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, 21 DE MARZO DEL 2,024

# ÍNDICE

[**ÍNDICE 1**](#_k1cgegpfdrrl)

[**INTRODUCCIÓN 1**](#_574nc6ehlpl)

[**OBJETIVOS 1**](#_3bsxbi49eln7)

[1. GENERAL 1](#_38w1k6cukb5i)

[2. ESPECÍFICOS 1](#_poh928tpbmga)

[**ALCANCES DEL SISTEMA 1**](#_twvsj8hvwtb9)

[**ESPECIFICACIÓN TÉCNICA 1**](#_hxtoxxv43krt)

[● REQUISITOS DE HARDWARE 1](#_mfriwpcdfh1l)

[● REQUISITOS DE SOFTWARE 1](#_8h6kxoigku56)

[**DESCRIPCIÓN DE LA SOLUCIÓN 2**](#_14axd4p9n3ry)

[**LÓGICA DEL PROGRAMA 2**](#_ng9quldintk8)

[❖ NOMBRE DE LA CLASE  
Captura de las librerías usadas 2](#_xush55kxig5)

[➢ Librerías 2](#_h84kbu3eveip)

[➢ Variables Globales de la clase \_(El nombre de su clase actual) 3](#_jmrvbw5ti5o0)

[➢ Función Main 3](#_lg5om9s2qe42)

[➢ Métodos y Funciones utilizadas 3](#_xbejjcbelcz7)

# 

# 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

## GENERAL

* 1. Proporcionar información técnica del programa al lector.

## ESPECÍFICOS

* 1. Objetivo 1: Especificar el funcionamiento y librerías de cada una de las clases del programa.
  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 ARCHVIO .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 pixel 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

## NOMBRE DE LA CLASE *Captura de las librerías usadas*

### Librerías

Breve descripción del funcionamiento de cada librería usada y breve descripción de para qué la usó

### Variables Globales de la clase \_(El nombre de su clase actual)

*Captura de sus variables globales*

Breve descripción generalizada de el uso de sus variables globales (No es necesario especificar para qué se usó cada una)

### Función Main

Breve descripción de para qué usaron la función main

*Captura del código de su función main*

### Métodos y Funciones utilizadas

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

* *Captura del nombre de su función que muestre el número de línea*  
  Breve descripción del uso de la función o para qué sirve
*   
  Breve descripción del uso de la función o para qué sirve