

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS

CATEDRÁTICO: ING. WILLIAM ESTUARDO ESCOBAR ARGUETA

TUTOR ACADÉMICO: JOSUÉ RODOLFO MORALES CASTILLO



MANUAL TÉCNICO
DE PROGRAMA
UDRIVE

ENNER ESAÍ MENDIZABAL CASTRO

CARNÉ: 202302220

SECCIÓN: B

GUATEMALA, 23 DE MARZO DEL 2,024

ÍNDICE

ÍNDICE	1
INTRODUCCIÓN	4
OBJETIVOS	4
1. GENERAL	4
2. ESPECÍFICOS	4
ALCANCES DEL SISTEMA	4
ESPECIFICACIÓN TÉCNICA	5
• REQUISITOS DE HARDWARE	5
• REQUISITOS DE SOFTWARE	5
DESCRIPCIÓN DE LA SOLUCIÓN	6
LÓGICA DEL PROGRAMA	9
➤ UDrive Librerías	9
➤ Variables Globales de la clase UDrive	9
➤ Función Main	10
➤ Métodos y Funciones utilizadas	10
❖ Vehiculo	12
➤ Librerías	12
➤ Variables Globales de la clase Vehiculo	12
➤ Función Vehículo	12
➤ Métodos y Funciones utilizadas	12
❖ Viajes	14
➤ Librerías	14
➤ Variables Globales de la clase Viajes	14
➤ Función Viajes	14
➤ Métodos y Funciones utilizadas	15

❖ Generar	16
➤ Librerías	16
➤ Variables Globales de la clase Generar	16
➤ Función Generar	16
➤ Métodos y Funciones utilizadas	17
❖ Historial	18
➤ Librerías	18
➤ Variables Globales de la clase Historial	18
➤ Función Historial	18
➤ Métodos y Funciones utilizadas	18
❖ Inicio	19
➤ Librerías	19
➤ Variables Globales de la clase Inicio	19
➤ Función Inicio	19
➤ Métodos y Funciones utilizadas	19
❖ Línea	20
➤ Librerías	20
➤ Variables Globales de la clase Inicio	20
➤ Función Línea	20
➤ Métodos y Funciones utilizadas	20
❖ Menu	22
➤ Librerías	22
➤ Variables Globales de la clase Inicio	22
➤ Función Menu	22
➤ Métodos y Funciones utilizadas	22
❖ Ruta1, Ruta2, Ruta3	24

➤ Librerías	24
➤ Variables Globales de la clase Ruta1, Ruta2, Ruta3.....	24
➤ Función Ruta1, Ruta2, Ruta3.....	24
➤ Métodos y Funciones utilizadas	25
❖ Ruta11, Ruta22, Ruta33	26
➤ Librerías	26
➤ Variables Globales de la clase Inicio	26
➤ Función Inicio	26
➤ Métodos y Funciones utilizadas	27

INTRODUCCIÓN

Este manual tiene la finalidad de proporcionar todos los aspectos técnicos y funcionales acerca del programa UDrive, para que el funcionamiento de este pueda ser clarificado y, si se deseara, replicado.

OBJETIVOS

1. GENERAL

- 1.1. Proporcionar información técnica del programa al lector.

2. ESPECÍFICOS

- 2.1. Objetivo 1: Especificar el funcionamiento y librerías de cada una de las clases del programa.
- 2.2. Objetivo 2: Proporcionar los requerimientos mínimos e indispensables para el funcionamiento correcto del programa UDrive.

ALCANCES DEL SISTEMA

Con este manual se pretende que el lector pueda alcanzar un tipo de comprensión más profundo que el que se obtendría mediante la lectura del manual de Usuario, proporcionando información más técnica del programa como una descripción más detallada y especificada de cada una de las clases, métodos y procedimientos del programa UDrive.

ESPECIFICACIÓN TÉCNICA

- **REQUISITOS DE HARDWARE**

- Procesador: 2 GHz o superior
- Disco duro: Espacio se al menos 1.5 GB
- RAM: 2GB o superior

- **REQUISITOS DE SOFTWARE**

- Sistema operativo: Windows, macOS o Linux
- Java development kit (JDK)
- Entorno de desarrollo integrado (IDE) compatible con Java (se recomienda NetBeans)
- Un archivo .csv que posea los datos que serán importados al programa.

DESCRIPCIÓN DE LA SOLUCIÓN

- **CREACIÓN DE LA INTERFAZ GRÁFICA:** Para toda la interfaz gráfica de este programa, usara tanto las librerías AWT y SWING, mezclando DragAndDrop o realizando la interfaz manualmente donde fuera necesario, para que, de esta manera, se pudiera optimizar y lograr lo mejor de ambas librerías.
- **LECTURA DEL ARCHIVO .CSV:** Para la lectura del archivo .csv, se decidió crear un filtro por medio de *FileNameExtensionFilter*, y escoger el archivo por medio delo buscador que se genera por medio de *JFileChooser*. Una vez con los datos del archivo .csv, se decidió tratar todo por medio de un vector de tipo String, el cual se encargaría de separar por String, cada uno de los elementos que, como debería ser, vendrían de 3 en 3. Luego de ingresar todos los datos dentro del vector de tipo String, se usó un ciclo que iría de 3 en 3, para que se capturaran los datos de forma ordenada y luego, con esos datos, se pudiera generar la tabla en donde se mostrarían los datos del archivo .CSV.
- **GENERACIÓN DE LA TABLAS:** Como se mencionó antes, se optó por la utilización mixta de AWT y SWING con DragAndDrop, por lo que las tablas fueron creadas mediante este último y se ingresar los datos en forma de vectores por cada fila, es decir que se crearon vectores y se ingresaron ordenadamente en cada fila para formar la tabla.
- **MANEJO DE CONDUCTORES Y VEHÍCULOS:** Para el manejo de estos, se optó por utilizar varios vectores que se encargarían de guardar la información de la posición de cada uno, así como la disponibilidad de estos por medio de los vectores de tipo lógico.
- **GENERACIÓN DE VIAJE:** Para la generación de los viajes, se utilizaron los valores de la tabla que se creó con la importancia del archivo .csv y los valores de los distintos vectores que proporcionaban información

sobre los conductores y su disponibilidad (que es lo más importante). Una vez se crea el viaje, este aparecerá dentro de la ventana de INICIO DE VIAJES, y el conductor y vehículo escogidos será removidos de la lista para que no se puedan escoger nuevamente, adicionalmente si se detecta, por medio de un vector booleano, que ninguno de los tres conductores está disponible, entonces el programa no permitirá que se puedan generar nuevos viajes.

- **VENTAN INICIO DE VIAJES:** Para la creación de esta ventana que es la más compleja de todas, se creó un ciclo el cual verificaría el valor del tipo de vehículo que se escogió (que está dentro de un vector previamente mencionado), para que de este modo se pudiera se coloque la imagen correspondiente al vehículo. Además, para que todo se colocara en orden , dependiendo de la posición, se sumó una cierta cantidad de pixeles que distanciaría cada una de las 3 rutas de las otras. Dentro de todo esto, se colocan los datos de cada viaje, la gasolina, etc.
- **HILOS:** Esta fue la solución más difícil de implementar. Para el movimiento de los vehículos, se usaron los hilos, los cuales tomarían el valor de la posición actual de la imagen como referencia y moverían los valores de la gasolina y del recorrido junto con este, adicionalmente detectarían cuándo es que el vehículo se queda sin gasolina para activar el botón (que también se movería junto con el vehículo pero sin ser visible) que permite recargar el tanque del vehículo y también detectar el momento en que se llega al destino mediante el valor de los pixeles para que se pueda detener y activar el botón que permite que se regrese al punto de origen. Junto con todo esto, actualiza los vales del consumo de la gasolina y el recorrido del vehículo para que le usuario pueda visualizarlo cada que se mueve el vehículo, que sería a 1km/s.
- **SERIALIZACIÓN DE LOS DATOS:** Este programa requirió que los valores se serializaran para que no se pierdan al cerrar el programa. Por lo

que, para la serialización, se usaron *ObjectInputStream*, *FileInputStream*, *ObjectOutputStream*, entre otros, para guardar o leer el archivo que se generaría o guardaría, se usó una matriz de tipo objeto para guardar este tipo de información y un *ArrayList*, también de tipo objeto, para que se pudiera guardar el historial.

LÓGICA DEL PROGRAMA

➤ UDrive

```
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.FileNotFoundException;
6  import java.io.FileOutputStream;
7  import java.io.ObjectInputStream;
8  import java.io.ObjectOutputStream;
9  import java.util.ArrayList;
10 import java.util.Scanner;
11 import javax.swing.JFileChooser;
12 import javax.swing.JFrame;
13 import javax.swing.JOptionPane;
14 import javax.swing.filechooser.FileNameExtensionFilter;
15
```

Librerías

Las librerías que se usaron en esta clase son necesarias para la creación de la interfaz gráfica, dado a que se trabaja con objetos de este tipo a pesar de no tener poseerla; la serialización de los datos, la lectura de estos valores, ArrayList para guardar los datos serializados del historial.

➤ Variables Globales de la clase UDrive

```
22 public static int ruta[] = new int[3]; //aquí guardar las posi
23 public static boolean libre[] = {true, true, true};
24 public static ArrayList<Linea> historial = new ArrayList<>();
25 public static Vehiculo[] guardado = new Vehiculo[3];
26 public static int ID = 99;
27
```

Las variables de tipo booleano se usan para la verificación de si un vehículo está en uso o un conductor; la de tipo vehículo se usa para la serialización, pues en esta se guardan los valores de las posiciones de los vehículos; el arraylist, del mismo modo se usa para que se pueda guardar la historia que luego se irá a guardar y, la variable llamada ID, se usa para realizar la tabla después de leer el archivo de tipo csv.

➤ Función Main

La función main en esta ocasión , se utiliza para llamar la clase que se encarga de dar la bienvenida y para, si ya hay un historial, leerlo.

```
30 public static void main(String[] args) {  
31     Inicio inicio = new Inicio();  
32     inicio.setLocationRelativeTo(null);  
33     inicio.setResizable(false);  
34     inicio.setVisible(true);  
35     tablear();  
36 }
```

➤ Métodos y Funciones utilizadas

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

- ```
44
45 + public static void leerCSV(JFrame frame) {...37 lines }
82
```

Esta función es la encargada de la lectura del archivo .CSV.

- ```
82  
83 + public static Object leerHistorial() {...12 lines }  
95
```

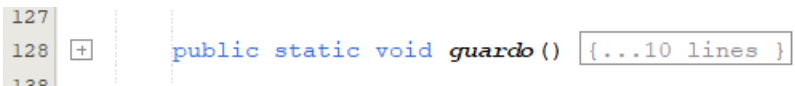
Es función es llamada en la función *tablear()*, y se encarga de la lectura del historial

- ```
93
96 + public static void tablear() {...20 lines }
116
```

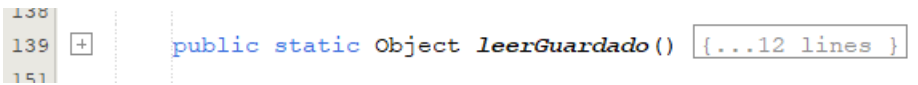
Esta función se encarga de la creación de la tabla a partir de la lectura del historial que se realizó con la lectura de la función anterior.

- ```
117 + public static void serializar() {...10 lines }  
127
```

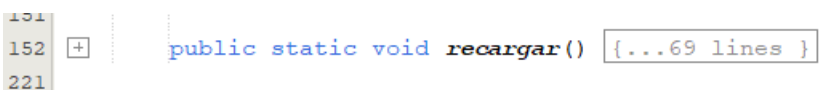
Esta función se encarga de la serialización del historial para su posterior lectura cuando se vuelva a abrir el programa.

-  `127
128 public static void guardar() { ...10 lines }`

Esta función se encarga de guardar los datos de las posiciones de los vehículos y sus relacionados de forma serializada para su posterior lectura.

-  `138
139 public static Object leerGuardado() { ...12 lines }`

Esta función se encarga de leer el archivo serializado que se generó a partir de la función anterior.

-  `151
152 public static void recargar() { ...69 lines }`

Esta función sirve para que, al pulsar el botón de “cargar”, se pueda obtener todos los valores que fueron serializados.

❖ Vehiculo

```
3
4 import java.io.Serializable;
5
```

➤ Librerías

Dado a que esta clase se serializa, entonces se usa este tipo de librería para ese fin.

➤ Variables Globales de la clase Vehiculo

```
12 private double gasolina, distancia, distanciado, trozo;
13 private String destino, inicio, Tl;
14 private int i, pos, l;
15 private boolean br, si, ini, available;
```

Las variables declaradas, sirven para guardar la información de manera ordenada, la cual después será usada para que cargar los datos de la serialización.

➤ Función Vehículo

Es el constructor de la clase.

```
16
17 public Vehiculo(double gasolina, double distancia, double distanciado, S
33
34
```

➤ Métodos y Funciones utilizadas

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

```

36 public double getGasolina() {
37     return gasolina;
38 }
39
40 public void setGasolina(double gasolina) {
41     this.gasolina = gasolina;
42 }
43
44 public double getDistancia() {
45     return distancia;
46 }
47
48 public void setDistancia(double distancia) {
49     this.distancia = distancia;
50 }
51
52 public String getDestino() {
53     return destino;
54 }
55
56 public void setDestino(String destino) {
57     this.destino = destino;
58 }
59
60 public String getInicio() {
61     return inicio;
62 }
63
64 public void setInicio(String inicio) {
65     this.inicio = inicio;
66 }
67
68 public boolean isBr() {
69     return br;
70 }
71
72 public void setBr(boolean br) {
73     this.br = br;
74 }
75
76 /**
77  * @return the i
78  */
79 public int getI() {
80     return i;
81 }

```

Todos los getters y setters de las variables previamente declaradas.

❖ Viajes

```
3 import java.awt.Color;
4 import java.awt.Font;
5 import java.awt.HeadlessException;
6 import java.awt.Image;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9 import java.time.LocalDateTime;
10 import java.time.format.DateTimeFormatter;
11 import javax.swing.*;
```

➤ **Librerías**

En esta clase es la parte principal del programa, por lo que esta, al tener interfaz gráfica, tiene las librerías requeridas de AWT y SWING, además usa la librería *java.time* para obtener la hora local de la computadora.

➤ Variables Globales de la clase Viajes

```

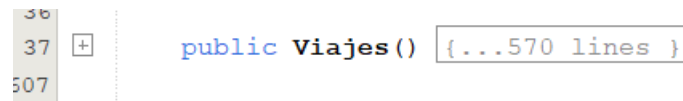
19 public static DateTimeFormatter Formto = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
20 public static Integer lb1, lb13, lb133, lb134, lb114, lb116, lb121, lb122, lb123, lb124, lb125, lb126, lb131, lb132, lb133, lb134, lb135, lb136,
21     LabelImagen1, LabelImagen2, LabelImagen3;
22 public static String d1, d2, d3, r1, r2, r3, r11, r12, r13, r14, r15, r16, r17, r18, r19, r20, r21, r22, r23, r24, r25, r26, r27, r28, r29, r30, r31, r32, r33, r34, r35, r36, r37, r38, r39, r40, r41, r42, r43, r44, r45, r46, r47, r48, r49, r50, r51, r52, r53, r54, r55, r56, r57, r58, r59, r60, r61, r62, r63, r64, r65, r66, r67, r68, r69, r70, r71, r72, r73, r74, r75, r76, r77, r78, r79, r80, r81, r82, r83, r84, r85, r86, r87, r88, r89, r90, r91, r92, r93, r94, r95, r96, r97, r98, r99, r100, r101, r102, r103, r104, r105, r106, r107, r108, r109, r110, r111, r112, r113, r114, r115, r116, r117, r118, r119, r120, r121, r122, r123, r124, r125, r126, r127, r128, r129, r130, r131, r132, r133, r134, r135, r136, r137, r138, r139, r140, r141, r142, r143, r144, r145, r146, r147, r148, r149, r150, r151, r152, r153, r154, r155, r156, r157, r158, r159, r160, r161, r162, r163, r164, r165, r166, r167, r168, r169, r170, r171, r172, r173, r174, r175, r176, r177, r178, r179, r180, r181, r182, r183, r184, r185, r186, r187, r188, r189, r190, r191, r192, r193, r194, r195, r196, r197, r198, r199, r200, r201, r202, r203, r204, r205, r206, r207, r208, r209, r210, r211, r212, r213, r214, r215, r216, r217, r218, r219, r220, r221, r222, r223, r224, r225, r226, r227, r228, r229, r230, r231, r232, r233, r234, r235, r236, r237, r238, r239, r240, r241, r242, r243, r244, r245, r246, r247, r248, r249, r250, r251, r252, r253, r254, r255, r256, r257, r258, r259, r260, r261, r262, r263, r264, r265, r266, r267, r268, r269, r270, r271, r272, r273, r274, r275, r276, r277, r278, r279, r280, r281, r282, r283, r284, r285, r286, r287, r288, r289, r290, r291, r292, r293, r294, r295, r296, r297, r298, r299, r300, r301, r302, r303, r304, r305, r306, r307, r308, r309, r310, r311, r312, r313, r314, r315, r316, r317, r318, r319, r320, r321, r322, r323, r324, r325, r326, r327, r328, r329, r330, r331, r332, r333, r334, r335, r336, r337, r338, r339, r340, r341, r342, r343, r344, r345, r346, r347, r348, r349, r350, r351, r352, r353, r354, r355, r356, r357, r358, r359, r360, r361, r362, r363, r364, r365, r366, r367, r368, r369, r370, r371, r372, r373, r374, r375, r376, r377, r378, r379, r380, r381, r382, r383, r384, r385, r386, r387, r388, r389, r390, r391, r392, r393, r394, r395, r396, r397, r398, r399, r400, r401, r402, r403, r404, r405, r406, r407, r408, r409, r410, r411, r412, r413, r414, r415, r416, r417, r418, r419, r420, r421, r422, r423, r424, r425, r426, r427, r428, r429, r430, r431, r432, r433, r434, r435, r436, r437, r438, r439, r440, r441, r442, r443, r444, r445, r446, r447, r448, r449, r450, r451, r452, r453, r454, r455, r456, r457, r458, r459, r460, r461, r462, r463, r464, r465, r466, r467, r468, r469, r470, r471, r472, r473, r474, r475, r476, r477, r478, r479, r480, r481, r482, r483, r484, r485, r486, r487, r488, r489, r490, r491, r492, r493, r494, r495, r496, r497, r498, r499, r500, r501, r502, r503, r504, r505, r506, r507, r508, r509, r510, r511, r512, r513, r514, r515, r516, r517, r518, r519, r520, r521, r522, r523, r524, r525, r526, r527, r528, r529, r530, r531, r532, r533, r534, r535, r536, r537, r538, r539, r540, r541, r542, r543, r544, r545, r546, r547, r548, r549, r550, r551, r552, r553, r554, r555, r556, r557, r558, r559, r560, r561, r562, r563, r564, r565, r566, r567, r568, r569, r570, r571, r572, r573, r574, r575, r576, r577, r578, r579, r580, r581, r582, r583, r584, r585, r586, r587, r588, r589, r590, r591, r592, r593, r594, r595, r596, r597, r598, r599, r600, r601, r602, r603, r604, r605, r606, r607, r608, r609, r610, r611, r612, r613, r614, r615, r616, r617, r618, r619, r620, r621, r622, r623, r624, r625, r626, r627, r628, r629, r630, r631, r632, r633, r634, r635, r636, r637, r638, r639, r640, r641, r642, r643, r644, r645, r646, r647, r648, r649, r650, r651, r652, r653, r654, r655, r656, r657, r658, r659, r660, r661, r662, r663, r664, r665, r666, r667, r668, r669, r670, r671, r672, r673, r674, r675, r676, r677, r678, r679, r680, r681, r682, r683, r684, r685, r686, r687, r688, r689, r690, r691, r692, r693, r694, r695, r696, r697, r698, r699, r700, r701, r702, r703, r704, r705, r706, r707, r708, r709, r710, r711, r712, r713, r714, r715, r716, r717, r718, r719, r720, r721, r722, r723, r724, r725, r726, r727, r728, r729, r730, r731, r732, r733, r734, r735, r736, r737, r738, r739, r740, r741, r742, r743, r744, r745, r746, r747, r748, r749, r750, r751, r752, r753, r754, r755, r756, r757, r758, r759, r760, r761, r762, r763, r764, r765, r766, r767, r768, r769, r770, r771, r772, r773, r774, r775, r776, r777, r778, r779, r780, r781, r782, r783, r784, r785, r786, r787, r788, r789, r790, r791, r792, r793, r794, r795, r796, r797, r798, r799, r800, r801, r802, r803, r804, r805, r806, r807, r808, r809, r810, r811, r812, r813, r814, r815, r81
```

De forma muy general, las variables que se utilizan a continuación se usan para la interfaz gráfica del programa, la actualización de los datos en tiempo real y muchas banderas para la validación de datos, eventos y posiciones.

➤ Función Viajes

Esta función simplemente genera las posiciones, vehículos y cantidades y valores dependiendo de los que se haya escogido en las pestañas anteriores por medio de la lectura del conjunto de las

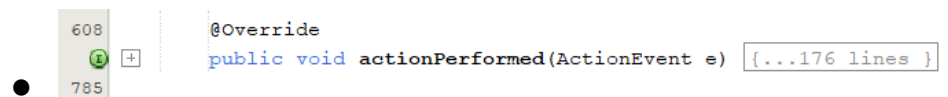
variables previamente declaradas, que será modificadas a través de las distintas clases y funciones.



```
36  
37 public Viajes() {...570 lines }  
507
```

➤ Métodos y Funciones utilizadas

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



```
608  
609 @Override  
610 public void actionPerformed(ActionEvent e) {...176 lines }  
785
```

Esta es la única función y se encarga de las acciones de los botones al ser presionados.

❖ Generar

```
3  import java.awt.Color;
4  import java.awt.Font;
5  import javax.swing.JOptionPane;
6
```

➤ Librerías

Esta al ser una ventana, simplemente posee las librerías necesarias para la ejecución del entorno gráfico.

➤ Variables Globales de la clase Generar


```
15  //
16  public static int l1=-1;
17  public static int l2=-1;
18  public static int l3=-1;
19  public static int l11, l12, l13;
20  public static int a=-1;
21  public static int b=-1;
22  public static int c=-1;
```

```
320  // Variables declaration - do not modify
321  private javax.swing.JButton jButton1;
322  private javax.swing.JButton jButton2;
323  private javax.swing.JComboBox<String> jComboBox1;
324  private javax.swing.JComboBox<String> jComboBox2;
325  private javax.swing.JComboBox<String> jComboBox3;
326  private javax.swing.JLabel jLabel1;
327  private javax.swing.JLabel jLabel2;
328  private javax.swing.JLabel jLabel3;
329  private javax.swing.JLabel jLabel4;
330  // End of variables declaration
```

Las variables de esta clase se encargan de guardar los datos que se obtienen de los *Combobox* para luego modificar los valores de los vectores que se encuentran en la clase *Udrive* para así modificar el tipo de vehículo y la disponibilidad de estos y de los conductores. Y por último estas los botones y esos elementos de la interfaz gráfica.


➤ Función Generar

Esta función simplemente fue usada para que se pueda indicar cuando es que no hay disponibilidad de conductores y también para generar las *combobox* para generar el viaje a partir de la tabla hecha con el archivo *.csv*-


25  `public Generar() { ...31 lines }`

➤ Métodos y Funciones utilizadas

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

-  `private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { ...78 lines }`

Esta es la función que se ejecuta cuando el botón de generar se presiona. Esta guarda todos los valores seleccionados y modifica las variables que se tengan que modificar para que se logre genera el viaje con el vehículo escogido.

-  `private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) { ...5 lines }`

Simplemente cierra la ventana cuando se presiona el botón que se encarga de eso.

❖ Historial

```
4 import javax.swing.table.DefaultTableModel;
```

➤ Librerías

Como en esta ventana se presenta la tabla del historial de todos los viajes, entonces necesita una tabla, que se crea con la librería que se muestra ahí.

➤ Variables Globales de la clase Historial

```
10 public class Historial extends javax.swing.JFrame {
11     public static DefaultTableModel h=new DefaultTableModel();
12     /**
13
14     // Variables declaration - do not modify
15     private javax.swing.JButton jButton1;
16     private javax.swing.JLabel jLabel1;
17     private javax.swing.JScrollPane jScrollPane1;
18     private javax.swing.JTable jTable1;
19     // End of variables declaration
20 }
```

Simplemente es una DefaultTableModel para crear la tabla y poder modificarla, además de los elementos de la interfaz gráfica.

➤ Función Historial

Solo genera la tabla y sus encabezados a partir de los datos que se van guardando conforme se completan los viajes de las rutas

```
15 public Historial() { ...9 lines }
```

➤ Métodos y Funciones utilizadas

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

```
97 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
98     this.dispose();
99 }
```

Cierra la pestaña cuando se presiona el botón que fue creado para hacer eso.

❖ Inicio

```
7  L  */
8      public class Inicio extends javax.swing.JFrame {
9
```

➤ Librerías

No posee ni una librería en especial además de ser una extensión de `javax.swing.JFrame`.

➤ Variables Globales de la clase Inicio

```
109      // Variables declaration - do not modify
110      private javax.swing.JButton jButton1;
111      private javax.swing.JLabel jLabel1;
112      // End of variables declaration
```

Simplemente es un label y un JButton.

➤ Función Inicio

Solo inicia los componentes de la ventana que serían un label y su botón.

```
13  [ ] public Inicio() {
14      [ ]     initComponents();
15      [ ] }
```

➤ Métodos y Funciones utilizadas

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

```
71  [ ] private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
72      [ ]     // TODO add your handling code here:
73      [ ]     Menu menu=new Menu();
74      [ ]     menu.setLocationRelativeTo(null);
75      [ ]     menu.setResizable(false);
76      [ ]     menu.setVisible(true);
77      [ ]     this.dispose();
```

Se encarga de instanciar la clase *menu*, que sería la pantalla principal del programa.

❖ Línea

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

➤ Librerías

Esta también es una clase que se usará para guardar datos de manera serializada, por lo que tiene la librería requerida para este fin.

➤ Variables Globales de la clase Inicio

```
10  
11 private String fechaf, fechai, distancia, vehiculo, gasolina;  
12
```

Posee las variables necesarias para guardar los datos de la tabla de historial.

➤ Función Línea

Es el constructor de la clase.

```
13  
14 public Línea(String fechaf, String fechai, String distancia, String vehiculo, String gasolina) {  
15     this.fechaf = fechaf;  
16     this.fechai = fechai;  
17     this.distancia = distancia;  
18     this.vehiculo = vehiculo;  
19     this.gasolina = gasolina;  
20 }
```

➤ Métodos y Funciones utilizadas

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

```

21 + /**...3 lines */
24 + public String getFechaf() {...3 lines }
27
28 + /**...3 lines */
31 + public void setFechaf(String fechaf) {...3 lines }
34
35 + /**...3 lines */
38 + public String getFechai() {...3 lines }
41
42 + /**...3 lines */
45 + public void setFechai(String fechai) {...3 lines }
48
49 + /**...3 lines */
52 + public String getDistancia() {...3 lines }
55
56 + /**...3 lines */
59 + public void setDistancia(String distancia) {...3 lines }
62
63 + /**...3 lines */
66 + public String getVehiculo() {...3 lines }
69
70 + /**...3 lines */
73 + public void setVehiculo(String vehiculo) {...3 lines }
76
77 + /**...3 lines */
80 + public String getGasolina() {...3 lines }
83
84 + /**...3 lines */
87 + public void setGasolina(String gasolina) {...3 lines }
90

```

Los getters y los setters.

❖ Menu

```
3
4  import javax.swing.table.DefaultTableModel;
5
6  /**
```

➤ Librerías

Solo es la librería para generar la tabla en donde se mostrarán los viajes.

➤ Variables Globales de la clase Inicio

```
10  public class Menu extends javax.swing.JFrame {
11      public static DefaultTableModel dt=new DefaultTableModel();
12      /**
13
14      // Variables declaration - do not modify
15      private javax.swing.JButton jButton1;
16      private javax.swing.JButton jButton3;
17      private javax.swing.JButton jButton4;
18      private javax.swing.JButton jButton5;
19      private javax.swing.JScrollPane jScrollPane1;
20      private javax.swing.JTable jTable1;
21      // End of variables declaration
22  }
```

Simplemente es el modelo de la tabla y los elementos dentro de la ventana.

➤ Función Menu

Crea la tabla con sus identificadores y los valores de .csv que se obtuvieron previamente mediante las distintas funciones.

```
15  public Menu() {
24  }
```

➤ Métodos y Funciones utilizadas

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

```

130
131
132     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
133         UDrive.leerCSV(this);
134         if (UDrive.ID >= 100) {
135             jButton3.setEnabled(true);
136         }
137     }

```

Es la ejecución del botón que se encargará de leer el .csv y de activar el botón para que se pueda ir a la ventana de viajes.

```

139
140     private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) { ...13 lines }
141
142

```

Este se encarga de instanciar la ventana que permite generar los viajes.

```

153
154     private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) { ...12 lines }
155
156

```

Este permite instanciar la ventana en donde aparecerán todos los viajes, las rutas, etc.

❖ Ruta1, Ruta2, Ruta3

```
9  L  */
10  public class Ruta1 extends Thread{
    .
    .
    .
8   public class Ruta2 extends Thread{
    .
    .
    .
8   public class Ruta3 extends Thread{
    .
    .
    .
```

➤ Librerías

No poseen ni una librería en especial, pero son hilos.

➤ Variables Globales de la clase Ruta1, Ruta2, Ruta3

```
10  public class Ruta1 extends Thread{
11      Viajes viaje;
12      private boolean b1=true;
    .
    .
    .
8   public class Ruta2 extends Thread{
9      Viajes viaje;
10     private boolean b1=true;
    .
    .
    .
8   public class Ruta3 extends Thread{
9       Viajes viaje;
10      private boolean b1=true;
```

Variables para modificar la pestaña por medio de los hilos y para poder finalizar la ejecución del hilo.

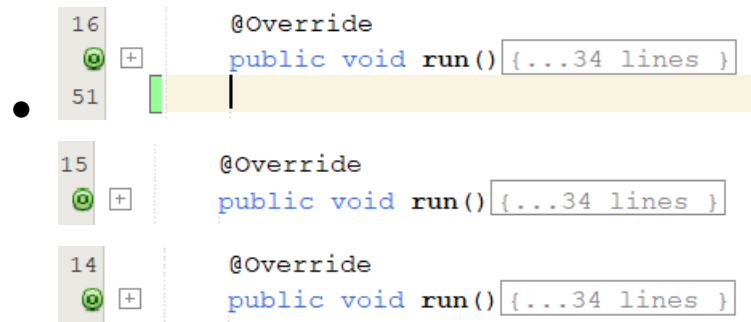
➤ Función Ruta1, Ruta2, Ruta3

Simplemente es el constructor de las clases.

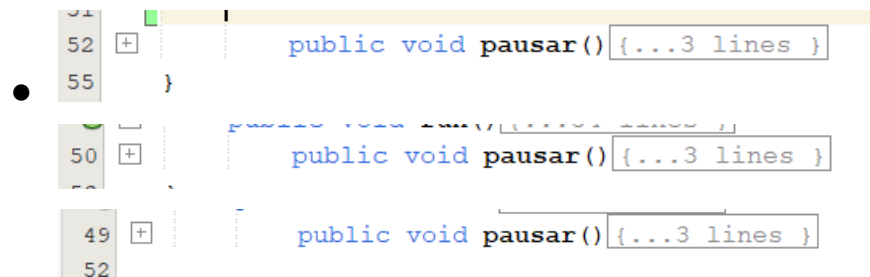
```
13  [ ] public Ruta1(Viajes viaje) {
14      [ ]     this.viaje = viaje;
15      [ ] }
    .
    .
    .
12  [ ] public Ruta2(Viajes viaje) {
13      [ ]     this.viaje = viaje;
14      [ ] }
    .
    .
    .
11  [ ] public Ruta3(Viajes viaje) {
12      [ ]     this.viaje = viaje;
13      [ ] }
```

➤ Métodos y Funciones utilizadas

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



Es el método que tiene cada una de las clases que se encarga de la ejecución de los hilos que, después de un segundo, mueve el vehículo una posición proporcional a la distancia que debe recorrer en realidad.



Estos simplemente modifican el valor de la variables que se mostró previamente y hacen que se termine la ejecución del hilo.

❖ Ruta11, Ruta22, Ruta33

```
9  //
10 public class Ruta11 extends Thread{
11     ...
12 }
10 public class Ruta22 extends Thread{
11     ...
12 }
9  //
10 public class Ruta33 extends Thread{
11     ...
12 }
```

➤ Librerías

No poseen librería en especial, pero son hilos.

➤ Variables Globales de la clase Inicio

```
10 public class Ruta11 extends Thread{
11     Viajes viaje;
12     private boolean b1=true;
13 }
10 public class Ruta22 extends Thread{
11     Viajes viaje;
12     private boolean b1=true;
13 }
10 public class Ruta33 extends Thread{
11     Viajes viaje;
12     private boolean b1=true;
13 }
```

Solamente son las variables para la instanciación de la ventana viajes y un booleano para la finalización del hilo.

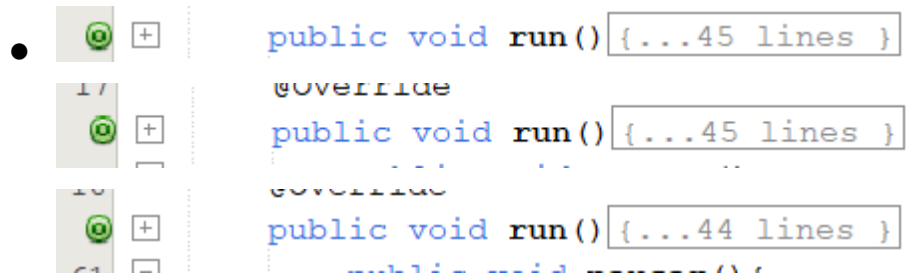
➤ Función Inicio

Es el constructor de la clase.

```
13 public Ruta11(Viajes viaje) {
14     this.viaje = viaje;
15 }
14 public Ruta22(Viajes viaje) {
15     this.viaje = viaje;
16 }
13 public Ruta33(Viajes viaje) {
14     this.viaje = viaje;
15 }
```

➤ Métodos y Funciones utilizadas

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



Esta es la función del hilo que se encarga de correrlo, modificando los datos y posiciones de los valores de la clase *viaje*. Esta se diferencia de la de Ruta1, Ruta2 y Ruta3 en que va en sentido contrario y que, al finalizar, serializa y guarda los datos en la tabla de historial.

Con todo esto, finaliza el manual técnico de UDrive, se espera que haya sido de gran ayuda y que permita que se pueda comprender más a profundidad el funcionamiento del programa con la ayuda de este programa. :)