Vehicle Information System - Assignment Documentation

Student Name: Edmund Asamoah Adomako

Course Title: Programming 1

**Title: Vehicle Information System Using Interfaces in Java**

**1. Introduction**

This project demonstrates the implementation of Java interfaces to create a modular and extensible Vehicle Information System for a car rental agency. It handles different types of vehicles such as Cars, Motorcycles, and Trucks. By enforcing a common structure through interfaces while allowing specific attributes for each vehicle type.

The aim is to provide a flexible and user-friendly software application that makes it easy to input and retrieve vehicle data interactively.

**2. Objectives**

* Implement interfaces to define behavior contracts for vehicle classes.
* Build classes that represent Cars, Motorcycles, and Trucks.
* Handle user input interactively and validate entries.
* Follow Java best practices for clean, maintainable code.
* Implement basic exception handling for input errors.
* Provide well-structured documentation to aid understanding.

**3. System Design**

Interfaces Implemented

**1. Vehicle**

Common to all vehicle types

Methods:

* String getMake()
* String getModel()
* int getYear()

**2. CarVehicle**

Specific to cars

Methods:

* void setNumDoors (int numDoors) / int getNumDoors ()
* void setFuelType (String fuelType) / String getFuelType ()

**3. MotorVehicle**

Specific to motorcycles

Methods:

* void setNumWheels (int numWheels) / int getNumWheels ()
* void setType(String type) / String getType()

**4.TruckVehicle**

Specific to trucks

Methods:

* void setCargoCapacity (double capacity) / double getCargoCapacity ()
* void setTransmissionType (String transmission) / String getTransmissionType ()

**4. Classes Implemented**

**1. Car**

* Implements CarVehicle
* Attributes: make, model, year, doors, fuel

**2. Motorcycle**

* Implements MotorVehicle
* Attributes: make, model, year, wheels, type

**3. Truck**

* Implements TruckVehicle
* Attributes: make, model, year, capacity, transmission

All classes provide constructors and getter/setter methods to interact with these attributes.

**5. Interface Definitions**

The system defines four interfaces:

* **Vehicle Interface:**

1. getMake()

2. getModel()

3. getYear()

* **CarVehicle Interface:**

4. setNumDoors(int numDoors)

5. getNumDoors()

6. setFuelType(String fuelType)

7. getFuelType()

* **MotorVehicle Interface:**

8. setNumWheels(int numWheels)

9. getNumWheels()

10. setType(String type)

11. getType()

* **TruckVehicle Interface:**

12. setCargoCapacity(double capacity)

13. getCargoCapacity()

14. setTransmissionType(String transmission)

15. getTransmissionType()

**6. Vehicle Class Implementations**

Each class implements the Vehicle interface and its specific vehicle-type interface.

* **Car Class:**

16. Implements Vehicle and CarVehicle interfaces

17. Has fields: make, model, year, numDoors, fuelType

18. Includes a toString method for display

* **Motorcycle Class:**

19. Implements Vehicle and MotorVehicle interfaces

20. Has fields: make, model, year, numWheels, type

21. Includes a toString method for display

* **Truck Class:**

22. Implements Vehicle and TruckVehicle interfaces

23. Has fields: make, model, year, cargoCapacity, transmissionType

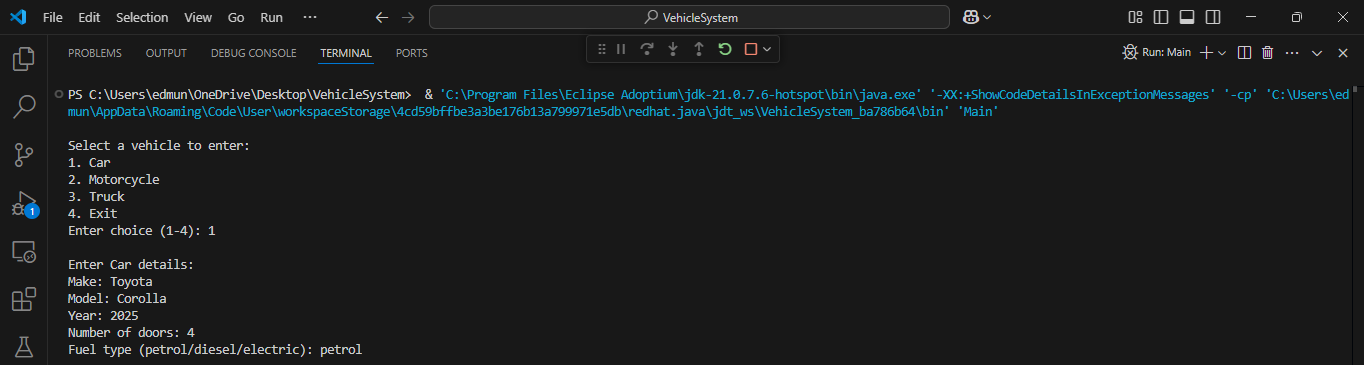
24. Includes a toString method for display

**7. Main Program Features**

* Uses Scanner to collect user input.
* Asks user to choose vehicle type (car, motorcycle, truck).
* Based on the choice, prompts for specific details.
* Displays the full details of the entered vehicle.
* Handles invalid inputs using try-catch for InputMismatchException.
* Ensures the Scanner is closed properly in a finally block.

**8. Output Example**

**User Input(Car):**



**Program Output:**



**User Input(Motorcycle):**

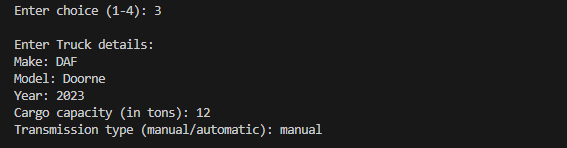
A screen shot of a computer

AI-generated content may be incorrect.

**Program Output:**



**User Input(Truck):**



**Program Output:**



**9. Code Quality**

* All variables have meaningful names (make, model, year, etc).
* Proper indentation and consistent formatting are used.
* Inline comments added for better readability.
* Code is split into clear blocks for each vehicle type.

**10. Error Handling**

* Invalid data entry (e.g., non-integer year or door number) is caught using InputMismatchException.
* Prompt messages guide users to input correct values.
* The program prevents crashing and provides meaningful error feedback.

**11. GUI Version with Swing**

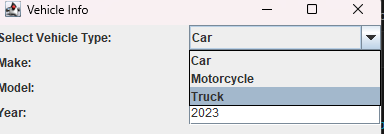
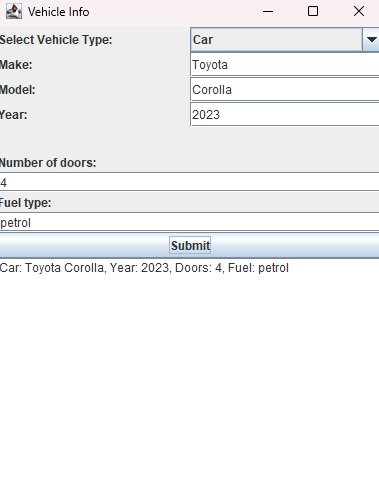
The GUI version, VehicleGUI, allows the user to select a vehicle type and enter relevant information via a form.

**Key Components:**

* JComboBox for selecting vehicle type
* JTextFields for inputting make, model, year, and type-specific fields
* Dynamic JPanel to switch input fields based on selected vehicle type
* Submit button to display the vehicle's details in a JTextArea
* Vehicle type listener to update the dynamic fields when type changes

The GUI provides a user-friendly way to interact with the system and understand Java Swing components

**User Input and Output:**



**12. Conclusion**

This Vehicle Information System provides a strong demonstration of using interfaces to ensure consistency across different classes in Java. The modular structure makes the codebase scalable and easy to maintain. Features like interactive user input, input validation, and specific implementations for each vehicle type add to the application's usability.

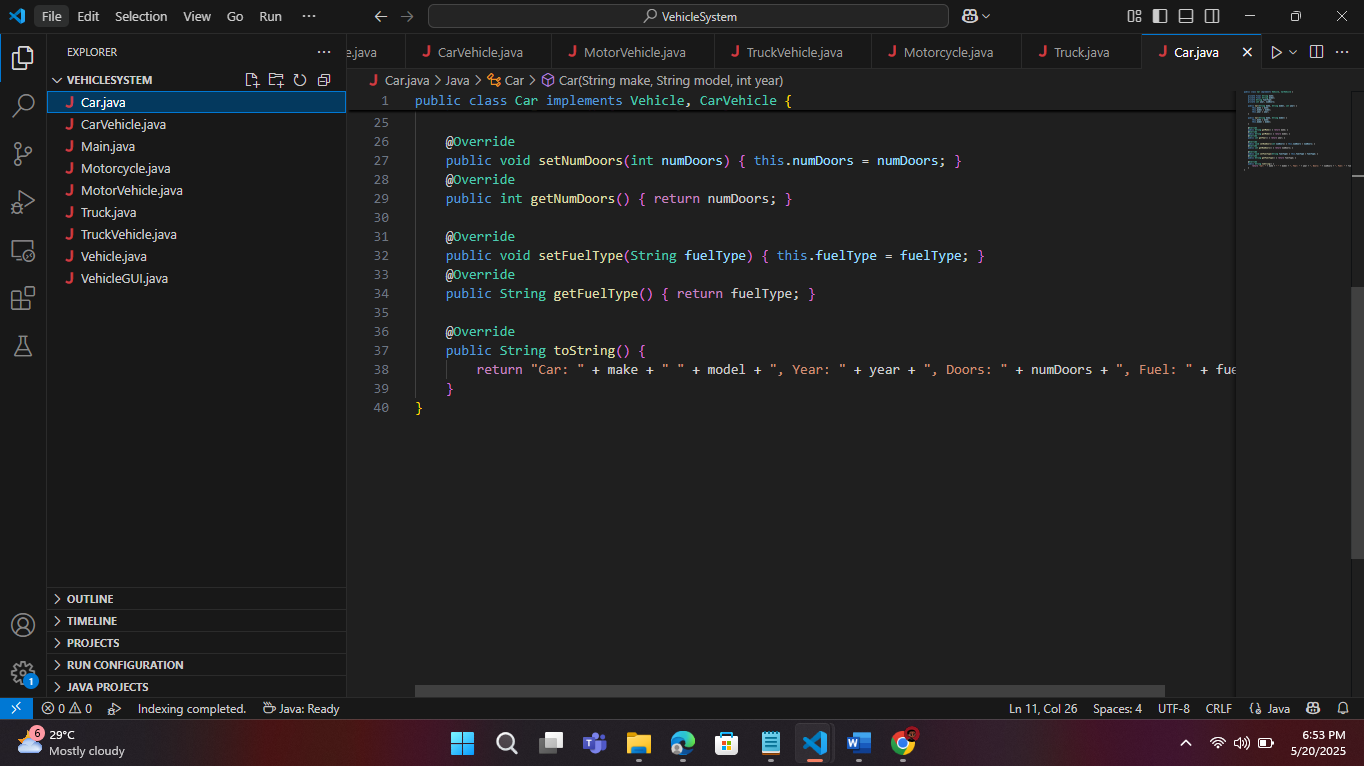
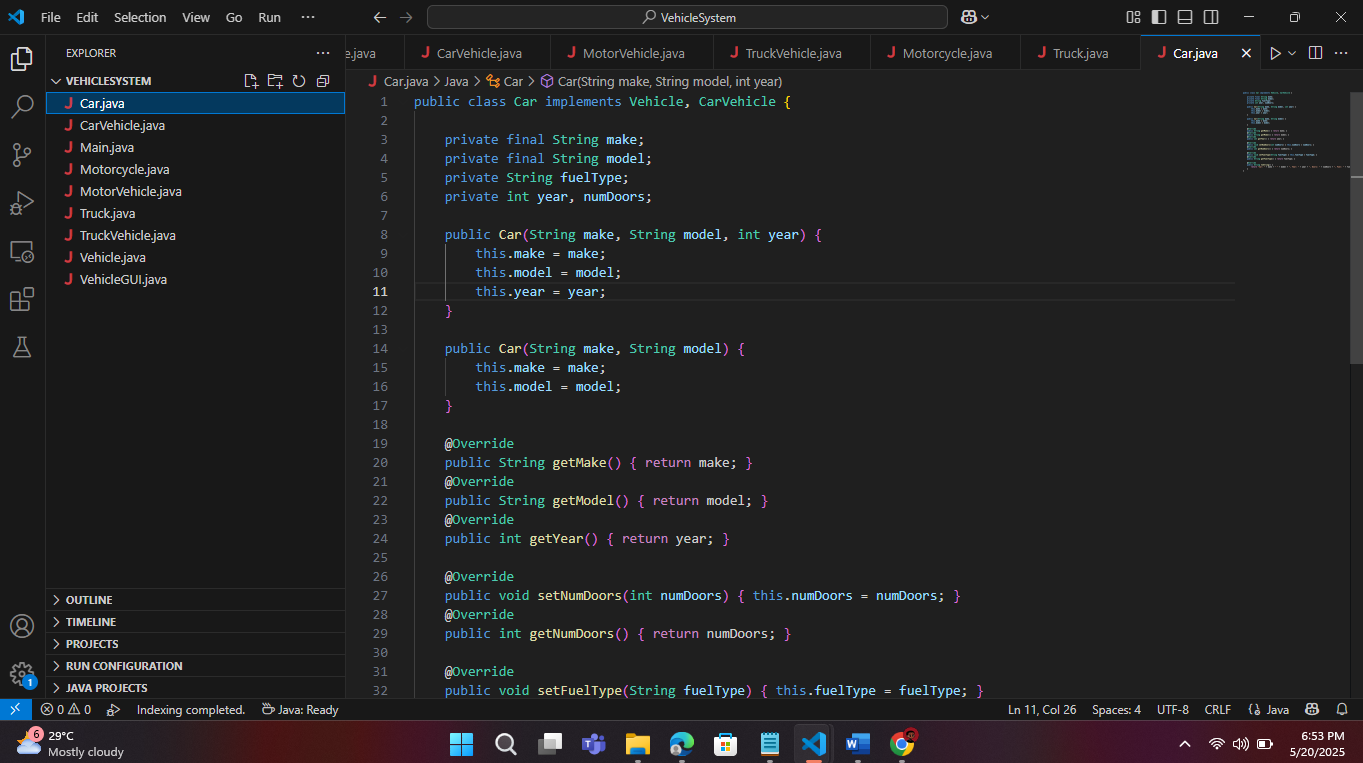
This project not only solidified the understanding of object-oriented programming principles but also introduced real-world software design thinking.

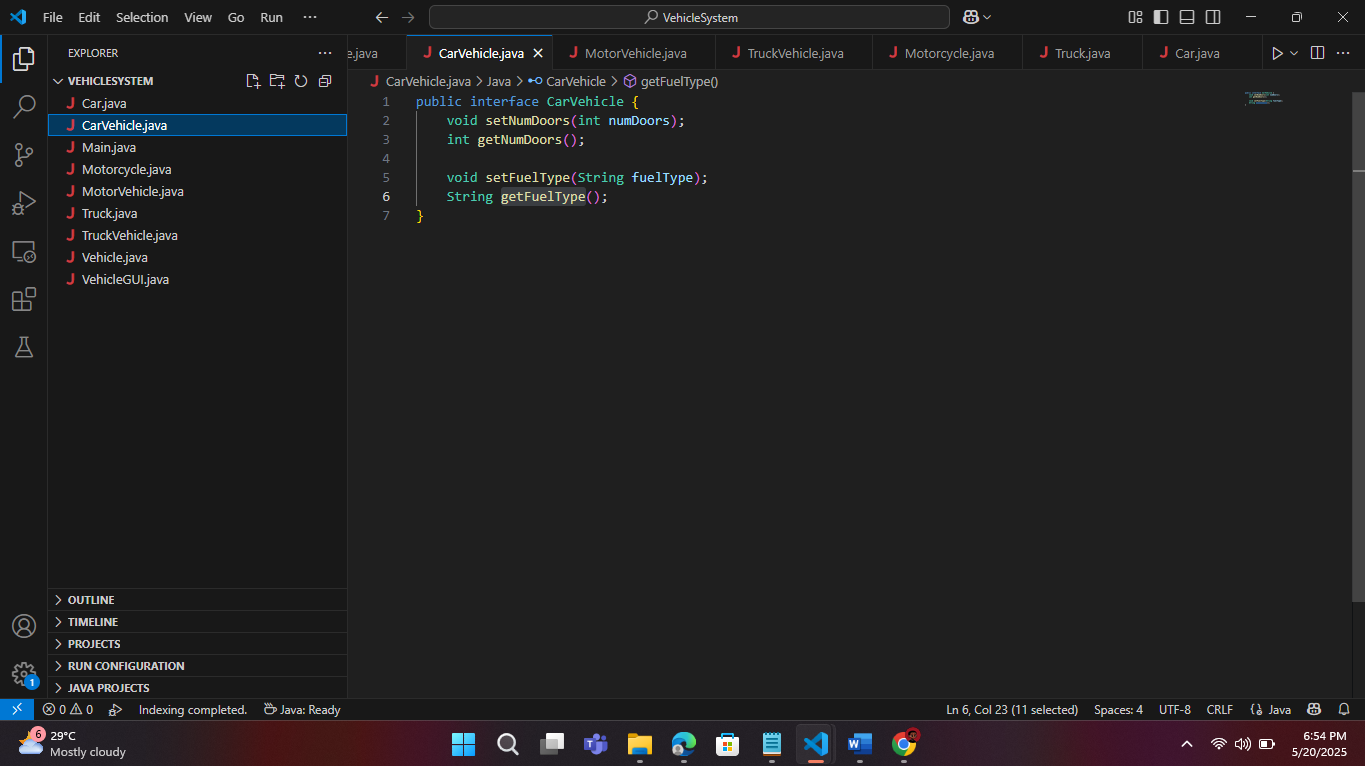
**13. References**

* Oracle. (n.d.). Java Interfaces. Retrieved from: https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html
* GeeksforGeeks. (n.d.). Java Interface. Retrieved from: https://www.geeksforgeeks.org/interfaces-in-java/
* W3Schools. (n.d.). Java Classes and Objects. Retrieved from: https://www.w3schools.com/java/java\_classes.asp

**14. Appendix**

* Source Code and Screenshot:





A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

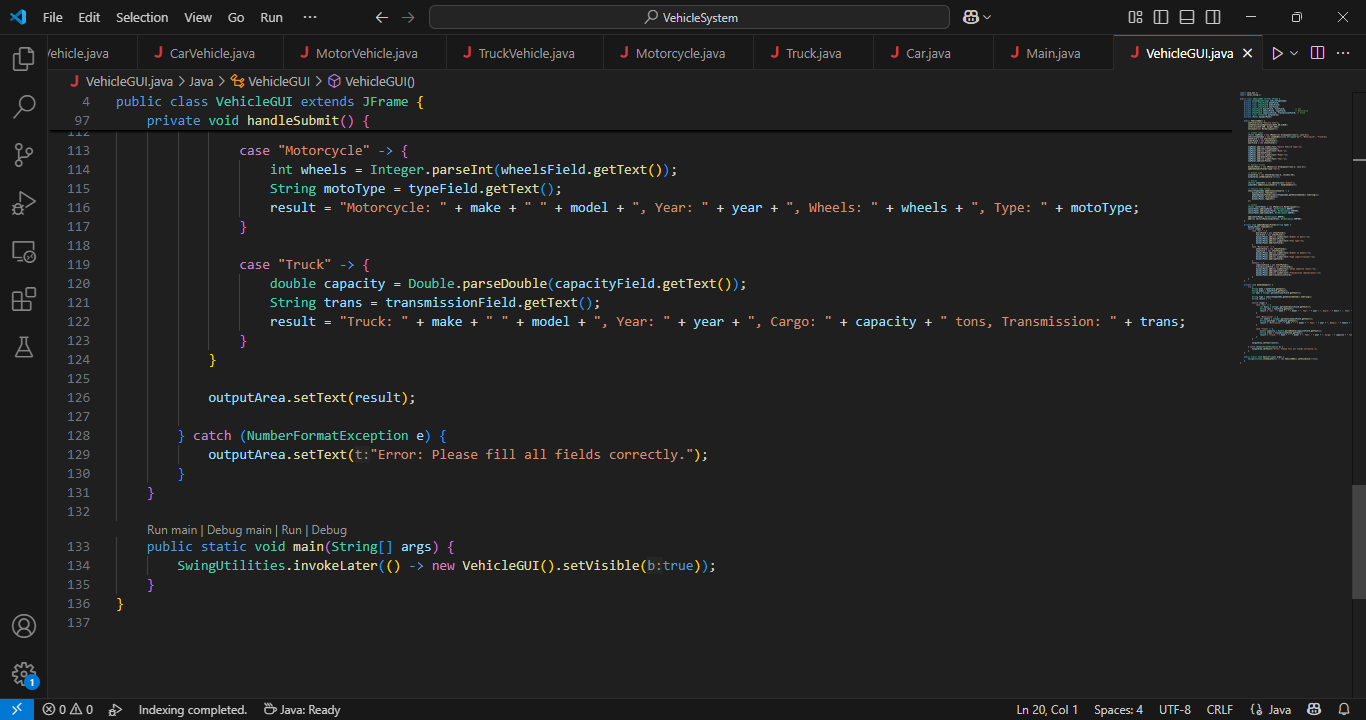
A screenshot of a computer

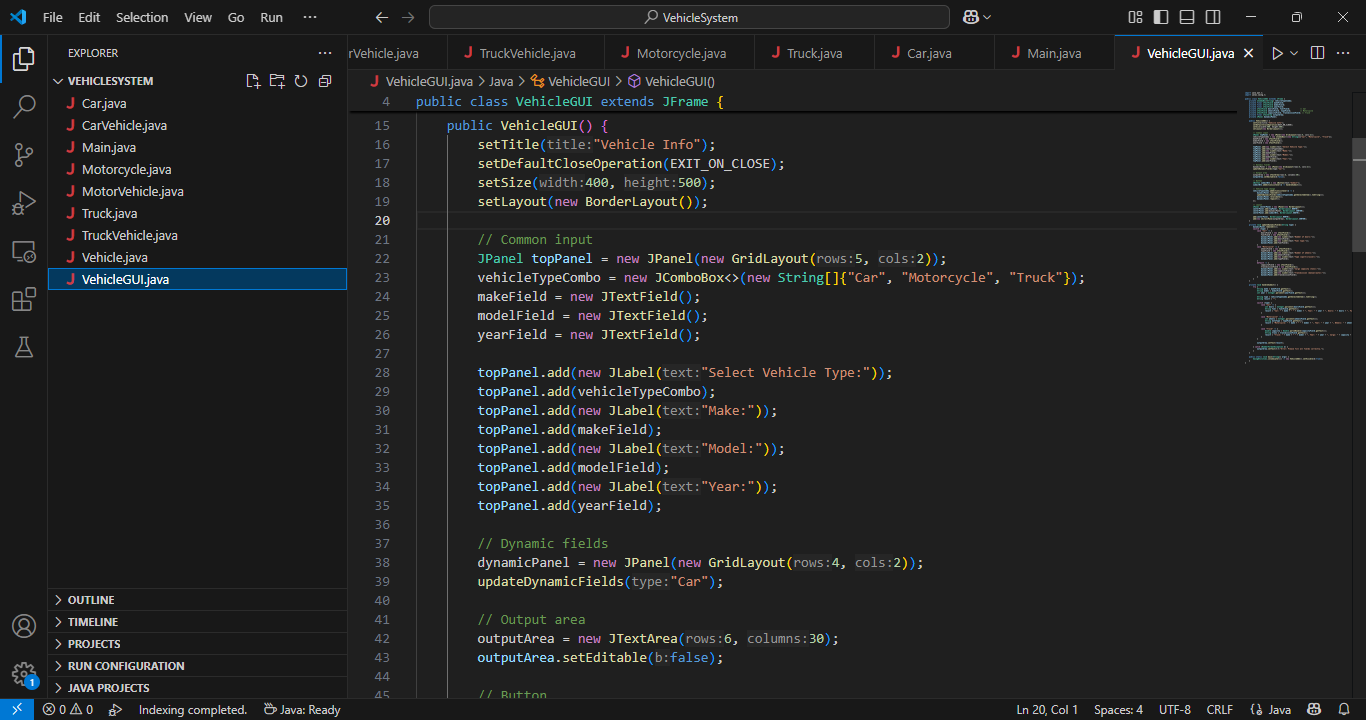
AI-generated content may be incorrect.

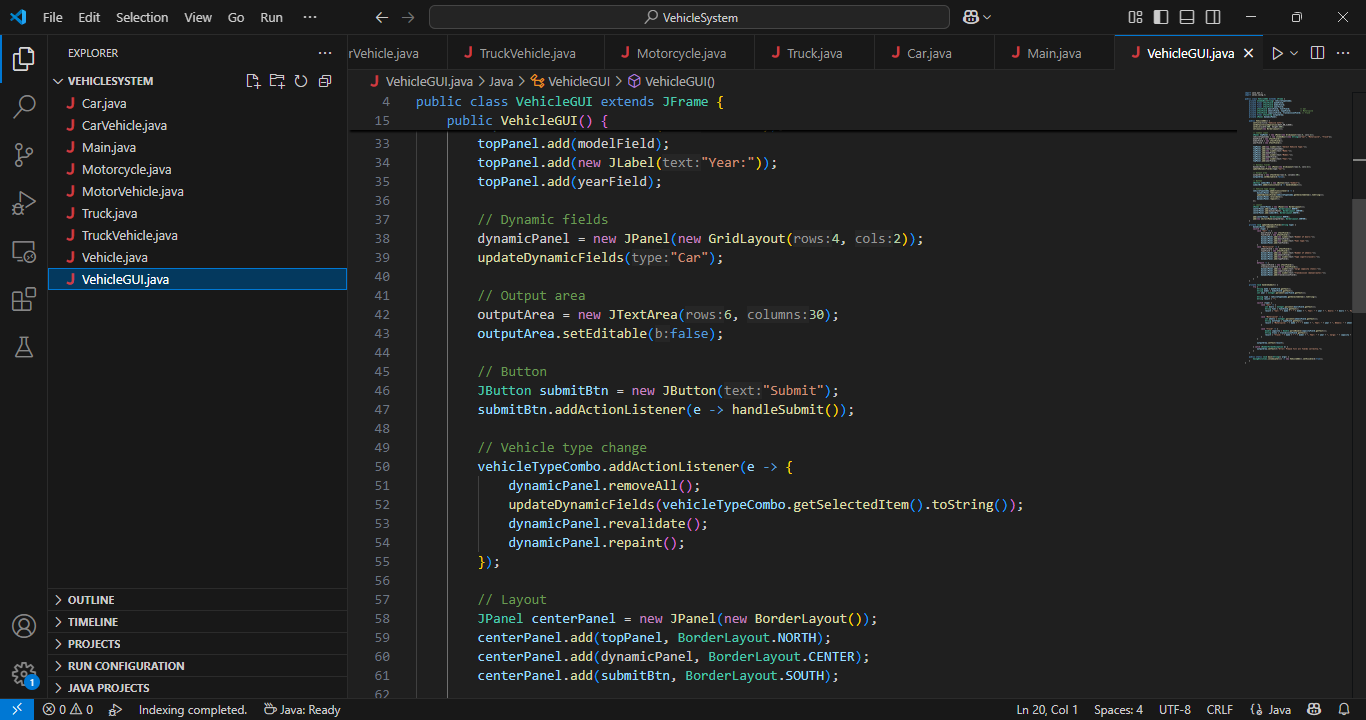
A screenshot of a computer

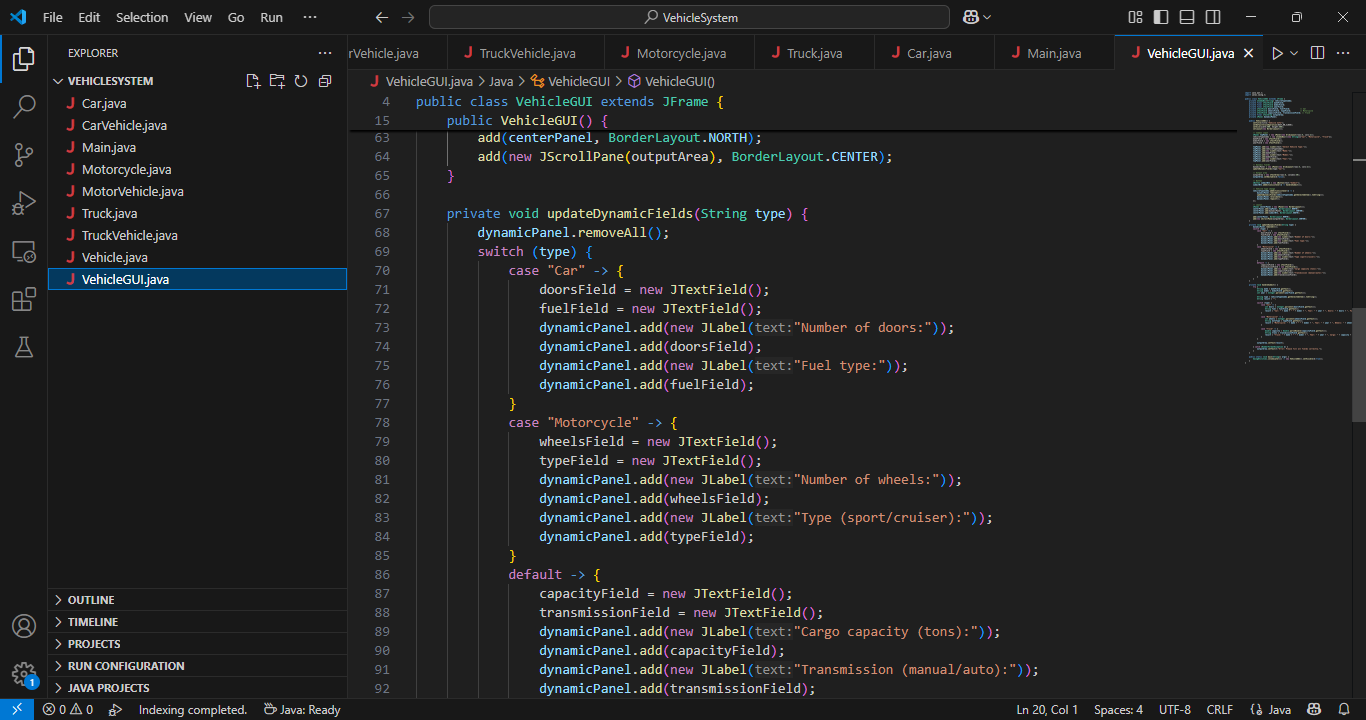
AI-generated content may be incorrect.

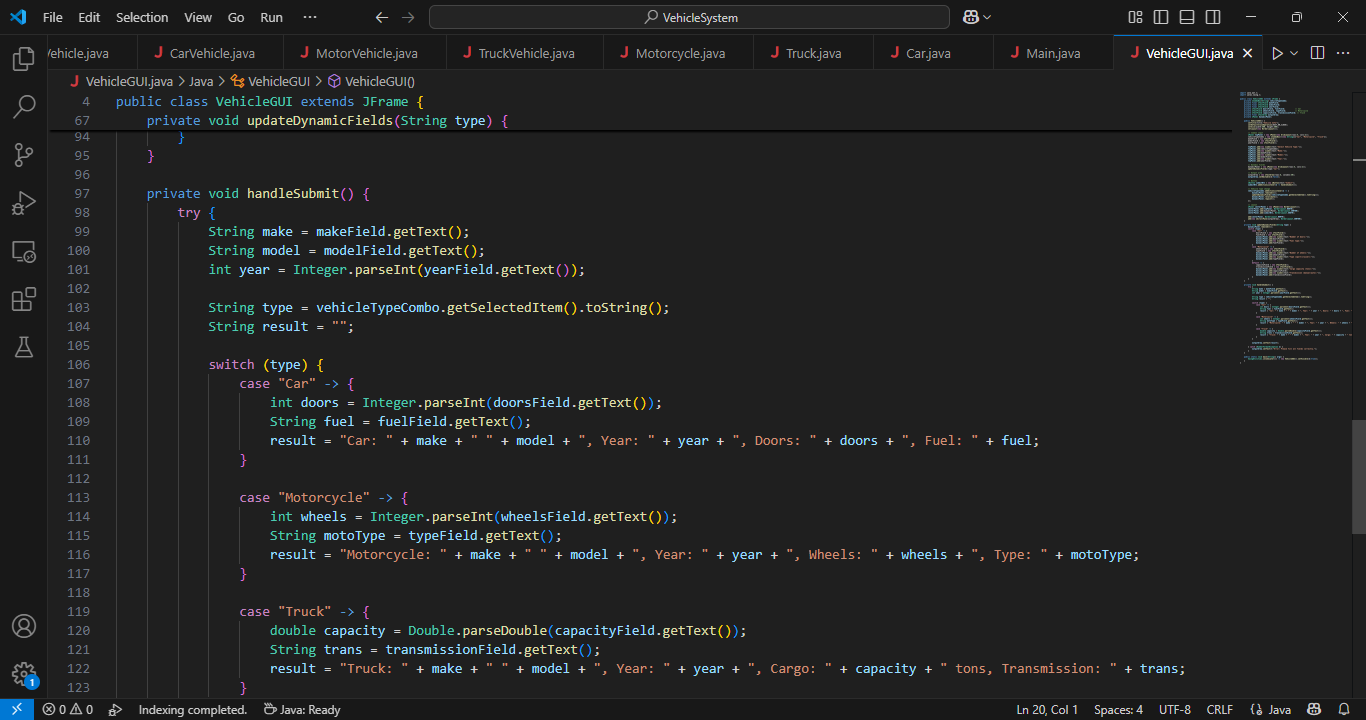
**GUI Version with Swing Screenshots and Source code**











Prepared by: Edmund Asamoah Adomako