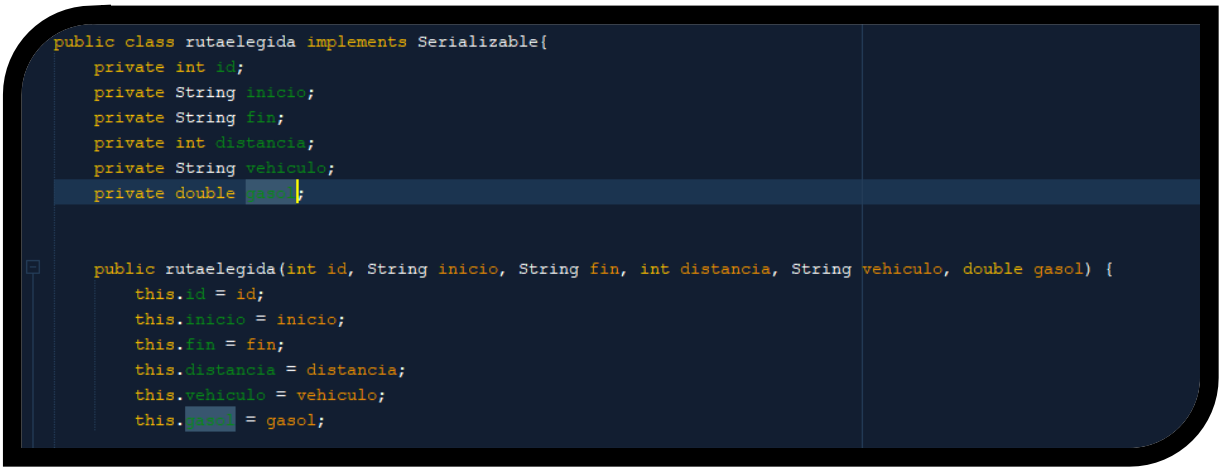


```
     * @return the id
     */
    public int getId() {
        return id;
    }

    /**
     * @param id the id to set
     */
    public void setId(int id) {
        this.id = id;
    }

    /**
     * @return the inicio
     */
    public String getInicio() {
        return inicio;
    }

    /**
     * @param inicio the inicio to set
     */
    public void setInicio(String inicio) {
        this.inicio = inicio;
    }
```

• • •

Constructor

Encargado de dictar los datos necesarios para la creación de un objeto de la clase



```java
public class rutaelegida implements Serializable{
    private int id;
    private String inicio;
    private String fin;
    private int distancia;
    private String vehiculo;
    private double gasol;


    public rutaelegida(int id, String inicio, String fin, int distancia, String vehiculo, double gasol) {
        this.id = id;
        this.inicio = inicio;
        this.fin = fin;
        this.distancia = distancia;
        this.vehiculo = vehiculo;
        this.gasol = gasol;
```

Clase Rutas

Esta clase es la encargada de ordenar los datos que vienen desde el csv

Atributo o variable	Tipo
Id	int
Inicio	String
Fin	String
Distancia	Int

Clase vehículo

Esta clase se encarga de crear los datos de los vehículos que serán mostrados en la pestaña de iniciar viajes

Atributo o variable	Tipo
Consumo	Double
Capacidad	Double
Nombre	String
Distanciatotal	Int
Distanciaactual	Int
Id	Int
Inicio	String
Fin	String

Paquete gui

Clase Cargaruta

Clase que además de contener la interfaz grafica, maneja la mayoría de los procesos del programa

Atributo o variable	Tipo
javax.swing.JLabel	final1
javax.swing.JLabel	final2
javax.swing.JLabel	final3
javax.swing.JLabel	inicio1
javax.swing.JLabel	inicio2
javax.swing.JLabel	inicio3
javax.swing.JLabel	nombre1
javax.swing.JLabel	nombre2
javax.swing.JLabel	nombre3
javax.swing.JLabel	vehi1
javax.swing.JLabel	vehi2
javax.swing.JLabel	vehi3
javax.swing.JButton	jButton1
javax.swing.JButton	jButton2
javax.swing.JButton	jButton3
javax.swing.JButton	jButton4
javax.swing.JButton	jButton5
javax.swing.JButton	jButton6
javax.swing.JComboBox	jComboBox1

javax.swing.JComboBox	jComboBox2
javax.swing.JComboBox	jComboBox3
javax.swing.JLabel	jLabel1
javax.swing.JLabel	jLabel2
javax.swing.JLabel	jLabel3
javax.swing.JLabel	jLabel5
javax.swing.JLabel	jLabel6
javax.swing.JLabel	jLabel7
javax.swing.JPanel	jPanel1
javax.swing.JPanel	jPanel2
javax.swing.JPanel	jPanel3
javax.swing.JPanel	jPanel4
javax.swing.JPanel	jPanel5
javax.swing.JPanel	jPanel6
javax.swing.JPanel	jPanel7
javax.swing.JScrollPane	jScrollPane1
javax.swing.JScrollPane	jScrollPane2
javax.swing.JTabbedPane	jTabbedPane1
javax.swing.JTable	jTable1
javax.swing.JButton	recargar1
javax.swing.JButton	recargar2
javax.swing.JButton	recargar3
javax.swing.JTable	tablafinal

javax.swing.JButton	iniciar1
javax.swing.JButton	iniciar2
javax.swing.JButton	iniciar3
javax.swing.JButton	iniciartodos
int	cont23

Método leerarchivo

Método que se encarga de abrir una ventana emergente para subir un archivo csv, y leerlo correctamente

Método llenartabla

Método encargado de llenar la tabla que aparece en la pantalla principal del programa

```
public void llenartabla() {


    DefaultTableModel t1 = new DefaultTableModel(new String[]{"id","inicio","fin","distancia"},listarutas.size());
    jTable1.setModel(t1);
    TableModel t2 = jTable1.getModel();



    for (int i = 0; i < listarutas.size(); i++) {

        rutas llenado = listarutas.get(i);
        t2.setValueAt(llenado.getId(), i, 0);
        t2.setValueAt(llenado.getInicio(), i, 1);
        t2.setValueAt(llenado.getFin(), i, 2);
        t2.setValueAt(llenado.getDistancia(), i, 3);
                                            }


}
```

Método llenartablafinal

Método encargado de llenar la tabla del historial de viajes

```
ublic void llenartablafinal() {


    DefaultTableModel t3 = new DefaultTableModel(new String[]{"id","inicio","fin","distancia","vehiculo","gasolina gastada"},rutaelegidalist.size());
    tablafinal.setModel(t3);
    TableModel t4 = tablafinal.getModel();

    for (int i = 0; i < main.Main.rutaelegidalist.size(); i++) {
        // clase a secas                linkedlist de main
        rutaelegida llenado2 = main.Main.rutaelegidalist.get(i);
        t4.setValueAt(llenado2.getId(), i, 0);
        t4.setValueAt(llenado2.getInicio(), i, 1);
        t4.setValueAt(llenado2.getFin(), i, 2);
        t4.setValueAt(llenado2.getDistancia(), i, 3);
        t4.setValueAt(llenado2.getVehiculo(), i, 4);
        t4.setValueAt(llenado2.getGasol(), i, 5);
                                            }


}
```

Método llenarcombo 1, 2 y 3

Método encargado de llenar con datos el combobox 1, 2 y 3

```
lic void llenarcombol(){
    for (int i = 0; i < listarutas.size(); i++) {
     jComboBox2.addItem(listarutas.get(i).getInicio());
     jComboBox2.addItem(listarutas.get(i).getFin());

    }

}

    public void llenarcombo2(){
    for (int i = 0; i < listarutas.size(); i++) {
     jComboBox3.addItem(listarutas.get(i).getFin());
     jComboBox3.addItem(listarutas.get(i).getInicio());

    }

}

    public void llenarcombo3(){

     jComboBoxl.addItem(motol);
     jComboBoxl.addItem(moto2);
     jComboBoxl.addItem(moto3);
     jComboBoxl.addItem(estandarl);
     jComboBoxl.addItem(estandar2);
     jComboBoxl.addItem(estandar3);
     jComboBoxl.addItem(premiuml);
     jComboBoxl.addItem(premium2);
     jComboBoxl.addItem(premium3);



}
```

Método Rutasinic

Método encargado de crear un nuevo objeto de vehículo (con un máximo de 3) y agregarlo a una linkedlist preparada para mantener los datos

Clase Editardist

Clase que posee una ventana y que permite editar la distancia de un viaje

Atributo o variable	Tipo
jButton1	javax.swing.JButton
jButton2	javax.swing.JButton
jLabel1	javax.swing.JLabel
jLabel3	javax.swing.JLabel
jTextField1	javax.swing.JTextField
jTextField2	javax.swing.JTextField

Clase Paradatos

Clase que contiene una interfaz grafica para mostrar diversos datos como la gasolina restante, etc…

Atributo o variable	Tipo
datos1	javax.swing.JLabel
Datos2	javax.swing.JLabel
Datos3	javax.swing.JLabel
jPanel1	javax.swing.JPanel

Paquete hilos

Clase Movimiento

Clase que contiene el hilo que hacer que todos los vehículos se muevan

Atributo o variable	Tipo
Vehiculo	javax.swing.JLabel
Limiteder	Double
Derecha	Boolean
Numvehiculo	int
Datos	Jlabel
Km	Int
Gaso	Double
Estimado	Double
Contadorex	int
Año	int
Mes	Int
Dia	Int
Hora	Int
minutos	Int
Segundos	int
X	int

Método pausar

Método encargado de pausar el hilo

```
    }
public void pausar(){
        setDerecha(false);
```

Método reanudar

Método encargado de reanudar el hilo

```
public void reanudar(){
        setDerecha(true);

}
```

Paquete main

Clase Main

Clase principal, encargada de agregar, crear y manejar las linkedlist de todas las clases que lo necesiten

Atributo o variable	Tipo
Ids	Int
Listarutas	LinkedList<>
Rutaelegidalist	LinkedList<>
Vehiculo	LinkedList<>
Entrada	ObjectOutputStream
Salida	ObjectInputStream
Entrada2	ObjectOutputStream
Salida2	ObjectInputStream
Datos2	javax.swing.JLabel
Datos3	javax.swing.JLabel
jPanel1	javax.swing.JPanel

Método Serializar (1 y 2)

Estos métodos se encargan de serializar la tabla inicial y la tabla del historial de viajes

```
ublic static void serializart1(){
try{
salida = new ObjectOutputStream(new FileOutputStream("Serializados/tabla1.bin"));
salida.writeObject(listarutas);
salida.close();
}catch(Exception e){}
}

public static void serializart2(){
try{
salida2 = new ObjectOutputStream(new FileOutputStream("Serializados/tabla2.bin"));
salida2.writeObject(rutaelegidalist);
salida2.close();
}catch(Exception e){}
}
```

Método Deserializar (1 y 2)

Métodos entregados de deserializar la tabla inicial y la tabla del historial de viajes

```
blic static Object deserializar1(){
try{
entrada = new ObjectInputStream(new FileInputStream("Serializados/tabla1.bin"));
LinkedList<rutas> rutasser = (LinkedList<rutas>) entrada.readObject();
entrada.close();
return rutasser;
}catch(Exception e){}
        return null;

}


public static Object deserializar2(){
try{
entrada2 = new ObjectInputStream(new FileInputStream("Serializados/tabla2.bin"));
LinkedList<rutaelegida> rutas2ser = (LinkedList<rutaelegida>) entrada2.readObject();
entrada2.close();
return rutas2ser;
}catch(Exception e){}
        return null;
}
```

Método añadir rutas

Por medio de este método puedes agregar rutas a la linkedlist

```
public static void añadirruta(rutas rutas){
    listarutas.add(rutas);
    serializart1();

}
```

Método rutalista

Por medio de este método puedes agregar un objeto a la lista de rutas creadas

```
public static void rutalista(rutaelegida rutae){
    rutaelegidalist.add(rutae);
    serializart2();
```

Método avehiculo

Por medio de este método puedes agregar vehículos a la linkedlist de vehiculos

```
public static void avehiculo(vehiculo vehi){
    vehiculo.add(vehi);


}
```

- **DIAGRAMAS**

Diagrama de Clases

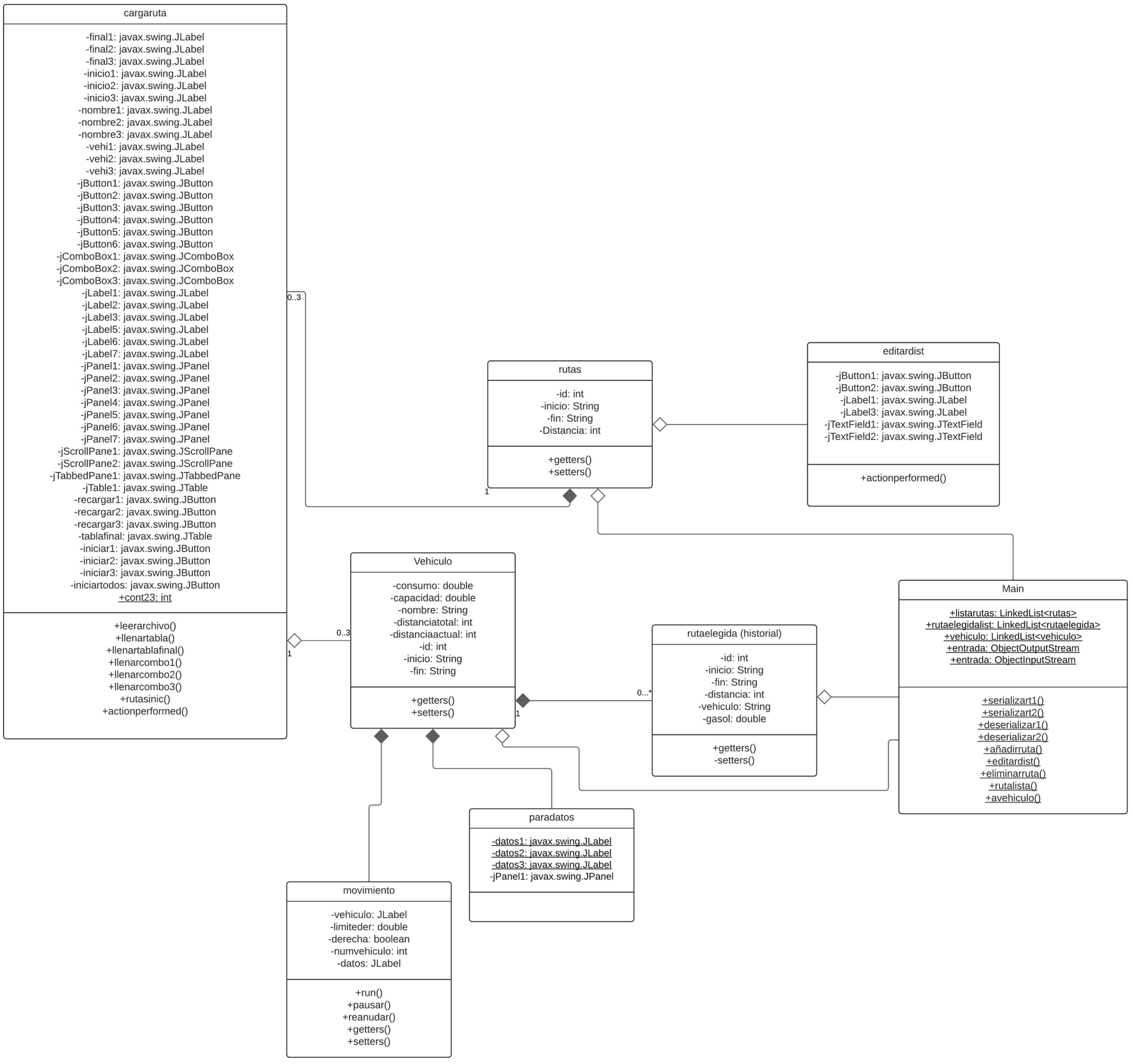


Diagrama de Flujo General