**Text

Description automatically generated with medium confidence**

**Text

Description automatically generated with medium confidence**

**Module Code & Module Title**

**CS4001NT Programming**

**Assessment Weightage & Type**

**30% Individual Coursework**

**Year and Semester**

**2022-23 Spring 2**

**Student Name: Geenesh Acharya**

**Group: L1C4**

**London Met ID: GEA0374@my.londonmet.ac.uk**

**College ID: np50cp4s220023@iic.edu.np**

**Assignment Due Date: August 05, 2022**

**Assignment Submission Date: July 30, 2022**

***I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarde******d.***

Table of Contents

[1. Introduction to Java 5](#_Toc110605110)

[1.1 Objectives of JAVA 6](#_Toc110605111)

[1.2 Tools used for this coursework 6](#_Toc110605112)

[**1.** **Visual Studio Code** 6](#_Toc110605113)

[**2.** **Blue J** 7](#_Toc110605114)

[**3.** **Google Chrome** 7](#_Toc110605115)

[4. Class Diagram 8](#_Toc110605116)

[4.2 Auto-Rickshaw Class (class of vehicle class): 9](#_Toc110605117)

[4.3 Electric Scooter Class (subclass of vehicle class): 10](#_Toc110605118)

[4.4 Transport GUI 11](#_Toc110605119)

[5. Relation between Classes 17](#_Toc110605120)

[6. Pseudo Code 18](#_Toc110605121)

[6.1 Pseudo-Code of Transport GUI 18](#_Toc110605122)

[7. Method Description 67](#_Toc110605123)

[8. Testing 72](#_Toc110605124)

[8.1 Test 01 72](#_Toc110605125)

[8.2 Test 02 75](#_Toc110605126)

[8.3 Test 3 80](#_Toc110605127)

[9. Error 87](#_Toc110605128)

[9.1 Syntax Errors 87](#_Toc110605129)

[9.2 Logic Error 88](#_Toc110605130)

[9.3 Semantics Error 91](#_Toc110605131)

[10. Conclusion 92](#_Toc110605132)

[References 93](#_Toc110605133)

[11. Appendix 95](#_Toc110605134)

**LIST OF TABLES**

[**Table 1 Class Diagram of Vehicle** 8](#_Toc110605155)

[**Table 2 Class Diagram of Auto-Rickshaw** 9](#_Toc110605156)

[**Table 3 Class Diagram of Electric Scooter** 10](#_Toc110605157)

[**Table 4 Class Diagram of TransportGUI** 16](#_Toc110605158)

[**Table 5 Testing to compile and run using CMD** 72](#_Toc110605159)

[**Table 6 Testing the autorickshaw and electric scooter buttons.** 76](#_Toc110605160)

**LIST OF FIGURES**

[**Figure 1 Detail image of JAVA** (Johari, 2021) 6](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605190)

[**Figure 2 Detail image of JAVA** (Johari, 2021) 6](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605191)

[**Figure 3 Image of Visual Studio Code** 7](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605192)

[**Figure 4 Image of Visual Studio Code** 7](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605193)

[**Figure 5 Image of Blue J** 8](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605194)

[**Figure 6 Image of Blue J** 8](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605195)

[**Figure 7 Image of Google Chrome** 8](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605196)

[**Figure 8 Image of Google Chrome** 8](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605197)

[**Figure 9 Image of Relation between Classes** 18](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605198)

[**Figure 10 Compiling and Running in Command Prompt** 74](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605199)

[**Figure 11 Image of TransportGUI** 75](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605200)

[**Figure 12 Image of Booked Auto Rickshaw** 78](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605201)

[**Figure 13 Empty Text field of Auto Rickshaw** 82](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605202)

[**Figure 14 Image of Invalid Integer in Add Auto Rickshaw** 83](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605203)

[**Figure 15 Image of Syntax Error** 88](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605204)

[**Figure 16 Image of Syntax Error** 88](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605205)

[**Figure 17 Correction of Syntax Error** 88](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605206)

[**Figure 18 Swapping the Clear Button** 89](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605207)

[**Figure 19 Logic Error not clearing text field in image** 90](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605208)

[**Figure 20 Correct Code of Clear Button** 90](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605209)

[**Figure 21The Clear Button is operational.** 91](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605210)

[**Figure 22 Image of Semantics Error** 92](#_Toc110605211)

[**Figure 23 Correction of Semantic Error** 92](https://d.docs.live.net/b31e21fda715343a/Documents/220023%20GEENESH%20ACHARYA.docx#_Toc110605212)

# Introduction to Java

In this project, I designed a program based on software that stores information about the Auto Rickshaw and Electric-Scooter. This project is all about coding and creating a software with evidence. We must test all codes to see if they display the required output.

It is an object-oriented language like C++, but with more advanced and simple features. This language is available to everyone and is compatible with all systems. Java, a cross- platform object- oriented programming language, was first released in 1995 by Sun Microsystems. Today, a wide range of products, including games, social networking applications, audio, and video programs, etc., require Java to function. Even while the most of modern Java apps now include the Java runtime, many other programs and even some websites still need a desktop Java installation to function. It might be referred to be an application development platform for computers. (Johari, 2021)

A picture containing diagram

Description automatically generatedLet's look at how certain technologies use Java as the core building block for all its features.

**Figure 1 Detail image of JAVA** (Johari, 2021)

**Figure 2 Detail image of JAVA** (Johari, 2021)

## Objectives of JAVA

The objectives of Java programming languages are as follow:

* It is simple to learn because it is built on the C++ programming language.
* It is an independent programming language that follows towards the "Write once, Run anywhere" concept.
* It is both an object-oriented and class-based programming language.
* It is a concurrent programming language that allows you to run many statements continuously rather than sequentially.

## Tools used for this coursework

In this coursework, I created classes, methods, declarations, and validation for buttons, among other things, using the Java programming language. The following list of resources I utilized in this coursework is followed by an explanation.

### **Visual Studio Code**

In simple words, Visual Studio Code is a code editor. "A free editor that helps the programmer create code, aids in debugging, and corrects the code using the Intelli-Sense method," is what Visual Studio Code is described as. In plain English, it makes it easier for people to develop code.

Although many claim that it is only partially an editor and an IDE, the choice is ultimately up to the developers. Any program or piece of software that we can see, or use relies on background code to function. Coding was traditionally done in conventional editors or even in simple editors like notepad! The coders were previously given some minimal assistance by these editors. (Pedamkar, 2022)

**Figure 3 Image of Visual Studio Code**

**Figure 4 Image of Visual Studio Code**

### **Blue J**

BlueJ is a Windows-based Java Development Kit platform (JDK). As an alternative for Blue, it is a free Java environment that was founded in 1999 at Monash University in Australia by Michael Kolling and John Rosenberg. Before installing BlueJ, you must first install JDK or a higher version. It was created to assist with object-oriented programming education and learning. The creation and testing of the things is interactive. The BlueJ user interface is more straightforward than the interface of most professional IDEs. (geeksforgeeks, 2020)

**Figure 5 Image of Blue J**

**Figure 6 Image of Blue J**

### **Graphical user interface, application Description automatically generatedGoogle Chrome**

The open-source Google Chrome browser may be used to execute web-based applications and access the Internet. The open-source Chromium project serves as the foundation for the Google Chrome web browser. Google launched Chrome in 2008 and updates it frequently. The operating systems Windows, Mac OS X, Linux, Android, and iOS all support it. The Google Chrome browser approaches Web security through sandboxing. Because each open website is operated as a separate process, it is possible to stop harmful code from one page from spreading to others (or the computer operating system at large). (Steele, 2013)

**Figure 7 Image of Google Chrome**

**Figure 8 Image of Google Chrome**

# Class Diagram

Class diagrams represent the static structure of a system and are fundamental to the object modeling process. You may use a single class diagram to represent an entire system, or you can use several class diagrams to represent the various parts of a system, depending on how complicated the system. The class diagrams are the system or subsystem's blueprints. Class diagrams are useful for modeling the system's constituent parts, showing the relationships among them, and describing the functions and services that each provide. (Anon., 2021) The class diagram I created for this assignment is as follows:

* 1. **Vehicle class (Parent Class)**

|  |
| --- |
| **Vehicle Class** |
| **-Vehicle ID: int**  **-Vehicle name: String**  **-Vehicle Weight: String**  **-Vehicle Color: String**  **-Vehicle speed: String** |
| **+Vehicle (int vehicleID, String vehicleName, String vehicleWeight, String vehicleColor, String vehicleSpeed)**  **+getVehicleID():int**  **+getVehicleName():String +getVehicleWeight():String +getVehicleColor():String +getVehicleSpeed():String +setvehicleSpeed(String +setvehicleSpeed):Void +setvehicleColor(String vehicleColor):Void +display():void** |

**Table 1 Class Diagram of Vehicle**

## Auto-Rickshaw Class (class of vehicle class):

|  |
| --- |
| **Auto-Rickshaw** |
| **-Engine Displacement: int**  **-Torque: String**  **-Number of Seats: int**  **-Fuel Tank Capacity: int**  **-Ground Clearance: String**  **-Charge Amount: int**  **-Booked Date: String**  **-isBooked: boolean** |
| **+AutoRickshaw (int vehicleId , String vehicleName , String torque, String**  **vehicleWeight, String vehicleColor, String vehicleSpeed, String groundClearance, int engineDisplacement)**  **+getengineDisplacement(): int**  **+gettorque(); String**  **+getNumberOfSeats(): int**  **+getFuelTankCapacity(): int**  **+getChargeAmout(): int**  **+getBookedData(): String**  **+getIsBooked(): boolean**  **+getgroundClearance(): String**  **+setChargeAmount(int chargeAmount): Void**  **+setNumberOfSeat(int numberOfSeats): void**  **+booking(String bookedDate, int chargeAmount, int numberOfSeat): void**  **+display(): void** |

**Table 2 Class Diagram of Auto-Rickshaw**

## Electric Scooter Class (subclass of vehicle class):

|  |
| --- |
| **Electric-Scooter Class** |
| **-Range: int**  **-Battery Capacity: int**  **-Price: int**  **-Charging Time: String**  **-Brand: String**  **-Mileage: String**  **-has Purchased: boolean**  **-has Sold: boolean** |
| **+ElectricScooter(String vehicleSpeed, String vehileWeight, String vehicleName, int vechileID,int batteryCapacity, String VehicleColor) +getRange(): int**  **+getbatteryCapacity(): Int**  **+getPrice():int**  **+getChargingTime(): String**  **+setBrand(String brand): void**  **+purchasedScooter(String brand, int price, String chargingTime,String mileage, int range):void**  **+display (): void** |

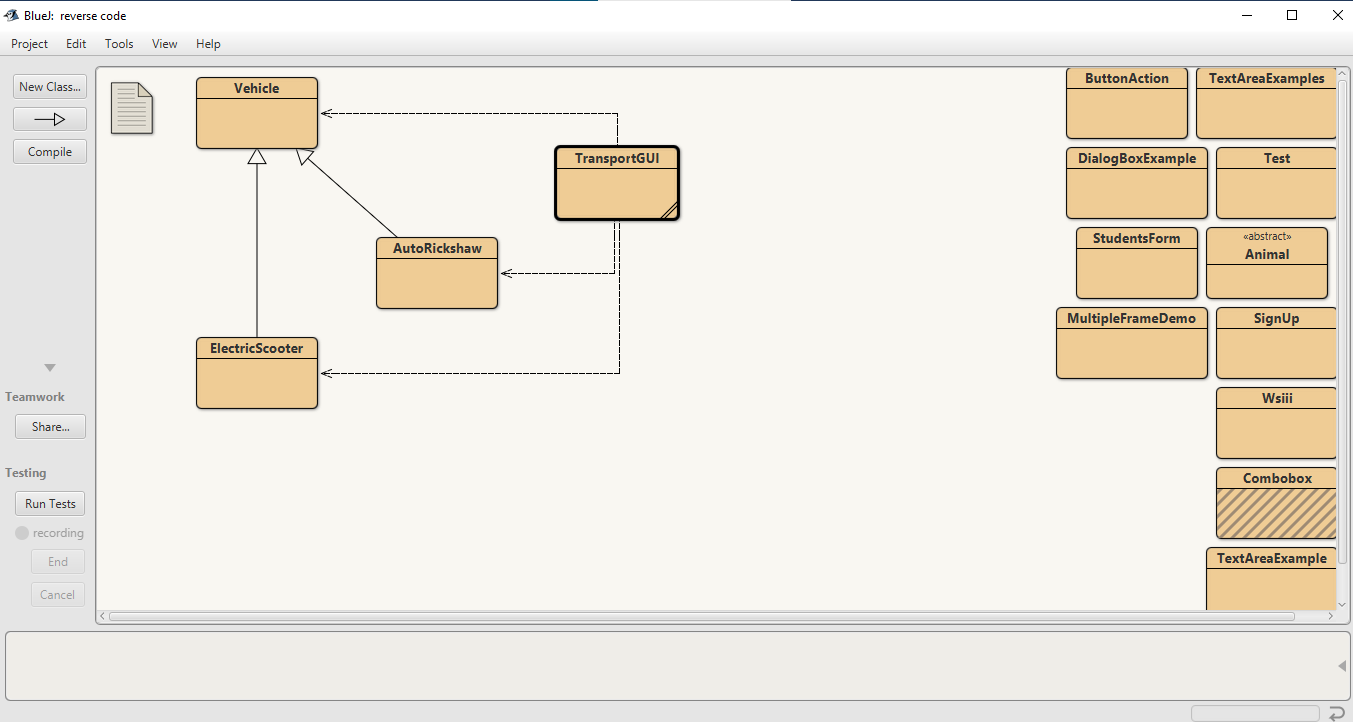
**Table 3 Class Diagram of Electric Scooter**

## Transport GUI

|  |
| --- |
| **Transport GUI** |
| **-frame1: JFrame**  **-frame 2: JFrame**  **-frame 3: JFrame**  **-panel1: JPanel**  **-panel2: JPanel**  **-panel3: JPanel**  **-panel4: JPanel**  **-panel5: JPanel**  **-embarkBtn: JButton**  **-embarkBtn1: JButton**  **-addAutoRickshawBtn: JButton**  **-bookAutoRickshawBtn: JButton**  **-backBtn: JButton**  **-displayBtn: JButton**  **-clearBtn: JButton**  **-addElectricScooterBtn: JButton**  **-purchaseScooterBtn: JButton**  **-sellScooterBtn: JButton**  **-back1Btn: JButton**  **-displayBtn1: JButton**  **-clear1Btn: JButton**  **-title: JLabel**  **-titleAuto: JLabel**  **-titleAuto1: JLabel**  **-titleAuto2: JLabel**  **-titleElectric: JLabel**  **- vehicleID: JLabel**  **-vehicleColor: JLabel**  **-torque: JLabel**  **-vehicleName: JLabel**  **-vehicleSpeed: JLabel**  **-fuelTankCapacity: JLabel**  **-vehicleWeight: JLabel**  **-engineDisplacement: JLabel**  **-groundClearance: JLabel**  **-vehicleId1: JLabel**  **-noOfSeats: JLabel**  **-chargeAmount: JLabel**  **-vehicleID2: JLabel**  **-vehicleWeight1: JLabel**  **-vehicleName1: JLabel**  **-vehicleColor1: JLabel**  **-vehicleSpeed1: JLabel**  **-batteryCapacity1: JLabel**  **vehicleID5: JLabel**  **-mileage: JLabel**  **-chargingTime: JLabel**  **-brand: JLabel**  **-range: JLabel**  **-costPrice: JLabel**  **-vehicleID4: JLabel**  **-sellingPrice: JLabel**  **-bookDate: JLabel**  **-vehicleIDtf: JTextField**  **-vehicleColortf: JTextField**  **-torquetf: JTextField**  **-vehicleNametf: JTextField**  **-vehicleSpeedtf: JTextField**  **-fuelTankCapacitytf: JTextField**  **-vehicleWeighttf: JTextField**  **-engineDisplacementtf: JTextField**  **-groundClearancetf: JTextField**  **-vehicleIdtf1: JTextField**  **-noOfSeatstf: JTextField**  **-chargeAmounttf: JTextField**  **-vehicleNametf1: JTextField**  **-vehicleIDtf2: JTextField**  **-vehicleWeighttf1: JTextField**  **-vehicleColortf1: JTextField**  **-vehicleSpeedtf1: JTextField**  **-batteryCapacitytf1: JTextField**  **-vehicleIDtf5: JTextField**  **-mileagetf: JTextField**  **-chargingTimetf: JTextField**  **-brandtf: JTextField**  **rangetf: JTextField**  **-costPricetf: JTextField**  **-vehicleIDtf4: JTextField**  **-sellingPricetf: JTextField**  **-year: JComboBox**  **-month: JComboBox**  **-day: JComboBox** |
| **+** **list: ArrayList<Vehicle>**  **+TransportGUI**  **+INVALID: int**  **+EMPTY: int**  **+getAutoVehicleIdAdd(): int**  **+getAutoVehicleName(): String**  **+getAutoVehicleWeight(): String**  **+getAutoVehicleColor(): String**  **+getAutoVehicleSpeed():String**  **+getEngineDisplacement(): int**  **+getAutoFuelTankCapacity(): int**  **+getAutoGroundClearance(): String**  **+getAutoTorque(): String**  **+getAutoVehicleId1Book(): int**  **+getAutoNoOfSeat(): int**  **+getAutoChargeAmount(): int**  **+getYear(): String**  **+getMonth(): string**  **+getday(): string**  **+getBookedDate(): string**  **+getAutoVehicleId2(): int**  **+getAutovehicleID2(): int**  **+getAutoVehicleName1(): string**  **+getAutoVehicleColor1(): string**  **+getAutoVehicleWeight1(): string**  **+getAutoVehicleSpeed1(): string**  **+getAutoBatteryCapacity(): int**  **+getAutoVehicleId3(): int**  **+getAutoMileage(): string**  **+getAutoBrand(): string**  **+getAutoRange(): int**  **+getAutoCostPrice(): int**  **+getAutoChargingTime():string**  **+getAutoVehicleID5(): int**  **+getAutoSellingPrice(): int**  **+unique(int ID): boolean**  **+addAuto() : void**  **+bookAuto(): void**  **+addElectric(): void**  **+purchaseElectric(): void**  **+sellElectric(): void** |

**Table 4 Class Diagram of TransportGUI**

# Relation between Classes

A schematic should appear in this screenshot. Each one of the diagram’s-colored rectangles corresponds to a class in our course work. We have courses with names like Vehicle, Electric Scooter, TransportGUI, and Auto Rickshaw.

**Figure 9 Image of Relation between Classes**

# Pseudo Code

The term "pseudo code" is widely used in algorithm-based professions such as programming. It gives the programmer the ability to see how an algorithm is applied. Simply defined, it is a manufactured representation of an algorithm. Because pseudo codes may be understood by programmers of any expertise level, they are widely employed to represent algorithms. Pseudocode is a phrase used to describe fake code or a representation of code that even a layperson with rudimentary programming knowledge may comprehend. (Geeksforgeeks, 2021)

## Pseudo-Code of Transport GUI

**CREATE** Class TransportGUI

**DECLEAR JFrame** frame 1, frame 2, frame 3

**DECLEAR JPanel** panel 1, panel 2, panel 3, panel 4, panel 5

**DECLEAR JButton** embarkBtn, embarkBtn1, addAutoRickshawBtn, bookAutoRickshawBtn, backBtn, displayBtn, clearBtn , addElectricScooterBtn, purchaseScooterBtn, sellScooterBtn, back1Btn, displayBtn1, clear1Btn

**DECLEAR JLabel** title, titleAuto, titleAuto1, titleAuto2, titleElectric, vehicleID, vehicleColor, torque, vehicleName, vehicleSpeed, fuelTankCapacity, vehicleWeight, engineDisplacement, groundClearance, vehicleId1, noOfSeats, chargeAmount, vehicleID2, vehicleWeight1, vehicleName1, vehicleColor1, vehicleSpeed1, batteryCapacity1, vehicleID5, mileage, chargingTime, brand, range, costPrice, vehicleID4, sellingPrice, bookDate

**DECLEAR JTextField** vehicleIDtf, vehicleColortf, torquetf, vehicleNametf, vehicleSpeedtf, fuelTankCapacitytf, vehicleWeighttf, engineDisplacementtf, groundClearancetf, vehicleIdtf1, noOfSeatstf, chargeAmounttf, vehicleNametf1, vehicleIDtf2, vehicleWeighttf1, vehicleColortf1, vehicleSpeedtf1, batteryCapacitytf1, vehicleIDtf5, mileagetf, chargingTimetf, brandtf, rangetf, costPricetf, vehicleIDtf4, sellingPricetf

**DECLEAR JComboBox** year, month , day

**INITIALIZE ARRAY LIST**

**INITIALIZE** frame 1

**SET SIZE** frame 1

**SET DEFULT CLOSE OPERATION** frame 1

**GET CONTENT PANE** frame 1

**SET LAYOUT** frame 1

**SET LOCATION RELATIVE TO NULL** frame 1

**INITILIZE** title font

**INITIALIZE** titleJLabel

**SET** foregroundtitle

**SET** fonttitle

**SET** bound title

**ADD** title to frame 1

**INITIALIZE** title AutoJLabel

**SET FONT** title Auto

**SET** bounds title Auto

**ADD** title Auto to frame 1

**INITIALIZE** title Electric JLabel

**SET FONT** title Electric

**SET** Bounds title Electric

**ADD** title Electric to frame 1

**INITIALIZE** embark Btn

**SET** bounds embark Btn

**SET** focus Painted embark Btn

**SET** background embark Btn

**SET** foreground embark Btn

**SET** border embark Btn

**ADD** embarkBtn to frame 1

**INITIALIZE** embark Btn1

**SET** bounds embark Btn1

**SET** focus Painted embark Btn1

**SET** background embark Btn1

**SET** foreground embark Btn1

**SET** border embark Btn1

**ADD** embarkBtn1 to frame 1

**SET** frame 1 visible

**INITIALIZE** frame 2

**SET** bounds frame 2

**SET** default Close Operation frame 2

**GET** content Pane frame 2

**SET** layout frame 2

**SET** location relative to frame 2

**INITIALIZE** title Auto 1

**SET** foreground title Auto 1

**SET** font title Auto 1

**SET** bounds title Auto 1

**ADD** title Auto 1 to frame 2

**INITIALIZE** panel 1

**SET** bounds panel 1

**SET** background panel 1

**CREATE** object add Auto Title panel 1

**SET TITLE JUSTIFICATION add** AutoTitle

**SET TITLE FONT** add Auto Title

**SET TITLE COLOR** add Auto Titte

**SET BORDER** add Auto Title

**ADD** panel 1 to frame 2

**SET LAYOUT** panel 1

**INITILIZE** vehicle Name

**ADD** vehicle Name to panel1

**SET BOUNDS** to vehicle Name

**INITILIZE** vehicle Name tf

**SET BOUNDS** vehicle Name tf

**SET BACKGROUND** vehicle Name tf

**SET BORDER** vehicle Name tf

**ADD** vehicle Name tf to Panel 1

**INITILIZE** vehicle ID

**ADD** vehicle ID to panel 1

**SET BOUNDS** vehicle ID

**INITILIZE** vehicle ID tf

**SET BOUNDS** vehicle ID tf

**SET BACKGROUND** vehicle ID tf

**SET BORDER** vehicle ID tf

**ADD** vehicle ID tf to panel 1

**INITILIZE** vehicle Weight

**ADD** vehicle Weight to panel 1

**SET BOUNDS** vehicle Weight

**INITILIZE** vehicle Weight tf

**SET BOUNDS** vehicle Weight tf

**SET BACKGROUND** vehicle Weight tf

**SET BORDER** vehicle Weight tf

**ADD** vehicle Weight tf to panel 1

**INITILIZE** vehicle Color

**ADD** vehicle Color

**SET BOUNDS** vehicle Color

**INITILIZE** vehicle Color tf

**SET BOUNDS** vehicle Color tf

**SET BACKGROUND** vehicle Color tf

**SET BORDER** vehicle Color tf

**ADD** vehicle Color tf to panel 1

**INITILIZE** torque

**ADD** torque to panel 1

**SET BOUNDS** torque

**INITILIZE** torque tf

**SET BOUNDS** torque tf

**SET BACKGROUND** torque tf

**SET BORDER** torque tf

**ADD** torque tf to panel 1

**INITILIZE** engine Displacement

**ADD** engine Displacement to panel 1

**SET BOUNDS** engine Displacement

**INITILIZE** engine Displacement tf

**SET BOUNDS** engine Displacement tf

**SET BACKGROUND** engine Displacement tf

**SET BORDER** engine Displacement tf

**ADD** engine Displacement tf to panel 1

**INITILIZE** fuel Tank Capacity

**ADD** fuel Tank Capacity to panel 1

**SET BOUNDS** fuel Tank Capacity

**INITILIZE** fuel Tank Capacity tf

**SET BOUNDS** fuel Tank Capacity tf

**SET BACKGROUND** fuel Tank Capacity tf

**SET BORDER** fuel Tank Capacity tf

**ADD** fuel Tank Capacity tf to panel 1

**INITILIZE** ground Clearance

**ADD** ground Clearance to panel 1

**SET BOUNDS** ground Clearance

**INITILIZE** ground Clearance tf

**SET BOUNDS** ground Clearance tf

**SET BACKGROUND** ground Clearance tf

**SET BORDER** ground Clearance tf

**ADD** ground Clearance tf to panel 1

**INITILIZE** vehicle Speed

**ADD** vehicle Speed to panel 1

**SET BOUNDS** vehicle Speed

**INITILIZE** vehicle Speed tf

**SET BOUNDS** vehicle Speed tf

**SET BACKGROUND** vehicle Speed tf

**SET BORDER** vehicle Speed tf

**ADD** vehicle Speed tf to panel 1

**INITILIZE** add Auto Rickshaw Btn

**SET BOUNDS** add Auto Rickshaw Btn

**SET FOCUS PAINTED** add Auto Rickshaw Btn

**SET BACKGROUND** add Auto Rickshaw Btn

**SET FOREGROUND** add Auto Rickshaw Btn

**SET BORDER** add Auto Rickshaw Btn

**ADD** add Auto Rickshaw Btn to panel 1

**INITILIZE** panel 2

**SET BOUNDS** panel 2

**SET BACKGROUND** panel 2

**CREATE** object add Auto Title 3

**SET TITLE JUSTIFICATION** add AutoTitle 3

**SET TITLE FONT** add Auto Title 3

**SET TITLE COLOR** add Auto Titte 3

**SET BORDER** add Auto Title 3

**SET BORDER** to panel 2

**ADD** panel 2 to frame 2

**SET LAYOUT** to panel 2

**INITILIZE** vehicle ID 1

**ADD** vehicle Id 1 to panel 2

**SET BOUNDS** vehicle ID 1

**INITILIZE** vehicle Id tf 1

**SET BOUNDS** vehicle Id tf 1

**SET BORDER** vehicle Id tf 1

**ADD** vehicle Id tf 1 to panel 2

**INITILIZE** no Of Seats

**SET BOUNDS** no Of Seat

**ADD** no Of Seat to panel 2

**INITILIZE** no OF Seats tf

**SET BOUNDS** no Of Seats tf

**SET BORDER** no Of Seats tf

**ADD** no OF Seats tf to panel 2

**INITILIZE** charge Amount

**SET BOUNDS** charge Amount

**ADD** charge Amount

**INITILIZE** charge Amount tf

**SET BOUNDS** charge Amount tf

**SET BORDER** charge Amount tf

**ADD** charge Amount tf

**INITILIZE** book Date

**SET BOUNDS** book Date

**ADD** book Date to panel 2

**INITILIZE** year

**SET BOUNDS** year

**ADD** year to panel 2

**INITILIZE** days

**SET BOUNDS** days

**ADD** days to panel 2

**INITILIZE** book Auto Rickshaw Btn

**SET BOUNDS** book Auto Rickshaw Btn

**SET FOCUS PAINTED** book Auto Rickshaw Btn

**SET BACKGROUND** book Auto Rickshaw Btn

**SET FOREGROUND** book Auto Rickshaw Btn

**SET BORDER** book Auto Rickshaw Btn

**ADD** book Auto Rickshaw Btn on panel 2

**INITILIZE** display Btn

**SET BOUNDS** display Btn

**SET FOCUS PAINTED** display Btn

**SET BACKGROUND** display Btn

**SET FOREGROUND** display Btn

**SET BORDER** display Btn

**ADD** display Btn on frame 2

**INITILIZE** clear Btn

**SET BOUNDS** clear Btn

**SET FOCUS PAINTED** clear Btn

**SET BACKGROUND** clear Btn

**SET FOREGROUND** d clear Btn

**SET BORDER** clear Btn

**ADD** clear Btn on frame 2

**INITILIZE** back Btn

**SET BOUNDS** back Btn

**SET FOCUS PAINTED** back Btn

**SET BACKGROUND** back Btn

**SET FOREGROUND** back Btn

**SET BORDER** back Btn

**ADD** back Btn on frame 2

**INITILIZE** frame 3

**SET BOUNDS** frame 3

**SET DEFULT CLOSE OPERATION** frame 3

**GET CONTENT PANE** frame 3

**SET LAYOUT** frame 3

**SET LOCATION RELATIVE TO** frame 3

**INITILIZE** title Auto 2

**SET FOREGROUND** title Auto 2

**SET FONT** title Auto 2

**SET BOUNDS** title Auto 2

**ADD** title Auto 2 to frame 3

**INITILIZE** panel 3

**SET** bounds panel 3

**SET** background panel 3

**CREATE** object add Scooter Title

**SET TITLE JUSTIFICATION** add Scooter Title **SET TITLE FONT** add Scooter Title

**SET TITLE COLOR** add Scooter Title

**SET BORDER** add Scooter Title

**SET BORDER** panel 3

**SET LAYOUT** panel 3

**ADD** panel 3 to frame 3

**INITILIZE** vehicle Name 1

**SET BOUNDS** vehicle Name 1

**ADD** vehicle Name 1 to panel 3

**INITILIZE** vehicle Name tf 1

**SET BOUNDS** vehicle Name tf 1

**SET BORDER** vehicle Name tf 1

**ADD** vehicle Name tf 1 to panel 3

**INITILIZE** vehicle ID 2

**SET BOUNDS** vehicle ID 2

**ADD** vehicle ID 2 to panel 3

**INITILIZE** vehicle ID tf 2

**SET BOUNDS** vehicle ID tf 2

**SET BORDER** vehicle ID tf 2

**ADD** vehicle ID tf 2 to panel 3

**INITILIZE** vehicle Weight 1

**SET BOUNDS** vehicle Weight 1

**ADD** vehicle Weight 1 to panel 3

**INITILIZE** vehicle Weight tf 1

**SET BOUNDS** vehicle Weight tf 1

**SET BORDER** vehicle Weight tf 1

**ADD** vehicle Weight tf 1 to panel 3

**INITILIZE** vehicle Color 1

**SET BOUNDS** vehicle Color 1

**ADD** vehicle Color 1 to panel 3

**INITILIZE** vehicle Color tf 1

**SET BOUNDS** vehicle Color tf 1

**SET BORDER** vehicle Color tf 1

**ADD** vehicle Color tf 1 to panel 3

**INITILIZE** battery Capacity 1

**SET BOUNDS** battery Capacity 1

**ADD** battery Capacity 1 to panel 3

**INITILIZE** battery Capacity tf 1

**SET BOUNDS** battery Capacity tf 1

**SET BORDER** battery Capacity tf 1

**ADD** battery Capacity tf 1 to panel 3

**INITILIZE** vehicle Speed 1

**SET BOUNDS** vehicle Speed 1

**ADD** vehicle Speed 1 to panel 3

**INITILIZE** vehicle Speed tf 1

**SET BOUNDS** vehicle Speed tf 1

**SET BORDER** vehicle Speed tf 1

**ADD** vehicle Speed tf 1 to panel 3

**INITILIZE** add Electric Scooter Btn

**SET BOUNDS** add Electric Scooter Btn **SET FOCUS PAINTED** add Electric Scooter Btn

**SET BACKGROUND** add Electric Scooter Btn

**SET FOREGROUND** add Electric Scooter Btn

**SET BORDER** add Electric Scooter Btn

**ADD** add Electric Scooter Btn on panel 3

**INITILIZE** panel 4

**SET** bounds panel 3

**SET** background panel 4

**CREATE** object add Scooter Title 4

**SET TITLE JUSTIFICATION** add Scooter Title 4 **SET TITLE FONT** add Scooter Title 4

**SET TITLE COLOR** add Scooter Title 4

**SET BORDER** add Scooter Title 4

**SET BORDER** panel 4

**SET LAYOUT** panel 4

**ADD** panel 4 to frame 3

**INITILIZE** vehicle ID 4

**SET BOUNDS** vehicle ID 4

**ADD** vehicle ID 4 to panel 4

**INITILIZE** vehicle ID tf 4

**SET BOUNDS** vehicle ID tf

**SET BORDER** vehicle ID tf 4

**ADD** vehicle ID 4 tf 1 to panel 4

**INITILIZE** selling price

**SET BOUNDS** selling price

**ADD** selling price to panel 4

**INITILIZE** selling price tf

**SET BOUNDS** selling price tf

**SET BORDER** selling price tf

**ADD** selling price tf to panel 4

**INITILIZE** sell Scooter Btn

**SET BOUNDS** sell Scooter Btn

**SET FOCUS PAINTED** sell Scooter Btn

**SET BACKGROUND** sell Scooter Btn

**SET FOREGROUND** sell Scooter Btn

**SET BORDER** sell Scooter Btn

**ADD** sell Scooter Btn on panel 4

**INITILIZE** panel 5

**SET** bounds panel 5

**SET** background panel 5

**CREATE** object add Scooter Title 5

**SET TITLE JUSTIFICATION** add Scooter Title 5 **SET TITLE FONT** add Scooter Title 5

**SET TITLE COLOR** add Scooter Title 5

**SET BORDER** add Scooter Title 5

**SET BORDER** panel 5

**SET LAYOUT** panel 5

**ADD** panel 5 to frame 3

**INITILIZE** vehicle ID 5

**SET BOUNDS** vehicle ID 5

**SET BORDER** vehicle ID 5

**Add** vehicle ID 5to panel 5

**INITILIZE** vehicle ID tf 5

**SET BOUNDS** vehicle ID tf 5

**SET BORDER** vehicle ID tf 5

**ADD** vehicle ID tf 5 to panel 5

**INITILIZE** mileage

**SET BOUNDS** mileage

**SET BORDER** mileage

**Add** mileageto panel 5

**INITILIZE** mileage tf

**SET BOUNDS** mileage tf

**SET BORDER** mileage tf

**ADD** mileage tf to panel 5

**INITILIZE** charging Time

**SET BOUNDS** charging Time

**SET BORDER** charging Time

**Add** charging Timeto panel 5

**INITILIZE** charging Time tf

**SET BOUNDS** charging Time tf

**SET BORDER** charging Time tf

**ADD** charging Time tf to panel 5

**INITILIZE** brand

**SET BOUNDS** brand

**SET BORDER** brand

**Add** brandto panel 5

**INITILIZE** brand tf

**SET BOUNDS** brand tf

**SET BORDER** brand tf

**ADD** brandtf to panel 5

**INITILIZE** range

**SET BOUNDS** range

**SET BORDER** range

**Add** range to panel 5

**INITILIZE** range tf

**SET BOUNDS** range tf

**SET BORDER** range tf

**ADD** range tfto panel 5

**INITILIZE** cost Price

**SET BOUNDS** cost Price

**SET BORDER** cost Price

**Add** cost Priceto panel 5

**INITILIZE** cost Price tf

**SET BOUNDS** cost Price tf

**SET BORDER** cost Price tf

**ADD** cost Price tf to panel 5

**INITILIZE** purchase Scooter Btn

**SET BOUNDS** purchase Scooter Btn

**SET FOCUS PAINTED** purchase Scooter Btn

**SET BACKGROUND** purchase Scooter Btn

**SET FOREGROUND** purchase Scooter Btn

**SET BORDER** purchase Scooter Btn

**ADD** purchase Scooter Btn on panel 5

**INITILIZE** display Btn 1

**SET BOUNDS** display Btn 1

**SET FOCUS PAINTED** display Btn 1

**SET BACKGROUND** display Btn 1

**SET FOREGROUND** display Btn 1

**SET BORDER** display Btn 1

**ADD** display Btn 1 on frame 3

**INITILIZE** clear Btn 1

**SET BOUNDS** clear Btn 1

**SET FOCUS PAINTED** clear Btn 1

**SET BACKGROUND** clear Btn 1

**SET FOREGROUND** clear Btn 1

**SET BORDER** clear Btn 1

**ADD** clear Btn 1 on frame 3

**ADD** Action Listener in embarkBtn

**CALL** anonymous method for action performs

**THEN**

**SET** frame1 visible to false

**SET** frame2 visible to true

**END** **METHOD**

**ADD** Action Listener in embarkBtn1

**CALL** anonymous method for action performs

**THEN**

**ADD** auto

**END** **METHOD**

**ADD** Action Listener in add Auto Rickshaw Btn

**CALL** anonymous method for action performs

**THEN**

**ADD** auto

**END** **METHOD**

**ADD** Action Listener in back Btn

**CALL** anonymous method for action performs

**THEN**

**SET** frame1 visible to false

**SET** frame2 visible to true

**END** **METHOD**

**ADD** Action Listener in back 1 Btn

**CALL** anonymous method for action performs

**THEN**

**SET** frame1 visible to false

**SET** frame2 visible to true

**END** **METHOD**

**ADD** Action Listener in display Btn 1

**CALL** anonymous method for action performs

**FOR** loop vehicle array

**IF** display instance of Electric Scooter

**CALL** electric Scooter display

**END** **METHOD**

**ADD** Action Listener in clear Btn

**CALL** anonymous method for action performs

**THEN**

**SET TEXT** vehicle ID tf

**SET TEXT** vehicle Name tf

**SET TEXT** vehicle Weight tf

**SET TEXT** ground Clearance tf

**SET TEXT** vehicle Speed tf

**SET TEXT** engine Displacement tf

**SET TEXT** vehicle Color tf

**SET TEXT** torque tf

**SET TEXT** fuel Tank Capacity tf

**SET TEXT** vehicle Id tf 1

**SET TEXT** no Of Seats tf

**SET TEXT** charge Amount tf

**SET SELECTED ITEM** year

**SET SELECTED ITEM** month

**SET SELECTED ITEM** Day

**END METHOD**

**ADD** Action Listener in clear 1 Btn

**CALL** anonymous method for action performs

**THEN**

**SET TEXT** vehicle ID tf 2

**SET TEXT** vehicle Name tf 1

**SET TEXT** vehicle Weight tf 1

**SET TEXT** vehicle Color tf 1

**SET TEXT** battery Capacity tf 1

**SET TEXT** vehicle Speed tf 1

**SET TEXT** vehicle ID tf 4

**SET TEXT** brand tf

**SET TEXT** mileage tf

**SET TEXT** range tf.

**SET TEXT** charging Time tf

**SET TEXT** cost Price tf

**SET TEXT** vehicle ID tf 5

**SET TEXT** selling Price tf

**END METHOD**

**ADD** Action Listener in book Auto Rickshaw Btn

**CALL** anonymous method for action performs

**THEN**

**BOOK** Auto

**END** **METHOD**

**ADD** Action Listener in book Auto Electric Scooter Btn

**CALL** anonymous method for action performs

**THEN**

**ADD** Electric

**END** **METHOD**

**ADD** Action Listener in book purchase Scooter Btn

**CALL** anonymous method for action performs

**THEN**

**PURCHASE** Electric

**END** **METHOD**

**ADD** Action Listener in book sell Scooter Btn

**CALL** anonymous method for action performs

**THEN**

**SELL** Electric

**END** **METHOD**

**FUNCTION** get Auto Vehicle Id Add

**DO**

**INITILIZE** vehicle Id as Invalid

**TRY**

**THEN**

**IF** (vehicleIDtf.getText().equals(""))

**THEN**

**SET** VehicleID to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** VehicleID from textfield

**IF** (VehicleID <= 0)

**THEN**

**SET** VehicleID to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** VehicleID

**END DO**

**END FUNCTION**

**FUNCTION** get Auto Vehicle Name

**DO**

**DECLARE** vehicle Name and **INITILIZE** as vehicleNametf.getText()

**FUNCTION** get Auto Vehicle Weight

**DO**

**DECLARE** vehicle Weight and **INITILIZE** as vehicleWeighttf.getText()

**FUNCTION** get Auto Vehicle Color

**DO**

**DECLARE** vehicle Color and **INITILIZE** as vehicleColortf.getText()

**FUNCTION** get Auto Vehicle Speed

**DO**

**DECLARE** vehicle Speed and **INITILIZE** as vehicleSpeedtf.getText()

**FUNCTION** get Engine Displacement

**DO**

**INITILIZE** engineDisplacement as Invalid

**TRY**

**THEN**

**IF** (engineDisplacementtf.getText().equals("")) **THEN**

**SET** engineDisplacement to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** engineDisplacement from textfield

**IF** (engineDisplacement<= 0)

**THEN**

**SET** engineDisplacement to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** engineDisplacement

**END DO**

**END FUNCTION**

**FUNCTION** get Auto fuelTankCapacity

**DO**

**INITILIZE** fuelTankCapacity as Invalid

**TRY**

**THEN**

**IF** (fuelTankCapacitytf.getText().equals("")) **THEN**

**SET** fuelTankCapacityto EMPTY

**END IF**

**ELSE**

**THEN**

**GET** fuelTankCapacity from textfield

**IF** (fuelTankCapacity<= 0)

**THEN**

**SET** fuelTankCapacity to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** fuelTankCapacity

**END DO**

**END FUNCTION**

**FUNCTION** get Auto Ground Clearance

**DO**

**DECLARE** Ground Clearance and INITILIZEas getAutoGroundClearance()

**FUNCTION** Torque

**DO**

**DECLARE** Torque and **INITILIZE** as getAutoTorque()

**FUNCTION** get Auto Vehicle Id 1 Book

**DO**

**INITILIZE** vehicleId1 as Invalid

**TRY**

**THEN**

**IF** (vehicleIdtf1.getText(). equals(vehicleId1"")) **THEN**

**SET** vehicleId1 to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** vehicleId1 from textfield

**IF** vehicleId1<= 0)

**THEN**

**SET** vehicleId1 to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** vehicleId1

**END DO**

**END FUNCTION**

**FUNCTION** get Auto No of Seat

**DO**

**INITILIZE** noOfSeats as Invalid

**TRY**

**THEN**

**IF** (vehicleIdtf1.getText(). equals(vehicleId1"")) **THEN**

**SET** noOfSeats to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** noOfSeats from textfield

**IF** noOfSeats<= 0)

**THEN**

**SET** noOfSeats to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** noOfSeats

**END DO**

**END FUNCTION**

**FUNCTION** getAutoChargeAmount()

**DO**

**INITILIZE** Charge Amount as Invalid

**TRY**

**THEN**

**IF** (chargeAmounttf.getText().equals("")) **THEN**

**SET** ChargeAmount to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** ChargeAmount from textfield

**IF** ChargeAmount<= 0)

**THEN**

**SET** ChargeAmount to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** ChargeAmount

**END DO**

**END FUNCTION**

**FUNCTION** getYear()

**DECLEAR** Yand **INITILIZE** year.getSelectedItem(**)**

**RETURN**

**FUNCTION** getYear()

**DECLEAR** Yand **INITILIZE** year.getSelectedItem(**)**

**RETURN**

**FUNCTION** get Month ()

**DECLEAR** dand **INITILIZE** month.getSelectedItem()

**RETURN**

**FUNCTION** get day ()

**DECLEAR** dand **INITILIZE** day.getSelectedItem()

**RETURN**

**FUNCTION** get Auto No of Seat

**DO**

**INITILIZE** noOfSeats as Invalid

**TRY**

**THEN**

**IF** (vehicleIdtf1.getText(). equals(vehicleId1"")) **THEN**

**SET** noOfSeats to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** noOfSeats from textfield

**IF** noOfSeats<= 0)

**THEN**

**SET** noOfSeats to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** noOfSeats

**END DO**

**END FUNCTION**

**FUNCTION** getAutoVehicleId2()

**DO**

**INITILIZE** vehicleID2as Invalid

**TRY**

**THEN**

**IF** (vehicleIDtf2.getText().equals("")) **THEN**

**SET** vehicleID2 to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** vehicleID2 from textfield

**IF** vehicleID2<= 0)

**THEN**

**SET** vehicleID2 to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** vehicleID2

**END DO**

**END FUNCTION**

**FUNCTION** getAutoVehicleName1()

**DO**

**DECLARE** VehicleName1 and **INITILIZE** as VehicleNametf1

**FUNCTION** getAutoVehicleColor1()

**DO**

**DECLARE** vehicleColor1 and **INITILIZE** as vehicleColortf1

**FUNCTION** getAutoVehicleWeight1()

**DO**

**DECLARE** vehicleWeight1 and **INITILIZE** as vehicleWeight1

**FUNCTION** getAutoVehicleSpeed1()

**DO**

**DECLARE** vehicleSpeed1 and **INITILIZE** as vehicleSpeed1

**FUNCTION** getAutoBatteryCapacity()

**DO**

**INITILIZE** batteryCapacity1 as Invalid

**TRY**

**THEN**

**IF** (batteryCapacitytf1.getText().equals("")) **THEN**

**SET** batteryCapacity1 to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** batteryCapacity1 from textfield

**IF** batteryCapacity1<= 0)

**THEN**

**SET** batteryCapacity1 to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** batteryCapacity1

**END DO**

**END FUNCTION**

**FUNCTION** getAutoVehicleId3()

**DO**

**INITILIZE** vehicleID3 as Invalid

**TRY**

**THEN**

**IF** (vehicleIDtf5.getText().equals("")) **THEN**

**SET** vehicleID3 to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** vehicleID3 from textfield

**IF** vehicleID3<= 0)

**THEN**

**SET** vehicleID3 to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** vehicleID3

**END DO**

**END FUNCTION**

**FUNCTION** getAutoMileage()

**DECLEAR mileage** and **INITILIZE** mileagetf

**RETURN**

**FUNCTION** getAutoBrand()

**DECLEAR brand** and **INITILIZE** brandtf

**RETURN**

**FUNCTION** getAutoRange()

**DO**

**INITILIZE** range as Invalid

**TRY**

**THEN**

**IF** (rangetf.getText().equals("")) **THEN**

**SET** range to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** range from textfield

**IF** range<= 0)

**THEN**

**SET** range to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** range

**END DO**

**END FUNCTION**

**FUNCTION** getAutoCostPrice()

**DO**

**INITILIZE** costPrice as Invalid

**TRY**

**THEN**

**IF** (costPricetf.getText().equals("")) **THEN**

**SET** costPrice to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** costPrice from textfield

**IF** costPrice<= 0)

**THEN**

**SET** costPrice to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** costPrice

**END DO**

**END FUNCTION**

**FUNCTION** getAutoChargingTime()

**DO**

**DECLARE chargingTime** and **INITILIZE** as chargingTime

**FUNCTION** getAutoVehicleID5()

**DO**

**INITILIZE** vehicleID5 as Invalid

**TRY**

**THEN**

**IF** (vehicleIDtf4.getText().equals("")) **THEN**

**SET** vehicleID5 to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** vehicleID5 from textfield

**IF** vehicleID5 <= 0)

**THEN**

**SET** vehicleID5 to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** vehicleID5

**END DO**

**END FUNCTION**

**FUNCTION** getAutoSellingPrice()

**DO**

**INITILIZE** sellingPrice as Invalid

**TRY**

**THEN**

**IF** (sellingPricetf.getText().equals("")) **THEN**

**SET** sellingPrice to EMPTY

**END IF**

**ELSE**

**THEN**

**GET** sellingPrice from textfield

**IF** sellingPrice <= 0)

**THEN**

**SET** sellingPrice to INVALID

**END IF**

**END ELSE**

**END** **TRY**

**CATCH**

**THEN**

**END** CATCH

**RETURN** sellingPrice

**END DO**

**END FUNCTION**

**FUNCTION isUnique(int)**

**DECLARE isunique and set true**

**FOR vehicleId in list**

**IF vehicleId.getVehicleId() equals ID**

**Set unique equals false**

**RETURN unique**

**FUNCTION addAuto**

**IF getAutoVehicleIdAdd()** equal to **EMPTY || getAutoVehicleName().equals("") || getAutoVehicleWeight().equals("") || getAutoVehicleColor().equals("") || getAutoVehicleSpeed().equals("")**

**|| getEngineDisplacement() == EMPTY || getAutoFuelTankCapacity()** equal to **EMPTY || getAutoGroundClearance().equals(""))**

**JOptionPane Show message**

**ELSE IF getAutoVehicleIdAdd** is INVALID

**JOPTIONPANE Show message**

**IF UNIQUE (getAutoVehicleIdAdd())**

**ADD**

**LIST(new AutoRickshaw(getAutoVehicleIdAdd(),getAutoVehicleName(), getAutoTorque(), getAutoVehicleWeight(), getAutoVehicleColor(), getAutoVehicleSpeed(), getAutoGroundClearance(), getEngineDisplacement(), getAutoFuelTankCapacity()**

**JOPTIONPANE Show message**

**ELSE**

**JOptionPane Show message**

**END METHOD**

**FUNCTION bookAuto()**

**IF (getAutoVehicleId1Book() == EMPTY || getAutoNoOfSeat() == EMPTY || getAutoChargeAmount() == EMPTY**

**JOptionPane.showMessageDialog**

**ELSE IF**

**(getAutoVehicleId1Book() IS INVALID || getAutoNoOfSeat() IS INVALID**

**|| getAutoChargeAmount() IS INVALID)**

**JOPTIONPANE Show message**

**ELSE IF**

**(((AutoRickshaw) id).getIsBooked()** is  **true**

**JOptionPane Show message**

**ELSE IF**

**(id.getvehicleID() is equal to getAutoVehicleId1Book()**

**JOptionPane ERROR MESSAGE**

**IF (!book && !found)**

**JOption Pane show Message Dialog**

**END METHOD**

**FUNCTION** sell Electric

**IF**

(getAutoVehicleID5() is EMPTY || getAutoSellingPrice() is EMPTY)

**J OPTION PANE SHOW MESSAGE DIALOG**

**ELSE IF**

**getAutoVehicleID5() is EMPTY || getAutoSellingPrice() is EMPTY)**

**J OPTION PANE SHOW MESSAGE DIALOG**

**FOR**

Vehicle id in list

**IF**

id instance of ElectricScooter

**IF**

Electric**scooter** is sold then false

**ELSE IF**

ElectricScooter id is getHasSold() then true

**J OPTION PANE SHOW MESSAGE DIALOG**

**ELSE IF**

id.getvehicleID() is equal to getAutoVehicleId1Book()

**J OPTION PANE SHOW MESSAGE DIALOG**

**END METHOD**

**MAIN FUNCTION**

TransportGUI

# Method Description

A method in object-oriented programming is a programmed process that is declared as part of a class and included in any object of that class. There can be several methods in a class. To maintain data integrity among the collection of objects in an application, a method in an object can only access the data that is known to that object. A method may be applied to several objects. (TechTarget Contributor, 2005)

Method description of Transport GUI is below: -

|  |  |  |
| --- | --- | --- |
| **S. N** | **Method** | **Description** |
|  | **TransportGUI()** | **It is the TransportGUI class constructor. This method helps in initializing the values of various characteristics and takes various attributes as parameters.** |
|  | **Main(String[])** | **This is the primary technique. It regulates every method and programming.** |
|  | **getAutoVehicleIDAdd()** | **This method helps in creating the vehicle IDvalid value as well as checking its various requirements for the add getAutoVehicleIDAdd() panel. Integers make up its data type.** |
|  | **getAutoVehicleNameAdd()** | **The obtaining text for the Name text field for the autorickshaw is stored in a variable. The user's choice for the autorickshaw's name is stored in a variable and returned by this function. It has a return type of String.** |
|  | **getAutoVehicleWeight()** | **This is a way for retrieving the value of getvehicleWeight(). This method evaluates the text that the user writes in the text box, stores it in a variable, and then returns it. It has a return type of String.** |
|  | **getAutoVehicleColor()** | **This function returns the text that was obtained and stores it in a string variable. It has a return type of String.** |
|  | **getAutoVehicleSpeed()** | **This approach uses a text field to retrieve the vehicle's speed. This function returns the text that was obtained and stores it in a string variable.** |
|  | **getEngineDisplacement()** | **This method attempts to convert the engine displacement value from a string variable into an integer. When converting, determine if the number is positive or not before returning the result.** |
|  | **getAutoFuelTankCapacity()** | **This method attempts to convert the value of the gasoline tank capacity from a string variable to an integer. When converting, determine if the number is positive or not before returning the result.** |
|  | **getAutoGroundClearance()** | **This function returns the text that was obtained and stores it in a string variable. It has a return type of String.** |
|  | **getAutoTorque()** | **The user's input is verified by this function, which then stores it in a variable and returns it. It has a return type of String.** |
|  | **getAutoVehicleID1Book()** | **This method is used to determine if the vehicle ID is unique or not and whether the data in the panel 2 book autorickshaw is accurate or not.** |
|  | **getAutovehicleId1()** | **This method is used to determine if the vehicle ID is unique or not and whether the data in the panel 1 add autorickshaw is accurate or not.** |
|  | **getAutoNoOfSeat()** | **This procedure saves the text from the "Seat Number" text field in a variable.  Its return type is int.** |
|  | **getAutoChargeAmount()** | **This function saves the text from the " Number of Seat" text field in a variable. Then attempt to convert to an integer. Check if it is smaller than 0 if the conversion succeeds, then return.** |
|  | **getYear()** | **This function creates an array list of years and stores them as strings.** |
|  | **getMonth()** | **This function creates an array list of months and stores them as strings.** |
|  | **getday()** | **This function creates an array list of days and stores them as strings.** |
|  | **getBookedDate()** | **When booking an autorickshaw, this method is used to provide the appropriate information in the dialog box. it has a void data type.** |
|  | **getAutoVehicleId2()** | **This method helps in creating the vehicle ID valid as well as checking its various requirements for the add getAutoVehicleId2() . Integers make up its data type.** |
|  | **getAutoVehicleName1** | **The obtaining text for the Name text field for the electric Scooter is stored in a variable. It has a return type of String.** |
|  | **getAutoVehicleColor1()** | **This function returns the text that was obtained and stores it in a string variable. It has a return type of String.** |
|  | **getAutoVehicleWeight1()** | **This is a way for retrieving the value of getvehicleWeight1(). This method evaluates the text that the user writes in the text box, stores it in a variable, and then returns it. It has a return type of String.** |
|  | **getAutoVehicleSpeed1()** | **This approach uses a text field to retrieve the vehicle's speed 1. This function returns the text that was obtained and stores it in a string variable.** |
|  | **getAutoBatteryCapacity()** | **This is a getting method to extract the text from the battery capacity and save it in a variable.Its return type is an int.** |
|  | **getAutoVehicleId3** | **This method helps in creating the vehicle ID 3 valid as well as checking its various requirements for the add getAutoVehicleId3() . Integers make up its data type.** |
|  | **getAutoMileage()** | **This function returns the text that was obtained and stores it in a string variable. It has a return type of String.** |
|  | **getAutoBrand()** | **This process helps to determine brand value. It has a string data type.** |
|  | **getAutoRange()** | **It only accepts valid data and is used to verify whether the vehicle ID in the bought panel is legitimate or unique.** |
|  | **getAutoCostPrice()** | **It helps to get the cost price from the text field.** |
|  | **getAutoChargingTime()** | **This method helps to determine the charging time value. String is the data type.** |
|  | **getAutoVehicleID5()** | **This method helps in creating the vehicle ID valid as well as checking its various requirements for the add getAutoVehicleId5() . Integers make up its data type.** |
|  | **getAutoSellingPrice()** | **If all the conditions are satisfied, this approach is utilized to sell the electric scooter after it has been bought.** |
|  | **unique (int ID)** | **The vehicle id should be unique in this program, for that reason. If the vehicle ID is unique, it returns a value; otherwise, a dialogue box appears.** |
|  | **addAuto()** | **Add Auto has been added to an array list, this procedure is used to reserve the autorickshaw. A message box is displayed if any fields are blank or incorrect.** |
|  | **bookAuto()** | **book Auto has been added to an array list, this procedure is used to reserve the autorickshaw. A message box is displayed if any fields are blank or incorrect.** |
|  | **addElectric()** | **Once an autorickshaw has been added to an array list, this procedure is used to reserve the vehicle. A message box is displayed if any fields are blank or incorrect.** |
|  | **purchaseElectric()** | **After purchasing the Electric Scooter, this method is followed to sell it. If any fields are empty or invalid when we click the Sell button on the Electric Scooter frame, the appropriate notice box pops up. The vehicle will then be sold if all the conditions are met.** |
|  | **sellElectric()** | **After purchasing the Electric Scooter, this method is followed to sell the vehicle will then be sold if all the conditions are met.** |
|  | **Main (String[])** | **It is Java's most vital method. The primary method in this coursework is to call the constructor.** |

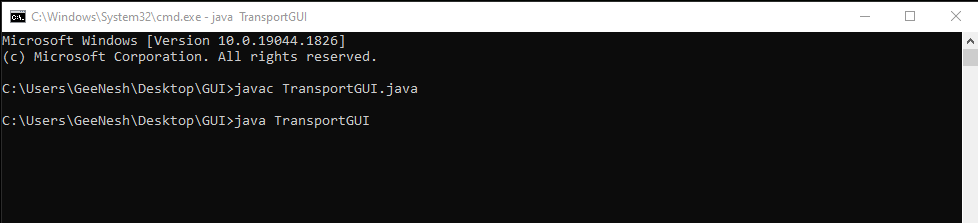
# Testing

All features of your application are covered by the thorough and functional test cases offered by Java testing. In this Coursework, the command is used to investigate each individual java file, and the inspection table with several snapshots is shown below:

## Test 01

|  |  |  |
| --- | --- | --- |
| **S. N** | **Testing** | |
|  | **Objective** | **Test that the program can be compiled and run using the command prompt** |
|  | **Action** | 1. **Launching the command prompt from the directory that contains the java file.** 2. **Compile java file** 3. **Run the Java program** |
|  | **Expected Result** | **The GUI will be launched when the TransportGUI java file has been compiled.** |
|  | **Actual Result** | **The GUI was opened after the compiled java file.** |
|  | **Conclusion** | **Test is Successful** |

**Table 5 Testing to compile and run using CMD**



**Figure 10 Compiling and Running in Command Prompt**

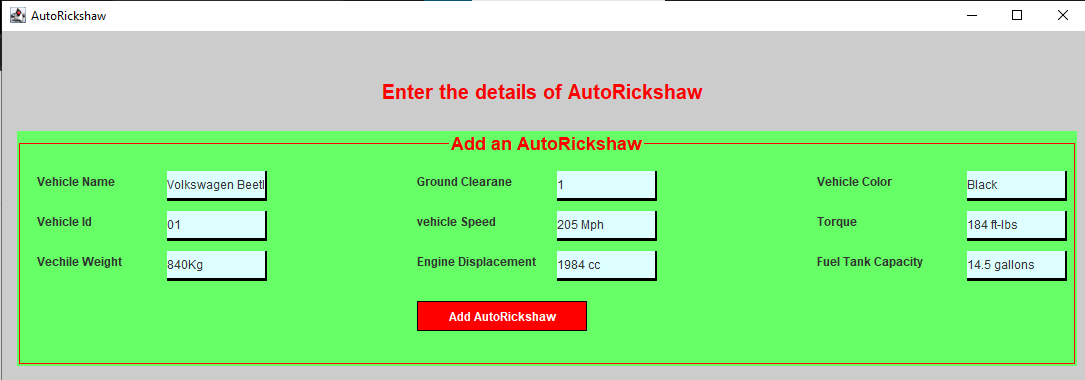
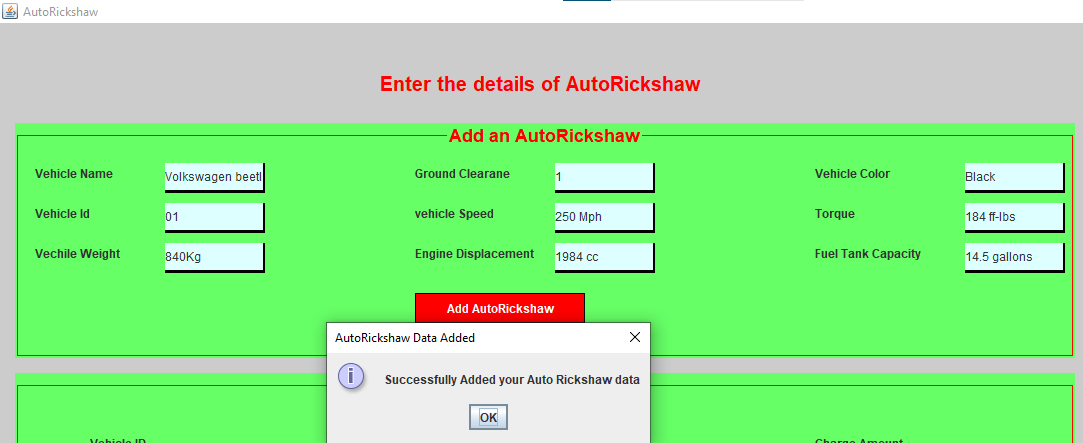
Graphical user interface, text

Description automatically generated

**Figure 11 Image of TransportGUI**

## Test 02

|  |  |  |
| --- | --- | --- |
| **S. N** | **Testing** | |
|  | **Objective** | **To test Buttons of Autorickshaw and Electric Scooter**   1. **Add the Autorickshaw** 2. **Add the Electric Scooter** 3. **Book the Autorickshaw** 4. **Purchase the Electric Scooter** 5. **Sell the Electric Scooter** |
|  | **Action** | * **To Add in Auto-Rickshaw**  1. **Open the Auto-Rickshaw Frame.** 2. **Fill up each text field on panel one with correct info.** 3. **Add the Auto-Rickshaw and it will pop up.**  * **To Book an Auto-Rickshaw**  1. **Open the Auto-Rickshaw Frame.** 2. **Fill up each text field on panel two with correct info.** 3. **Book the Auto-Rickshaw and it will pop up.**  * **To Book, Purchase and Sell the Electric Scooter.**  1. **Open the Electric-Scooter Frame** 2. **Fill up each text field on panel three, four and five with correct info.** 3. **Add, Purchase and Sell the Electric Scooter respectively.** |
|  | **Expected Result** | **Both the auto rickshaw and the electric scooter should be added, together with their objects, to the array list.** |
|  | **Actual Result** | **Both the Auto-Rickshaw and the Electric Scooter were added, purchased, and sold. The array list had both the auto-rickshaw and the electric scooter objects.** |
|  | **Conclusion** | **Test is Successful** |

**Table 6 Testing the autorickshaw and electric scooter buttons.**

**Figure 9 Inserting the details of Autorickshaw**

**Figure 9 Inserting the details of Autorickshaw**

**Figure 10 Image of Added Autorickshaw data**

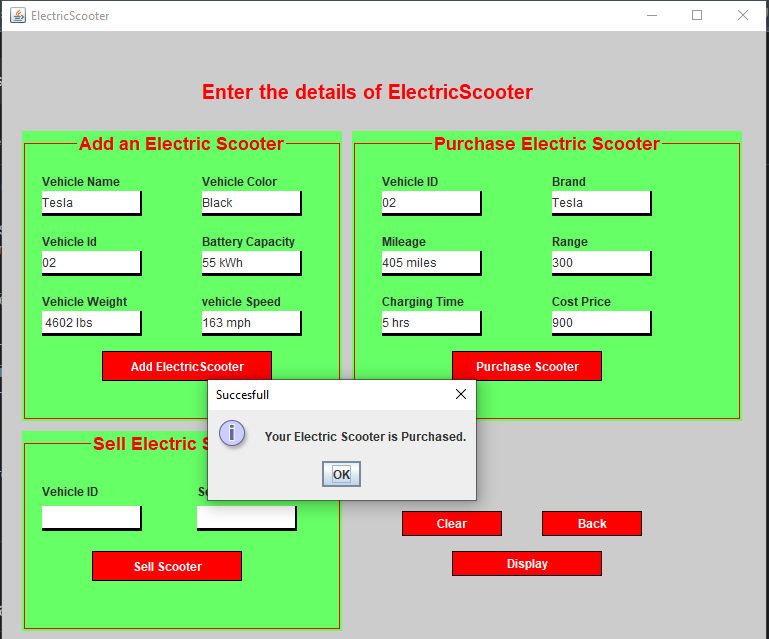
Graphical user interface

Description automatically generatedGraphical user interface, diagram

Description automatically generated

**Figure 11 Image of Added Electric Scooter data**

**Figure 12 Image of Booked Auto Rickshaw**



**Figure 12 Image of Purchased Electric Scooter**

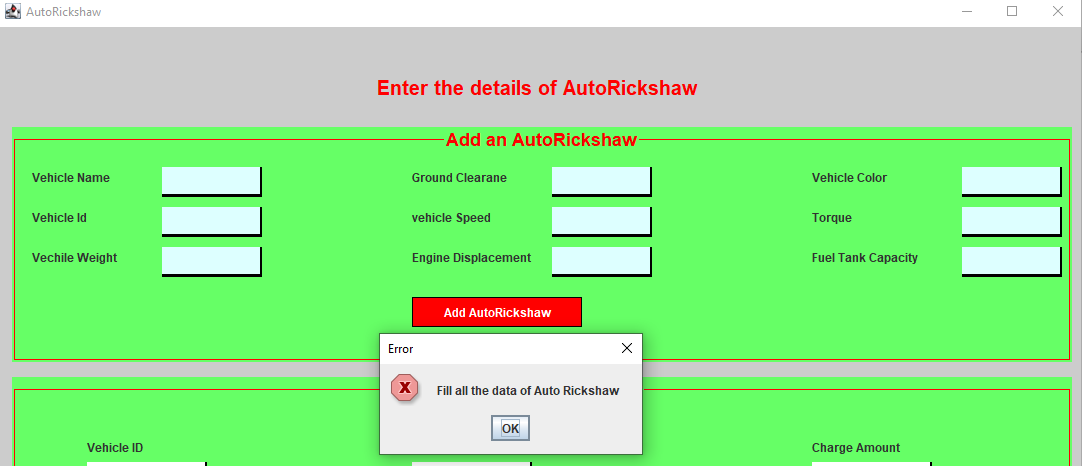
Graphical user interface

Description automatically generated

**Figure 13 Image of successful Sold of electric Scooter**

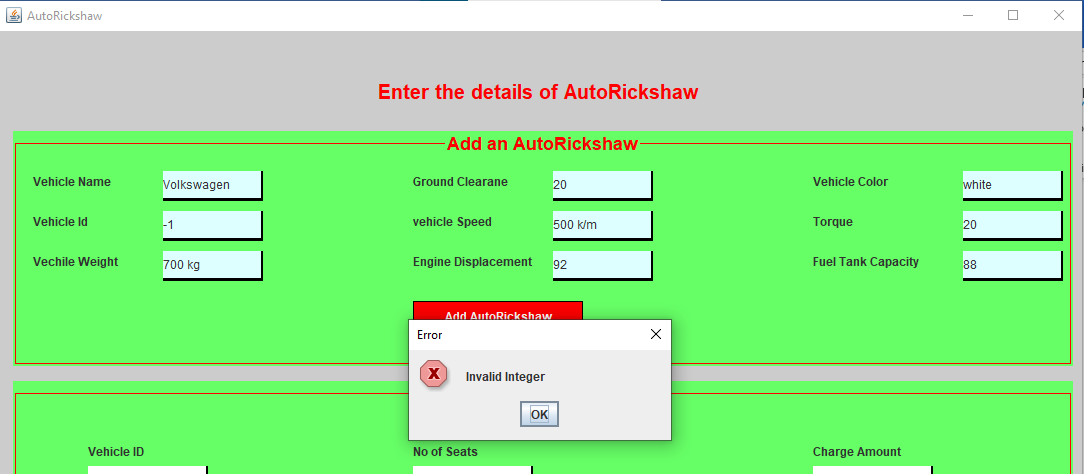
## Test 3

|  |  |  |
| --- | --- | --- |
| **S. N** | **Testing** | |
|  | **Objective** | **To test that appropriate dialog boxes, appear when unsuitable values are entered for the Vehicle ID** |
|  | **Action** | 1. **Empty text field was added in Add Auto Rickshaw.** 2. **Negative value was entered in Add Auto Rickshaw** 3. **Value of Auto Rickshaw was entered twice.** 4. **Negative value was entered in Electric Scooter.** 5. **Value of Electric Scooter was entered twice.** 6. **Empty text field was booked in Auto Rickshaw.** 7. **Value of Add Electric Scooter was entered twice.** 8. **Empty text field was booked in Add Electric Scooter.** 9. **Negative value was entered in purchase Electric Scooter** 10. **Value of purchase Electric Scooter was entered twice.** 11. **Empty text field was added in purchase Electric Scooter.** 12. **Selling Electric Scooter for twice error.** 13. **Empty text field was added in purchase Electric Scooter.** |
|  | **Expected Result** | **For each error, the appropriate pop-up message with the explicit objective should be presented.** |
|  | **Actual Result** | **The appropriate pop-up message was shown with the appropriate message.** |
|  | **Conclusion** | **Test is Successful** |



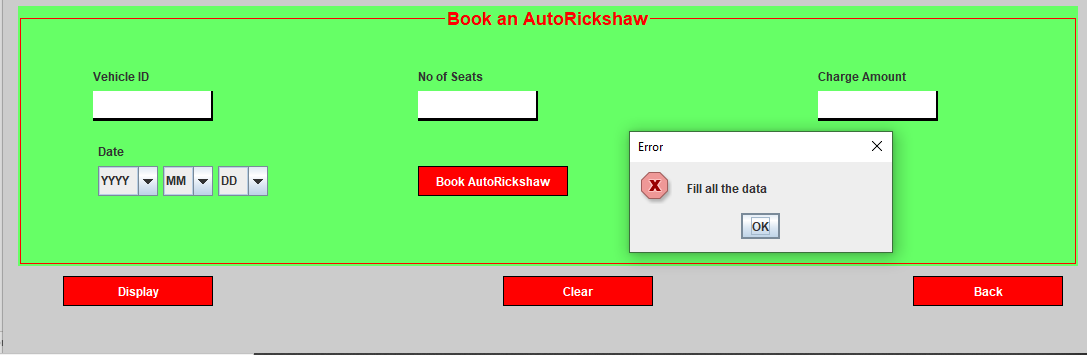
**Figure 13 Empty Text field of Auto Rickshaw**

Graphical user interface

Description automatically generated

**Figure 15 Image of using Same duplicate Vehicle ID**

**Figure 14 Image of Invalid Integer in Add Auto Rickshaw**

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated

**Figure 18 Image Empty text field in Book Autorickshaw**

**Figure 17 Image of Already Booked Auto Rickshaw**

**Figure 16 Image of Invalid Integer in Book Autorickshaw**

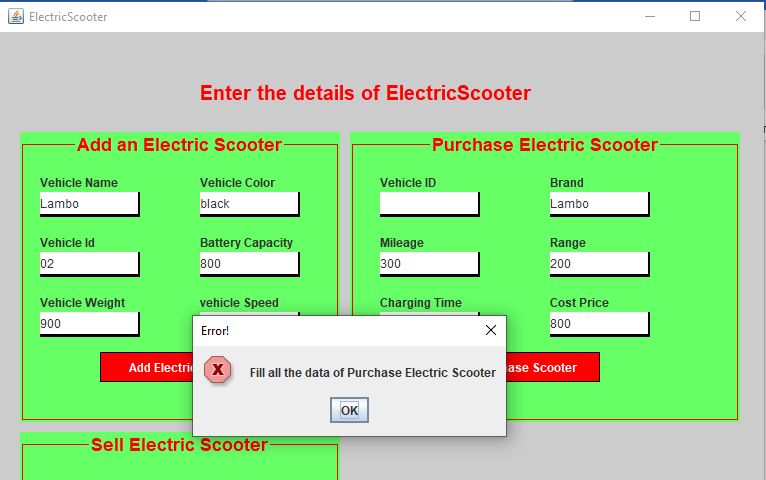
Graphical user interface

Description automatically generatedA picture containing graphical user interface

Description automatically generated

**Figure 18 Image of Missing Data in Electric Scooter**

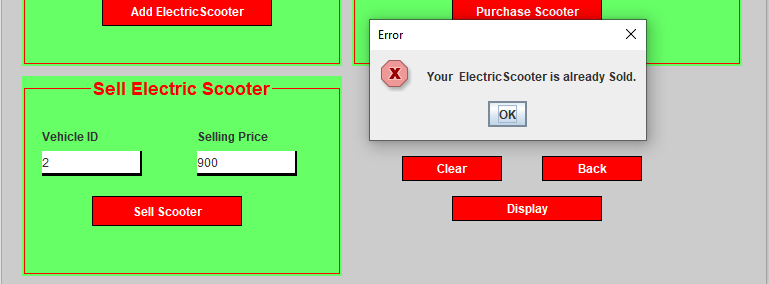
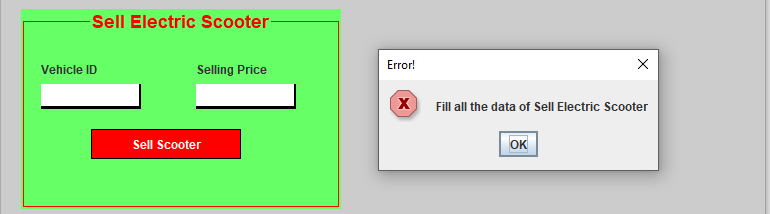
**Figure 17 Image of Duplicate Vehicle ID in add Electric Scooter**

Diagram

Description automatically generated

**Figure 20 Empty text field on Purchase Electric Scooter**

**Figure 19 Image of Invalid Integer in vehicle ID on Purchase Electric Scooter**



**Figure 22 Image of Empty Text Field Error**

**Figure 21 Image of Selling the Sold Scooter error**

# Error

Error is a term used to describe a user action that is prohibited, and which causes the program to operate improperly. Programming mistakes frequently go unnoticed until the program is compiled or run. Therefore, mistakes should be fixed before to compiling and execution. The following are some errors and their correction:

## Syntax Errors

A syntax error happens when data is input into a computer in an unfamiliar or inappropriate format. An example of a syntax error would be if someone wrote "dotcom" in place of ".com" while typing an email address, rendering the message undeliverable. Information input in a poor or inappropriate format might have been created by a programmer, a user, or even computer software itself. (Gaudet, 2022) Example of Syntax error is below:

* **Background pattern

  Description automatically generatedError: The main method is missing a curly bracket.**

**Figure 15 Image of Syntax Error**

**Figure 16 Image of Syntax Error**

* **Background pattern

  Description automatically generatedCorrection: The missing curly bracket has been added.**

**Figure 17 Correction of Syntax Error**

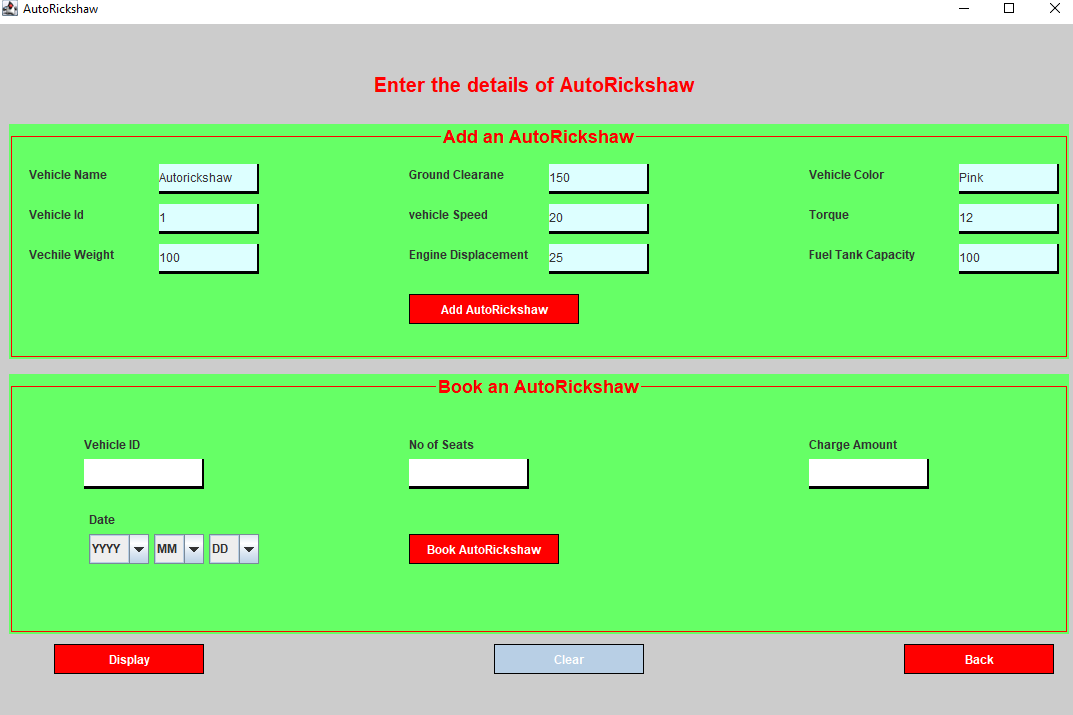
## Logic Error

An instance of a logic error is when a computer program receives a result that is illogical but is not flagged as an error. A software that has a logic fault is particularly risky since it could declare false findings as true. Since a logic mistake won't make the software crash, it's challenging for the application's creators to find and fix the problem. (Computer Hope, 2018) Example of Logic Error is below:

* **Graphical user interface, text, application

  Description automatically generatedError: Electric scooter is kept in the text area. So, the clear button is inoperative.**

**Figure 18 Swapping the Clear Button**



**Figure 19 Logic Error not clearing text field in image**

* Text

  Description automatically generated**Correction: The clear button is functional and the auto rickshaw's text field is preserved.**

**Figure 20 Correct Code of Clear Button**

Graphical user interface

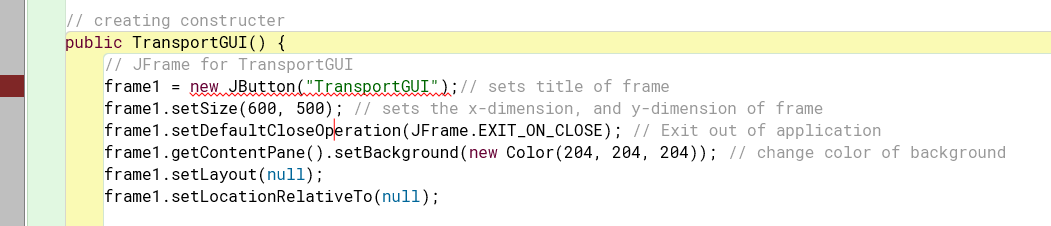
Description automatically generated

**Figure 21The Clear Button is operational.**

## Semantics Error

A program's semantics are the collection of rules that define its meaning. Semantics allows us to ensure that a program will achieve the desired outcome. This sort of mistake will allow a program to run, but it won't provide the desired outcome. (Mbula, 2022)

* **Error:** **JFrame has been set up as a JButton initially.**

****

**Figure 22 Image of Semantics Error**

* **Text

  Description automatically generatedCorrection: In order to fix that, we must initialize fame 1 as JFrame.**

**Figure 23 Correction of Semantic Error**

# Conclusion

At last, but not least I am finally completed this coursework with a lot of hard work and research. Through this coursework, we have gained knowledge of a variety of topics, including declaration, how to design getter methods, action listeners, etc.

This course module's emphasis is on using the Java programming language to construct a whole GUI with appropriate validation. All students were helped by this coursework to thoroughly understand the subject and to create and comprehend the appropriate concepts for this programming. There were times when I had trouble programming in purchase Electric Scooter, but I was able to get through them with the help of my teacher, a friend, and the material presented on Google Classroom. This module coursework represents the beginning stages in learning how to execute our final year project (FYP).

Furthermore, we have been able to test our analytic abilities thanks to this coursework. This module is beneficial for taking our analytical abilities to the next level. In this way, I accomplished my work with the help of my senior classmate, and module instructor. I would like to appreciate every one of them.

# References

Anon., 2021. *IBM.* [Online]   
Available at: https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams  
[Accessed 02 08 2022].

Computer Hope, 2018. *ComputerHope.* [Online]   
Available at: https://www.computerhope.com/jargon/l/logierro.htm  
[Accessed 01 08 2022].

Gaudet, H., 2022. *EasyTechJunkie.* [Online]   
Available at: https://www.easytechjunkie.com/what-is-a-syntax-error.htm  
[Accessed 01 08 2022].

geeksforgeeks, 2020. *geeksforgeeks.* [Online]   
Available at: https://www.geeksforgeeks.org/introduction-of-bluej/  
[Accessed 31 07 2022].

Geeksforgeeks, 2021. *geeksforgeeks.* [Online]   
Available at: https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/  
[Accessed 31 07 2022].

Johari, A., 2021. *Edureka.* [Online]   
Available at: https://www.edureka.co/blog/what-is-java/  
[Accessed 30 07 2022].

Mbula, A. L., 2022. *educative.* [Online]   
Available at: https://www.educative.io/answers/what-is-the-difference-between-syntax-and-semantic-errors  
[Accessed 04 08 2022].

Pedamkar, P., 2022. *EDUCBA.* [Online]   
Available at: https://www.educba.com/what-is-visual-studio-code/  
[Accessed 31 07 2022].

Steele, C., 2013. *TechTarget.* [Online]   
Available at: https://www.techtarget.com/searchmobilecomputing/definition/Google-Chrome-browser  
[Accessed 31 07 2022].

TechTarget Contributor, 2005. *TechTarget.* [Online]   
Available at: https://www.techtarget.com/whatis/definition/method  
[Accessed 31 07 2022].

# Appendix

//improting used pacakage

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.border.TitledBorder;

import java.awt.event.ActionEvent;

import javax.swing.BorderFactory;

import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionListener;

import javax.swing.JPanel;

import java.util.ArrayList;

import java.util.List;

public class TransportGUI {

// creating instance variable

private JFrame frame1;

private JFrame frame2;

private JFrame frame3;

private JPanel panel1;

private JPanel panel2;

private JPanel panel3;

private JPanel panel4;

private JPanel panel5;

private JButton embarkBtn, embarkBtn1, addAutoRickshawBtn, bookAutoRickshawBtn, backBtn, displayBtn, clearBtn,

addElectricScooterBtn, purchaseScooterBtn, sellScooterBtn, back1Btn, displayBtn1, clear1Btn;

private JLabel title, titleAuto, titleAuto1, titleAuto2, titleElectric, vehicleID, vehicleColor, torque,

vehicleName, vehicleSpeed, fuelTankCapacity, vehicleWeight, engineDisplacement, groundClearance, vehicleId1,

noOfSeats, chargeAmount, vehicleID2, vehicleWeight1, vehicleName1, vehicleColor1, vehicleSpeed1,

batteryCapacity1, vehicleID5, mileage, chargingTime, brand, range, costPrice, vehicleID4, sellingPrice,

bookDate;

private JTextField vehicleIDtf, vehicleColortf, torquetf, vehicleNametf, vehicleSpeedtf, fuelTankCapacitytf,

vehicleWeighttf, engineDisplacementtf, groundClearancetf, vehicleIdtf1, noOfSeatstf, chargeAmounttf,

vehicleNametf1, vehicleIDtf2, vehicleWeighttf1, vehicleColortf1, vehicleSpeedtf1, batteryCapacitytf1,

vehicleIDtf5, mileagetf, chargingTimetf, brandtf, rangetf, costPricetf, vehicleIDtf4, sellingPricetf;

private JComboBox year, month, day;

ArrayList<Vehicle> list = new ArrayList<Vehicle>();

// creating constructer

public TransportGUI() {

// JFrame for TransportGUI

frame1 = new JFrame("TransportGUI");// sets title of frame

frame1.setSize(600, 500); // sets the x-dimension, and y-dimension of frame

frame1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // Exit out of application

frame1.getContentPane().setBackground(new Color(204, 204, 204)); // change color of background

frame1.setLayout(null);

frame1.setLocationRelativeTo(null);

// Title for frame 1

Font titleFont = new Font(Font.SANS\_SERIF, Font.BOLD, 20);// Creating Object

title = new JLabel("'Booking system of vehicle'");

title.setForeground(Color.red);

title.setFont(titleFont);

title.setBounds(165, 20, 300, 100);

frame1.add(title);

// Title for Auto Rickshaw

titleAuto = new JLabel("Auto Rickshaw");

titleAuto.setFont(titleFont);

titleAuto.setBounds(205, 120, 300, 30);

frame1.add(titleAuto);

// Title for Electric Scooter

titleElectric = new JLabel("Electric Scooter");

titleElectric.setFont(titleFont);

titleElectric.setBounds(200, 220, 200, 100);

frame1.add(titleElectric);

// Creating Button For Auto Rickshaw

embarkBtn = new JButton("Embark Me A");

embarkBtn.setBounds(200, 150, 150, 30);

embarkBtn.setFocusPainted(false);

embarkBtn.setBackground(new Color(255, 0, 0));

embarkBtn.setForeground(new Color(255, 255, 255));// change button text color

embarkBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

frame1.add(embarkBtn);

// Creating Button For Electric Scooter

embarkBtn1 = new JButton("Embark Me E");

embarkBtn1.setBounds(200, 285, 150, 30);

embarkBtn1.setFocusPainted(false);

embarkBtn1.setBackground(new Color(255, 0, 0));

embarkBtn1.setForeground(new Color(255, 255, 255));// change button text color

embarkBtn1.setBorder(BorderFactory.createLineBorder(Color.black, 1));

frame1.add(embarkBtn1);

frame1.setVisible(true);

// Creating Frame 2

frame2 = new JFrame("AutoRickshaw");

frame2.setBounds(100, 20, 1100, 800);

frame2.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // Exit out of application

frame2.getContentPane().setBackground(new Color(204, 204, 204)); // change color of background

frame2.setLayout(null);

frame2.setLocationRelativeTo(null);

// Title for AutoRickshaw

titleAuto1 = new JLabel("Enter the details of AutoRickshaw");

titleAuto1.setForeground(Color.red);

titleAuto1.setFont(titleFont);

titleAuto1.setBounds(380, 10, 900, 100);

frame2.add(titleAuto1);

// Creating Panel 1

panel1 = new JPanel();

panel1.setBounds(15, 100, 1060, 235);

panel1.setBackground(new Color(102, 255, 102));

TitledBorder addAutoTitle = BorderFactory.createTitledBorder("Add an AutoRickshaw");

addAutoTitle.setTitleJustification(TitledBorder.CENTER);

addAutoTitle.setTitleFont(new Font(Font.SANS\_SERIF, Font.BOLD, 18));

addAutoTitle.setTitleColor(Color.red);

addAutoTitle.setBorder(BorderFactory.createLineBorder(Color.RED));

panel1.setBorder(addAutoTitle);

frame2.add(panel1);

panel1.setLayout(null);

// Adding on Panel 1

// Jlabel for vehicle

vehicleName = new JLabel("Vehicle Name");

panel1.add(vehicleName);

vehicleName.setBounds(20, 40, 200, 20);

// Input field of name

vehicleNametf = new JTextField();

vehicleNametf.setBounds(150, 40, 100, 30);

vehicleNametf.setBackground(new Color(221, 255, 255));

vehicleNametf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(vehicleNametf);

// JLabel for vehicle id

vehicleID = new JLabel("Vehicle Id");

panel1.add(vehicleID);

vehicleID.setBounds(20, 80, 200, 20);

// Input field of vehicle id

vehicleIDtf = new JTextField();

vehicleIDtf.setBounds(150, 80, 100, 30);

vehicleIDtf.setBackground(new Color(221, 255, 255));

vehicleIDtf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(vehicleIDtf);

// JLabel for vehicle weight

vehicleWeight = new JLabel("Vechile Weight");

panel1.add(vehicleWeight);

vehicleWeight.setBounds(20, 120, 200, 20);

// Input field of vehicle weight

vehicleWeighttf = new JTextField();

vehicleWeighttf.setBounds(150, 120, 100, 30);

vehicleWeighttf.setBackground(new Color(221, 255, 255));

vehicleWeighttf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(vehicleWeighttf);

// JLabel for vehicle Color

vehicleColor = new JLabel("Vehicle Color");

panel1.add(vehicleColor);

vehicleColor.setBounds(800, 40, 200, 20);

// Input field of vehicle weight

vehicleColortf = new JTextField();

vehicleColortf.setBounds(950, 40, 100, 30);

vehicleColortf.setBackground(new Color(221, 255, 255));

vehicleColortf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(vehicleColortf);

// JLabel for Torque

torque = new JLabel("Torque");

panel1.add(torque);

torque.setBounds(800, 80, 200, 20);

// Input field of torque

torquetf = new JTextField();

torquetf.setBounds(950, 80, 100, 30);

torquetf.setBackground(new Color(221, 255, 255));

torquetf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(torquetf);

// JLabel for Engine displacement

engineDisplacement = new JLabel("Engine Displacement");

panel1.add(engineDisplacement);

engineDisplacement.setBounds(400, 120, 340, 20);

// Input field of engine Displscement

engineDisplacementtf = new JTextField();

engineDisplacementtf.setBounds(540, 120, 100, 30);

engineDisplacementtf.setBackground(new Color(221, 255, 255));

engineDisplacementtf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(engineDisplacementtf);

// JLabel for Fuel Tank Capacity

fuelTankCapacity = new JLabel("Fuel Tank Capacity");

panel1.add(fuelTankCapacity);

fuelTankCapacity.setBounds(800, 120, 340, 20);

// Input field feul tank capacity

fuelTankCapacitytf = new JTextField();

fuelTankCapacitytf.setBounds(950, 120, 100, 30);

fuelTankCapacitytf.setBackground(new Color(221, 255, 255));

fuelTankCapacitytf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(fuelTankCapacitytf);

// JLabel for ground clearance

groundClearance = new JLabel("Ground Clearane");

groundClearance.setBounds(400, 40, 340, 20);

panel1.add(groundClearance);

// Input field for ground clearance

groundClearancetf = new JTextField();

groundClearancetf.setBounds(540, 40, 100, 30);

groundClearancetf.setBackground(new Color(221, 255, 255));

groundClearancetf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(groundClearancetf);

// JLabel for Fuel vehicle speed

vehicleSpeed = new JLabel("vehicle Speed");

vehicleSpeed.setBounds(400, 80, 200, 20);

panel1.add(vehicleSpeed);

// Input field for vehicle speed

vehicleSpeedtf = new JTextField();

vehicleSpeedtf.setBounds(540, 80, 100, 30);

vehicleSpeedtf.setBackground(new Color(221, 255, 255));

vehicleSpeedtf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel1.add(vehicleSpeedtf);

// Adding Buttons on pane1 of Add Auto Rickshaw

addAutoRickshawBtn = new JButton("Add AutoRickshaw");

addAutoRickshawBtn.setBounds(400, 170, 170, 30);

addAutoRickshawBtn.setFocusPainted(false);

addAutoRickshawBtn.setBackground(new Color(255, 0, 0));

addAutoRickshawBtn.setForeground(new Color(255, 255, 255));// change button text color

addAutoRickshawBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

panel1.add(addAutoRickshawBtn);

// Adding on Panel 2

// Creating Panel 2 for AutoRickshaw

panel2 = new JPanel();

panel2.setBounds(15, 350, 1060, 260);

panel2.setBackground(new Color(102, 255, 102));

TitledBorder addAutoTitle3 = BorderFactory.createTitledBorder("Book an AutoRickshaw");

addAutoTitle3.setTitleJustification(TitledBorder.CENTER);

addAutoTitle3.setTitleFont(new Font(Font.SANS\_SERIF, Font.BOLD, 18));

addAutoTitle3.setTitleColor(Color.red);

addAutoTitle3.setTitleJustification(TitledBorder.CENTER);

addAutoTitle3.setBorder(BorderFactory.createLineBorder(Color.RED));

panel2.setBorder(addAutoTitle3);

frame2.add(panel2);

panel2.setLayout(null);

// Adding on panel 2 on AutoRickshaw

// JLabel for vehicle id

vehicleId1 = new JLabel("Vehicle ID");

panel2.add(vehicleId1);

vehicleId1.setBounds(75, 60, 100, 20);

// Input feild of vehicle id

vehicleIdtf1 = new JTextField();

vehicleIdtf1.setBounds(75, 85, 120, 30);

vehicleIdtf1.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel2.add(vehicleIdtf1);

// JLabel for Number of Seat

noOfSeats = new JLabel("No of Seats");

noOfSeats.setBounds(400, 60, 200, 20);

panel2.add(noOfSeats);

// Input field of Number of Seat

noOfSeatstf = new JTextField();

noOfSeatstf.setBounds(400, 85, 120, 30);

noOfSeatstf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel2.add(noOfSeatstf);

// JLabel for Charge Amount

chargeAmount = new JLabel("Charge Amount");

chargeAmount.setBounds(800, 60, 100, 20);

panel2.add(chargeAmount);

// Input field of Charge Amount

chargeAmounttf = new JTextField();

chargeAmounttf.setBounds(800, 85, 120, 30);

chargeAmounttf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel2.add(chargeAmounttf);

String years[] = { "YYYY", "2020", "2021", "2022", "2023", "2024", "2025" };

String Months[] = { "MM", "01", "02", "03", "04", "05", "05", "06", "07", "08", "09", "10", "11", "12" };

String Days[] = { "DD", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14",

"15", "16",

"17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "28", "29", "30", "31" };

// JLabel for Combobox

bookDate = new JLabel("Date");

bookDate.setBounds(80, 130, 50, 30);

panel2.add(bookDate);

year = new JComboBox<>(years);

year.setBounds(80, 160, 60, 30);

panel2.add(year);

month = new JComboBox<>(Months);

month.setBounds(145, 160, 50, 30);

panel2.add(month);

day = new JComboBox<>(Days);

day.setBounds(200, 160, 50, 30);

panel2.add(day);

// Creating Buttons for Frame 2

// Adding Buttons on pane2 of Book an Auto Rickshaw

bookAutoRickshawBtn = new JButton("Book AutoRickshaw");

bookAutoRickshawBtn.setBounds(400, 160, 150, 30);

bookAutoRickshawBtn.setFocusPainted(false);

bookAutoRickshawBtn.setBackground(new Color(255, 0, 0));

bookAutoRickshawBtn.setForeground(new Color(255, 255, 255));// change button text color

bookAutoRickshawBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

panel2.add(bookAutoRickshawBtn);

// Adding Display button on Frame 2

displayBtn = new JButton("Display");

displayBtn.setBounds(60, 620, 150, 30);

displayBtn.setFocusPainted(false);

displayBtn.setBackground(new Color(255, 0, 0));

displayBtn.setForeground(new Color(255, 255, 255));// change button text color

displayBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

frame2.add(displayBtn);

// Adding Clear button on Frame 2

clearBtn = new JButton("Clear");

clearBtn.setBounds(500, 620, 150, 30);

clearBtn.setFocusPainted(false);

clearBtn.setBackground(new Color(255, 0, 0));

clearBtn.setForeground(new Color(255, 255, 255));// change button text color

clearBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

frame2.add(clearBtn);

// Adding Back button on Frame 2

backBtn = new JButton("Back");

backBtn.setBounds(910, 620, 150, 30);

backBtn.setFocusPainted(false);

backBtn.setBackground(new Color(255, 0, 0));

backBtn.setForeground(new Color(255, 255, 255));// change button text color

backBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

frame2.add(backBtn);

// Creating Frame for Electric Scooter

frame3 = new JFrame("ElectricScooter");

frame3.setBounds(100, 20, 780, 650);

frame3.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // Exit out of application

frame3.getContentPane().setBackground(new Color(204, 204, 204)); // change color of background

frame3.setLayout(null);

frame3.setLocationRelativeTo(null);

// Title for ElectricScooter

titleAuto2 = new JLabel("Enter the details of ElectricScooter");

titleAuto2.setForeground(Color.red);

titleAuto2.setFont(titleFont);

titleAuto2.setBounds(200, 10, 900, 100);

frame3.add(titleAuto2);

// Creating Panel 3

panel3 = new JPanel();

panel3.setBounds(20, 100, 320, 290);

panel3.setBackground(new Color(102, 255, 102));

TitledBorder addScooterTitle = BorderFactory.createTitledBorder("Add an Electric Scooter");

addScooterTitle.setTitleJustification(TitledBorder.CENTER);

addScooterTitle.setTitleFont(new Font(Font.SANS\_SERIF, Font.BOLD, 18));

addScooterTitle.setTitleColor(Color.red);

addScooterTitle.setTitleJustification(TitledBorder.CENTER);

addScooterTitle.setBorder(BorderFactory.createLineBorder(Color.RED));

panel3.setBorder(addScooterTitle);

panel3.setLayout(null);

frame3.add(panel3);

// Adding on Panel 3 on ElectricScooter

// JLabel for Vehicle Name

vehicleName1 = new JLabel("Vehicle Name");

vehicleName1.setBounds(20, 40, 200, 20);

panel3.add(vehicleName1);

// Input field of name

vehicleNametf1 = new JTextField();

vehicleNametf1.setBounds(20, 60, 100, 25);

vehicleNametf1.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel3.add(vehicleNametf1);

// JLabel for vehicle id

vehicleID2 = new JLabel("Vehicle Id");

vehicleID2.setBounds(20, 100, 200, 20);

panel3.add(vehicleID2);

// Input field of vehicle id

vehicleIDtf2 = new JTextField();

vehicleIDtf2.setBounds(20, 120, 100, 25);

vehicleIDtf2.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel3.add(vehicleIDtf2);

// JLabel for vehicle weight

vehicleWeight1 = new JLabel("Vehicle Weight");

vehicleWeight1.setBounds(20, 160, 100, 20);

panel3.add(vehicleWeight1);

// Input field of vehicle weight

vehicleWeighttf1 = new JTextField();

vehicleWeighttf1.setBounds(20, 180, 100, 25);

vehicleWeighttf1.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel3.add(vehicleWeighttf1);

// JLabel for vehicle color

vehicleColor1 = new JLabel("Vehicle Color");

vehicleColor1.setBounds(180, 40, 400, 20);

panel3.add(vehicleColor1);

// Input feild of vehicle color

vehicleColortf1 = new JTextField();

vehicleColortf1.setBounds(180, 60, 100, 25);

vehicleColortf1.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel3.add(vehicleColortf1);

// JLabel for Battery capacity

batteryCapacity1 = new JLabel("Battery Capacity");

batteryCapacity1.setBounds(180, 100, 340, 20);

panel3.add(batteryCapacity1);

// Input field for Battery capacity

batteryCapacitytf1 = new JTextField();

batteryCapacitytf1.setBounds(180, 120, 100, 25);

batteryCapacitytf1.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel3.add(batteryCapacitytf1);

// JLabel for Fuel vehicle speed

vehicleSpeed1 = new JLabel("vehicle Speed");

vehicleSpeed1.setBounds(180, 160, 200, 20);

panel3.add(vehicleSpeed1);

// Input feild for vehicle speed

vehicleSpeedtf1 = new JTextField();

vehicleSpeedtf1.setBounds(180, 180, 100, 25);

vehicleSpeedtf1.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel3.add(vehicleSpeedtf1);

// Adding Buttons on pane3 of Add ElectricScooter

addElectricScooterBtn = new JButton("Add ElectricScooter");

addElectricScooterBtn.setBounds(80, 220, 170, 30);

addElectricScooterBtn.setFocusPainted(false);

addElectricScooterBtn.setBackground(new Color(255, 0, 0));

addElectricScooterBtn.setForeground(new Color(255, 255, 255));// change button text color

addElectricScooterBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

panel3.add(addElectricScooterBtn);

// Creating Panel 4

panel4 = new JPanel();

panel4.setBounds(20, 400, 320, 200);

panel4.setBackground(new Color(102, 255, 102));

TitledBorder addScooterTitle4 = BorderFactory.createTitledBorder("Sell Electric Scooter");

addScooterTitle4.setTitleJustification(TitledBorder.CENTER);

addScooterTitle4.setTitleFont(new Font(Font.SANS\_SERIF, Font.BOLD, 18));

addScooterTitle4.setTitleColor(Color.red);

addScooterTitle4.setTitleJustification(TitledBorder.CENTER);

addScooterTitle4.setBorder(BorderFactory.createLineBorder(Color.RED));

panel4.setBorder(addScooterTitle4);

panel4.setLayout(null);

frame3.add(panel4);

// Adding on Panel 4 Sell Electric Scooter

// JLabel for Vehicle ID

vehicleID4 = new JLabel("Vehicle ID");

vehicleID4.setBounds(20, 50, 340, 20);

panel4.add(vehicleID4);

// Input field for Vehicle ID

vehicleIDtf4 = new JTextField();

vehicleIDtf4.setBounds(20, 75, 100, 25);

vehicleIDtf4.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel4.add(vehicleIDtf4);

// JLabel for Mileage

sellingPrice = new JLabel("Selling Price");

sellingPrice.setBounds(175, 50, 340, 20);

panel4.add(sellingPrice);

// Input field for Mileage

sellingPricetf = new JTextField();

sellingPricetf.setBounds(175, 75, 100, 25);

sellingPricetf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel4.add(sellingPricetf);

// Adding Buttons on pane 4 of Sell ElectricScooter

sellScooterBtn = new JButton("Sell Scooter");

sellScooterBtn.setBounds(70, 120, 150, 30);

sellScooterBtn.setFocusPainted(false);

sellScooterBtn.setBackground(new Color(255, 0, 0));

sellScooterBtn.setForeground(new Color(255, 255, 255));// change button text color

sellScooterBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

panel4.add(sellScooterBtn);

// Creating Panel 5

panel5 = new JPanel();

panel5.setBounds(350, 100, 390, 290);

panel5.setBackground(new Color(102, 255, 102));

TitledBorder addScooterTitle5 = BorderFactory.createTitledBorder("Purchase Electric Scooter");

addScooterTitle5.setTitleJustification(TitledBorder.CENTER);

addScooterTitle5.setTitleFont(new Font(Font.SANS\_SERIF, Font.BOLD, 18));

addScooterTitle5.setTitleColor(Color.red);

addScooterTitle5.setTitleJustification(TitledBorder.CENTER);

addScooterTitle5.setBorder(BorderFactory.createLineBorder(Color.RED));

panel5.setBorder(addScooterTitle5);

panel5.setLayout(null);

frame3.add(panel5);

// Adding on Panel 5 of Purchase Electric Scooter

// JLabel for Vehicle ID

vehicleID5 = new JLabel("Vehicle ID");

vehicleID5.setBounds(30, 40, 340, 20);

panel5.add(vehicleID5);

// Input field for Vehicle ID

vehicleIDtf5 = new JTextField();

vehicleIDtf5.setBounds(30, 60, 100, 25);

vehicleIDtf5.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel5.add(vehicleIDtf5);

// JLabel for Mileage

mileage = new JLabel("Mileage");

mileage.setBounds(30, 100, 340, 20);

panel5.add(mileage);

// Input field for Mileage

mileagetf = new JTextField();

mileagetf.setBounds(30, 120, 100, 25);

mileagetf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel5.add(mileagetf);

// JLabel for Charging Time

chargingTime = new JLabel("Charging Time");

chargingTime.setBounds(30, 160, 340, 20);

panel5.add(chargingTime);

// Input field for Charging Time

chargingTimetf = new JTextField();

chargingTimetf.setBounds(30, 180, 100, 25);

chargingTimetf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel5.add(chargingTimetf);

// JLabel for Brand

brand = new JLabel("Brand");

brand.setBounds(200, 40, 340, 20);

panel5.add(brand);

// Input field for Brand

brandtf = new JTextField();

brandtf.setBounds(200, 60, 100, 25);

brandtf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel5.add(brandtf);

// JLabel for Range

range = new JLabel("Range");

range.setBounds(200, 100, 340, 20);

panel5.add(range);

// Input field for Range

rangetf = new JTextField();

rangetf.setBounds(200, 120, 100, 25);

rangetf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel5.add(rangetf);

// JLabel for Cost Price

costPrice = new JLabel("Cost Price");

costPrice.setBounds(200, 160, 340, 20);

panel5.add(costPrice);

// Input field for Cost Price

costPricetf = new JTextField();

costPricetf.setBounds(200, 180, 100, 25);

costPricetf.setBorder(BorderFactory.createMatteBorder(0, 0, 3, 2, Color.black));

panel5.add(costPricetf);

// Creating Button of Frame 3 and Panel 5

// Adding button on Purchase Electric Scooter

purchaseScooterBtn = new JButton("Purchase Scooter");

purchaseScooterBtn.setBounds(100, 220, 150, 30);

purchaseScooterBtn.setFocusPainted(false);

purchaseScooterBtn.setBackground(new Color(255, 0, 0));

purchaseScooterBtn.setForeground(new Color(255, 255, 255));// change button text color

purchaseScooterBtn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

panel5.add(purchaseScooterBtn);

// Creating Buttons for Frame 3

// Adding Display 1 button on Frame 3

displayBtn1 = new JButton("Display");

displayBtn1.setBounds(450, 520, 150, 25);

displayBtn1.setFocusPainted(false);

displayBtn1.setBackground(new Color(255, 0, 0));

displayBtn1.setForeground(new Color(255, 255, 255));// change button text color

displayBtn1.setBorder(BorderFactory.createLineBorder(Color.black, 1));

frame3.add(displayBtn1);

// Adding Clear 1 button on Frame 3

clear1Btn = new JButton("Clear");

clear1Btn.setBounds(400, 480, 100, 25);

clear1Btn.setFocusPainted(false);

clear1Btn.setBackground(new Color(255, 0, 0));

clear1Btn.setForeground(new Color(255, 255, 255));// change button text color

clear1Btn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

frame3.add(clear1Btn);

// Adding Back 1 button on Frame 3

back1Btn = new JButton("Back");

back1Btn.setBounds(540, 480, 100, 25);

back1Btn.setFocusPainted(false);

back1Btn.setBackground(new Color(255, 0, 0));

back1Btn.setForeground(new Color(255, 255, 255));// change button text color

back1Btn.setBorder(BorderFactory.createLineBorder(Color.black, 1));

frame3.add(back1Btn);

// ActionListeners For Embark A

embarkBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

frame1.setVisible(false);

frame2.setVisible(true);

}

});

// ActionListeners for Embark E

embarkBtn1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

frame1.setVisible(false);

frame3.setVisible(true);

}

});

// ActionListeners for Add AutoRickshaw

addAutoRickshawBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addAuto();

}

});

// ActionListeners for Back Button

backBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

frame1.setVisible(true);

frame2.setVisible(false);

}

});

// ActionListeners for Back Button 1

back1Btn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

frame1.setVisible(false);

frame2.setVisible(true);

}

});

// ActionListeners for Display Button

displayBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

for (Vehicle disp : list) {

((AutoRickshaw) disp).display();

}

}

});

// ActionListeners for Display Button 1

displayBtn1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

for (Vehicle disp : list) {

if (disp instanceof ElectricScooter) {

((ElectricScooter) disp).display();

}

}

}

});

// ActionListeners for Clear Button

clearBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

vehicleIDtf.setText("");

vehicleNametf.setText("");

vehicleWeighttf.setText("");

groundClearancetf.setText("");

vehicleSpeedtf.setText("");

engineDisplacementtf.setText("");

vehicleColortf.setText("");

torquetf.setText("");

fuelTankCapacitytf.setText("");

vehicleIdtf1.setText("");

noOfSeatstf.setText("");

chargeAmounttf.setText("");

year.setSelectedItem("YYYY");

month.setSelectedItem("MM");

day.setSelectedItem("DD");

}

});

// ActionListeners for Clear Button 1

clear1Btn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

vehicleIDtf2.setText("");

vehicleNametf1.setText("");

vehicleWeighttf1.setText("");

vehicleColortf1.setText("");

batteryCapacitytf1.setText("");

vehicleSpeedtf1.setText("");

vehicleIDtf4.setText("");

brandtf.setText("");

mileagetf.setText("");

rangetf.setText("");

chargingTimetf.setText("");

costPricetf.setText("");

vehicleIDtf5.setText("");

sellingPricetf.setText("");

}

});

// ActionListnear for Book AutoRickshaw

bookAutoRickshawBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

bookAuto();

}

});

// ActionListeners for Add ElectricScooter

addElectricScooterBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addElectric();

}

});

// ActionListeners for Purchase Scooter

purchaseScooterBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

purchaseElectric();

}

});

// ActionListeners for Sell Electric Scooter

sellScooterBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

sellElectric();

}

});

}

// Declearing AutoRickhaw Panel 1

public final static int INVALID = -1;

public final static int EMPTY = -2;

public int getAutoVehicleIdAdd() {

int vehicleId = INVALID;

try {

if (vehicleIDtf.getText().equals("")) {

vehicleId = EMPTY;

} else {

vehicleId = Integer.parseInt(vehicleIDtf.getText());

if (vehicleId <= 0) {

vehicleId = INVALID;

}

}

} catch (NumberFormatException e) {

}

return vehicleId;

}

// Declearing Vehicle Name

public String getAutoVehicleName() {

String vehicleName = vehicleNametf.getText();

return vehicleName;

}

// Declearing Vehicle Weight

public String getAutoVehicleWeight() {

String vehicleWeight = vehicleWeighttf.getText();

return vehicleWeight;

}

// Declearing Vehicle Color

public String getAutoVehicleColor() {

String vehicleColor = vehicleColortf.getText();

return vehicleColor;

}

// Declearing Vehicle Speed

public String getAutoVehicleSpeed() {

String vehicleSpeed = vehicleSpeedtf.getText();

return vehicleSpeed;

}

// Declearing Engine Displacement

public int getEngineDisplacement() {

int engineDisplacement = INVALID;

try {

if (engineDisplacementtf.getText().equals("")) {

engineDisplacement = EMPTY;

} else {

engineDisplacement = Integer.parseInt(engineDisplacementtf.getText());

if (engineDisplacement <= 0) {

engineDisplacement = INVALID;

}

}

} catch (NumberFormatException e) {

}

return engineDisplacement;

}

// Declearing Fuel TankCapacity

public int getAutoFuelTankCapacity() {

int fuelTankCapacity = INVALID;

try {

if (fuelTankCapacitytf.getText().equals("")) {

fuelTankCapacity = EMPTY;

} else {

fuelTankCapacity = Integer.parseInt(fuelTankCapacitytf.getText());

if (fuelTankCapacity <= 0) {

fuelTankCapacity = INVALID;

}

}

} catch (NumberFormatException e) {

}

return fuelTankCapacity;

}

// Declearing Ground Clearance

public String getAutoGroundClearance() {

String GroundClearance = groundClearancetf.getText();

return GroundClearance;

}

// Declearing Torque

public String getAutoTorque() {

String torque = torquetf.getText();

return torque;

}

// Declearing For AutoRickhaw in Panel 2

// Declearing VehicleId1

public int getAutoVehicleId1Book() {

int vehicleId1 = INVALID;

try {

if (vehicleIdtf1.getText().equals("")) {

vehicleId1 = EMPTY;

} else {

vehicleId1 = Integer.parseInt(vehicleIdtf1.getText());

if (vehicleId1 <= 0) {

vehicleId1 = INVALID;

}

}

} catch (NumberFormatException e) {

}

return vehicleId1;

}

public int getAutovehicleId1() {

int vehicleId1 = INVALID;

try {

if (vehicleIdtf1.getText().equals("")) {

vehicleId1 = EMPTY;

} else {

vehicleId1 = Integer.parseInt(vehicleIdtf1.getText());

if (vehicleId1 <= 0) {

vehicleId1 = INVALID;

}

}

} catch (NumberFormatException e) {

}

return vehicleId1;

}

// Declearing No of Seat

public int getAutoNoOfSeat() {

int noOfSeats = INVALID;

try {

if (noOfSeatstf.getText().equals("")) {

noOfSeats = EMPTY;

} else {

noOfSeats = Integer.parseInt(noOfSeatstf.getText());

if (noOfSeats <= 0) {

noOfSeats = INVALID;

}

}

} catch (NumberFormatException e) {

}

return noOfSeats;

}

public int getAutonoOfSeats() {

int noOfSeats = INVALID;

try {

if (noOfSeatstf.getText().equals("")) {

noOfSeats = EMPTY;

} else {

noOfSeats = Integer.parseInt(noOfSeatstf.getText());

if (noOfSeats <= 0) {

noOfSeats = INVALID;

}

}

} catch (NumberFormatException e) {

}

return noOfSeats;

}

// Declearing Charge Amount

public int getAutoChargeAmount() {

int ChargeAmount = INVALID;

try {

if (chargeAmounttf.getText().equals("")) {

ChargeAmount = EMPTY;

} else {

ChargeAmount = Integer.parseInt(chargeAmounttf.getText());

if (ChargeAmount <= 0) {

ChargeAmount = INVALID;

}

}

} catch (NumberFormatException e) {

}

return ChargeAmount;

}

public int getAutochargeAmount() {

int ChargeAmount = INVALID;

try {

if (vehicleIdtf1.getText().equals("")) {

ChargeAmount = EMPTY;

} else {

ChargeAmount = Integer.parseInt(chargeAmounttf.getText());

if (ChargeAmount <= 0) {

ChargeAmount = INVALID;

}

}

} catch (NumberFormatException e) {

}

return ChargeAmount;

}

// Declearing Date

public String getYear() {

String y = year.getSelectedItem().toString();

return y;

}

public String getMonth() {

String m = month.getSelectedItem().toString();

return m;

}

public String getday() {

String d = day.getSelectedItem().toString();

return d;

}

public String getBookedDate() {

String bookedDate = getYear() + "-" + getMonth() + "-" + getday();

return bookedDate;

}

// Declearing For Electric Scooter Panel 3

// Declearing VehicleID 2

public int getAutoVehicleId2() {

int vehicleID2 = INVALID;

try {

if (vehicleIDtf2.getText().equals("")) {

vehicleID2 = EMPTY;

} else {

vehicleID2 = Integer.parseInt(vehicleIDtf2.getText());

if (vehicleID2 <= 0) {

vehicleID2 = INVALID;

}

}

} catch (NumberFormatException e) {

}

return vehicleID2;

}

public int getAutovehicleID2() {

int vehicleID2 = INVALID;

try {

if (vehicleIDtf2.getText().equals("")) {

vehicleID2 = EMPTY;

} else {

vehicleID2 = Integer.parseInt(vehicleIDtf2.getText());

if (vehicleID2 <= 0) {

vehicleID2 = INVALID;

}

}

} catch (NumberFormatException e) {

}

return vehicleID2;

}

// Declearing VehicleName

public String getAutoVehicleName1() {

String VehicleName1 = vehicleNametf1.getText();

return VehicleName1;

}

// Declearing Vehicle Color

public String getAutoVehicleColor1() {

String vehicleColor1 = vehicleColortf1.getText();

return vehicleColor1;

}

// Declearing Vehicle Weight

public String getAutoVehicleWeight1() {

String vehicleWeight1 = vehicleWeighttf1.getText();

return vehicleWeight1;

}

// Declearing Vehicle Speed

public String getAutoVehicleSpeed1() {

String vehicleSpeed1 = vehicleSpeedtf1.getText();

return vehicleSpeed1;

}

// Declearing BatteryCapacity

public int getAutoBatteryCapacity() {

int batteryCapacity1 = INVALID;

try {

if (batteryCapacitytf1.getText().equals("")) {

batteryCapacity1 = EMPTY;

} else {

batteryCapacity1 = Integer.parseInt(batteryCapacitytf1.getText());

if (batteryCapacity1 <= 0) {

batteryCapacity1 = INVALID;

}

}

} catch (NumberFormatException e) {

}

return batteryCapacity1;

}

public int getAutobatteryCapacity() {

int batteryCapacity1 = INVALID;

try {

if (batteryCapacitytf1.getText().equals("")) {

batteryCapacity1 = EMPTY;

} else {

batteryCapacity1 = Integer.parseInt(batteryCapacitytf1.getText());

if (batteryCapacity1 <= 0) {

batteryCapacity1 = INVALID;

}

}

} catch (NumberFormatException e) {

}

return batteryCapacity1;

}

// Declearing For Electric Scooter Panel 4

// Declearing VehicleID 3

public int getAutoVehicleId3() {

int vehicleID3 = INVALID;

try {

if (vehicleIDtf5.getText().equals("")) {

vehicleID3 = EMPTY;

} else {

vehicleID3 = Integer.parseInt(vehicleIDtf5.getText());

if (vehicleID3 <= 0) {

vehicleID3 = INVALID;

}

}

} catch (NumberFormatException e) {

}

return vehicleID3;

}

// Declearing Mileage

public String getAutoMileage() {

String mileage = mileagetf.getText();

return mileage;

}

// Declearing Brand

public String getAutoBrand() {

String brand = brandtf.getText();

return brand;

}

// Declearing Range

public int getAutoRange() {

int range = INVALID;

try {

if (rangetf.getText().equals("")) {

range = EMPTY;

} else {

range = Integer.parseInt(rangetf.getText());

if (range <= 0) {

range = INVALID;

}

}

} catch (NumberFormatException e) {

}

return range;

}

public int getAutorange() {

int range = INVALID;

try {

if (rangetf.getText().equals("")) {

range = EMPTY;

} else {

range = Integer.parseInt(rangetf.getText());

if (range <= 0) {

range = INVALID;

}

}

} catch (NumberFormatException e) {

}

return range;

}

// Declearing CostPrice

public int getAutoCostPrice() {

int costPrice = INVALID;

try {

if (costPricetf.getText().equals("")) {

costPrice = EMPTY;

} else {

costPrice = Integer.parseInt(costPricetf.getText());

if (costPrice <= 0) {

costPrice = INVALID;

}

}

} catch (NumberFormatException e) {

}

return costPrice;

}

public int getAutocostPrice() {

int costPrice = INVALID;

try {

if (costPricetf.getText().equals("")) {

costPrice = EMPTY;

} else {

costPrice = Integer.parseInt(costPricetf.getText());

if (costPrice <= 0) {

costPrice = INVALID;

}

}

} catch (NumberFormatException e) {

}

return costPrice;

}

// Declearing Charging Time

public String getAutoChargingTime() {

String chargingTime = chargingTimetf.getText();

return chargingTime;

}

// Declearing For Electric Scooter Panel 5

// Declearing VehicleID 5

public int getAutoVehicleID5() {

int vehicleID5 = INVALID;

try {

if (vehicleIDtf4.getText().equals("")) {

vehicleID5 = EMPTY;

} else {

vehicleID5 = Integer.parseInt(vehicleIDtf4.getText());

if (vehicleID5 <= 0) {

vehicleID5 = INVALID;

}

}

} catch (NumberFormatException e) {

}

return vehicleID5;

}

// Declearing Selling Price

public int getAutoSellingPrice() {

int sellingPrice = INVALID;

try {

if (sellingPricetf.getText().equals("")) {

sellingPrice = EMPTY;

} else {

sellingPrice = Integer.parseInt(sellingPricetf.getText());

if (sellingPrice <= 0) {

sellingPrice = INVALID;

}

}

} catch (NumberFormatException e) {

}

return sellingPrice;

}

public int getAutosellingPrice() {

int sellingPrice = INVALID;

try {

if (sellingPricetf.getText().equals("")) {

sellingPrice = EMPTY;

} else {

sellingPrice = Integer.parseInt(sellingPricetf.getText());

if (sellingPrice <= 0) {

sellingPrice = INVALID;

}

}

} catch (NumberFormatException e) {

}

return sellingPrice;

}

// Creating Unique ID

public boolean unique(int ID) {

boolean unique = true;

for (Vehicle vehicleId : list) {

if (vehicleId.getvehicleID() == ID) {

unique = false;

}

}

return unique;

}

// Getter method of Auto Rickhaw and Electric Scooter

// Getter method of Add AutoRickshaw

public void addAuto() {

System.out.println(list.size());

if (getAutoVehicleIdAdd() == EMPTY || getAutoVehicleName().equals("") || getAutoVehicleWeight().equals("")

|| getAutoVehicleColor().equals("") || getAutoVehicleSpeed().equals("")

|| getEngineDisplacement() == EMPTY || getAutoFuelTankCapacity() == EMPTY

|| getAutoGroundClearance().equals("")) {

JOptionPane.showMessageDialog(frame1, "Fill all the data of Auto Rickshaw", "Error",

JOptionPane.ERROR\_MESSAGE);

return;

} else if (getAutoVehicleIdAdd() == INVALID) {

JOptionPane.showMessageDialog(frame1, "Invalid Integer", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

if (unique(getAutoVehicleIdAdd())) {

list.add(new AutoRickshaw(getAutoVehicleIdAdd(), getAutoVehicleName(), getAutoTorque(),

getAutoVehicleWeight(), getAutoVehicleColor(), getAutoVehicleSpeed(), getAutoGroundClearance(),

getEngineDisplacement(), getAutoFuelTankCapacity()));

JOptionPane.showMessageDialog(frame1, "Successfully Added your Auto Rickshaw data",

"Auto Rickshaw data is Booked", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(frame1, "Enter Unique Id of Auto Rickshaw data", "Auto Rickshaw Data",

JOptionPane.INFORMATION\_MESSAGE);

}

}

// Getter method of Book AutoRickshaw

public void bookAuto() {

if (getAutoVehicleId1Book() == EMPTY || getAutoNoOfSeat() == EMPTY || getAutoChargeAmount() == EMPTY

|| getday().equals("DD") || getMonth().equals("MM") || getYear().equals("YYYY")) {

JOptionPane.showMessageDialog(frame1, "Fill all the data", "Error", JOptionPane.ERROR\_MESSAGE);

return;

} else if (getAutoVehicleId1Book() == INVALID || getAutoNoOfSeat() == INVALID

|| getAutoChargeAmount() == INVALID) {

JOptionPane.showMessageDialog(frame1, "Invalid Integer", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

boolean book = false;

boolean found = false;

for (Vehicle id : list) {

if (id instanceof AutoRickshaw) {

if (id.getvehicleID() == getAutoVehicleId1Book()) {

found = true;

if (((AutoRickshaw) id).getIsBooked() == false) {

book = true;

((AutoRickshaw) id).booking(getBookedDate(), getAutoChargeAmount(), getAutoNoOfSeat());

JOptionPane.showMessageDialog(frame1, "Your Auto Rickshaw is Booked.", "Succesfull",

JOptionPane.INFORMATION\_MESSAGE);

} else if (((AutoRickshaw) id).getIsBooked() == true) {

book = false;

JOptionPane.showMessageDialog(frame1, "Your Auto Rickshaw is already booked.",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

} else if (id.getvehicleID() == getAutoVehicleId1Book()) {

found = true;

book = false;

JOptionPane.showMessageDialog(frame1, "Invalid vehicle Id",

"Error",

JOptionPane.ERROR\_MESSAGE);

}

}

if (!book && !found) {

JOptionPane.showMessageDialog(frame1, "Your vehicle id is not added",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

// Getter method for Frame 2 Add Electric Scooter

// Add an Electric Scooter

public void addElectric() {

if (getAutoVehicleId2() == EMPTY || getAutoVehicleName1().isEmpty() || getAutoVehicleColor1().isEmpty()

|| getAutoVehicleWeight1().isEmpty() || getAutoVehicleSpeed1().isEmpty()

|| getAutoBatteryCapacity() == EMPTY) {

JOptionPane.showMessageDialog(frame2, "Fill all the data of Electric Scooter", "Error",

JOptionPane.ERROR\_MESSAGE);

return;

} else if (getAutoVehicleId2() == EMPTY || getAutoVehicleName1().isEmpty() || getAutoVehicleColor1().isEmpty()

|| getAutoVehicleWeight1().isEmpty() || getAutoVehicleSpeed1().isEmpty()

|| getAutoBatteryCapacity() == EMPTY) {

JOptionPane.showMessageDialog(frame2, "Invalid Integer", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

if (unique(getAutoVehicleId2())) {

list.add(new Vehicle(getAutoVehicleId2(), getAutoVehicleName1(), getAutoVehicleColor1(),

getAutoVehicleWeight1()));

JOptionPane.showMessageDialog(frame2, "Successfully Added your Electric Scooter data", "Congratulations!",

JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(frame2, "Enter Unique Id of Electric Scooter data!", "Electric Scooter",

JOptionPane.INFORMATION\_MESSAGE);

}

}

// Getter method for Purchase Electric Scooter

public void purchaseElectric() {

if (getAutoVehicleId3() == EMPTY || getAutoMileage().isEmpty() || getAutoBrand().isEmpty()

|| getAutoRange() == EMPTY || getAutoCostPrice() == EMPTY || getAutoChargingTime().isEmpty()) {

JOptionPane.showMessageDialog(frame1, "Fill all the data of Purchase Electric Scooter", "Error!",

JOptionPane.ERROR\_MESSAGE);

return;

} else if (getAutoVehicleId3() == INVALID || getAutoRange() == INVALID || getAutoCostPrice() == INVALID) {

JOptionPane.showMessageDialog(frame2, "Invalid Integer", "Error!", JOptionPane.ERROR\_MESSAGE);

return;

}

boolean purchase = false;

boolean found = false;

for (Vehicle id : list) {

if (id instanceof ElectricScooter) {

if (id.getvehicleID() == getAutoVehicleId3()) {

found = true;

if (((ElectricScooter) id).getHasPurchased() == false) {

purchase = true;

((AutoRickshaw) id).booking(getBookedDate(), getAutoChargeAmount(), getAutoNoOfSeat());

JOptionPane.showMessageDialog(frame1, "Your Electric Scooter is Purchased.", "Succesfull",

JOptionPane.INFORMATION\_MESSAGE);

} else if (((AutoRickshaw) id).getIsBooked() == true) {

purchase = false;

JOptionPane.showMessageDialog(frame1, "Your ElectricScooter is already Purchased.",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

if (!purchase && !found) {

JOptionPane.showMessageDialog(frame1, "Your vehicle id is not added",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

// Getter method for sell Electrric Scooter

public void sellElectric() {

if (getAutoVehicleID5() == EMPTY || getAutoSellingPrice() == EMPTY) {

JOptionPane.showMessageDialog(frame1, "Fill all the data of Sell Electric Scooter", "Error!",

JOptionPane.ERROR\_MESSAGE);

return;

} else if (getAutoVehicleID5() == EMPTY || getAutoSellingPrice() == EMPTY) {

JOptionPane.showMessageDialog(frame2, "Invalid Integer", "Error!", JOptionPane.ERROR\_MESSAGE);

return;

}

boolean sell = false;

boolean found = false;

for (Vehicle id : list) {

if (id instanceof ElectricScooter) {

if (id.getvehicleID() == getAutoVehicleId1Book()) {

found = true;

if (((ElectricScooter) id).getHasSold() == false) {

sell = true;

((ElectricScooter) id).purchasedScooter(getAutoBrand(), getAutoCostPrice(),

getAutoChargingTime(), getAutoMileage(), getAutoRange());

JOptionPane.showMessageDialog(frame1, "Your Electric Scooter is Sold.", "Succesfull",

JOptionPane.INFORMATION\_MESSAGE);

} else if (((ElectricScooter) id).getHasSold() == true) {

sell = false;

JOptionPane.showMessageDialog(frame1, "Your ElectricScooter is already Sold.",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

} else if (id.getvehicleID() == getAutoVehicleId1Book()) {

found = true;

sell = false;

JOptionPane.showMessageDialog(frame2, "Invalid vehicle Id",

"Error",

JOptionPane.ERROR\_MESSAGE);

}

}

if (!sell && !found) {

JOptionPane.showMessageDialog(frame2, "Your vehicle id is not added",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

// Main method

public static void main(String[] args) {

new TransportGUI();

}

}