**Vehicle Information System Documentation**

**1. Introduction**

**1.1 Overview**

The Vehicle Information System is a software application designed for a car rental agency to manage information about different types of vehicles, including cars, motorcycles, and trucks. The system enforces a common behavior among various vehicle types through the use of interfaces, ensuring consistency in structure and functionality.

**1.2 Purpose**

The purpose of the Vehicle Information System is to provide a flexible and extensible platform for handling diverse vehicle types within a car rental agency. The use of interfaces allows for a consistent structure, and the system is designed to be interactive, engaging, and user-friendly.

**1.3 Scope**

The scope of the Vehicle Information System includes the design and implementation of interfaces and classes representing different vehicle types, as well as a main program for user interaction. The system emphasizes modularity, code quality, and error handling.

**2. System Architecture**

**2.1 Overview of the System Structure**

The system comprises several interfaces and classes that represent different vehicle types. The key components include:

* **Vehicle (Interface):** Defines methods for retrieving and setting basic vehicle details such as make, model, and year.
* **CarVehicle (Interface):** Extends Vehicle and includes additional methods for managing car-specific details like the number of doors and fuel type.
* **Car (Class):** Implements CarVehicle, providing concrete implementations for retrieving and setting details of a car.
* **MotorVehicle (Interface):** Extends Vehicle and includes methods for managing motorcycle-specific details like the number of wheels and motorcycle type.
* **Motorcycle (Class):** Implements MotorVehicle, offering concrete implementations for retrieving and setting details of a motorcycle.
* **TruckVehicle (Interface):** Extends Vehicle and includes methods for managing truck-specific details such as cargo capacity and transmission type.
* **Truck (Class):** Implements TruckVehicle, providing concrete implementations for retrieving and setting details of a truck.
* **VehicleInformationSystem (Main Program):** Allows users to interactively create, input, and display details of various vehicle objects. It incorporates a menu-driven system for ease of use.

**2.2 Relationships Between Classes**

The relationships between classes are established through the implementation of interfaces. Each vehicle type class (Car, Motorcycle, Truck) implements the corresponding vehicle type interface (CarVehicle, MotorVehicle, TruckVehicle), which in turn extends the base Vehicle interface.

**3. Class and Interface Descriptions**

**3.1 Vehicle (Interface)**

**3.1.1 Purpose**

The Vehicle interface serves as a contract specifying methods for retrieving and setting basic vehicle details.

**3.1.2 Methods**

* **getMake(): String**
  + Returns the make of the vehicle.
* **getModel(): String**
  + Returns the model of the vehicle.
* **getYear(): int**
  + Returns the manufacture year of the vehicle.
* **setMake(String make): void**
  + Sets the make of the vehicle.
* **setModel(String model): void**
  + Sets the model of the vehicle.
* **setYear(int year): void**
  + Sets the year of manufacture of the vehicle.

**3.2 CarVehicle (Interface) extends Vehicle**

**3.2.1 Purpose**

The CarVehicle interface extends the base Vehicle interface and includes additional methods for managing car-specific details.

**3.2.2 Methods**

* **getNumDoors(): int**
  + Returns the number of doors of the car.
* **getFuelType(): String**
  + Returns the fuel type of the car.
* **setNumDoors(int numDoors): void**
  + Sets the number of doors of the car.
* **setFuelType(String fuelType): void**
  + Sets the fuel type of the car.

**3.3 Car (Class) implements CarVehicle**

**3.3.1 Purpose**

The Car class provides concrete implementations for the CarVehicle interface, allowing storage and retrieval of specific attributes related to cars.

**3.3.2 Methods**

* **getMake(): String**
  + Returns the make of the car.
* **getModel(): String**
  + Returns the model of the car.
* **getYear(): int**
  + Returns the year of manufacture of the car.
* **setMake(String make): void**
  + Sets the make of the car.
* **setModel(String model): void**
  + Sets the model of the car.
* **setYear(int year): void**
  + Sets the year of manufacture of the car.
* **getNumDoors(): int**
  + Returns the number of doors of the car.
* **getFuelType(): String**
  + Returns the fuel type of the car.
* **setNumDoors(int numDoors): void**
  + Sets the number of doors of the car.
* **setFuelType(String fuelType): void**
  + Sets the fuel type of the car.

**3.4 MotorVehicle (Interface) extends Vehicle**

**3.4.1 Purpose**

The MotorVehicle interface extends the base Vehicle interface and includes additional methods for managing motorcycle-specific details.

**3.4.2 Methods**

* **getNumWheels(): int**
  + Returns the number of wheels of the motorcycle.
* **getMotorcycleType(): String**
  + Returns the type of the motorcycle (sport, cruiser, off-road).
* **setNumWheels(int numWheels): void**
  + Sets the number of wheels of the motorcycle.
* **setMotorcycleType(String motorcycleType): void**
  + Sets the type of the motorcycle.

**3.5 Motorcycle (Class) implements MotorVehicle**

**3.5.1 Purpose**

The Motorcycle class provides concrete implementations for the MotorVehicle interface, allowing storage and retrieval of specific attributes related to motorcycles.

**3.5.2 Methods**

* **getMake(): String**
  + Returns the make of the motorcycle.
* **getModel(): String**
  + Returns the model of the motorcycle.
* **getYear(): int**
  + Returns the year of manufacture of the motorcycle.
* **setMake(String make): void**
  + Sets the make of the motorcycle.
* **setModel(String model): void**
  + Sets the model of the motorcycle.
* **setYear(int year): void**
  + Sets the year of manufacture of the motorcycle.
* **getNumWheels(): int**
  + Returns the number of wheels of the motorcycle.
* **getMotorcycleType(): String**
  + Returns the type of the motorcycle.
* **setNumWheels(int numWheels): void**
  + Sets the number of wheels of the motorcycle.
* **setMotorcycleType(String motorcycleType): void**
  + Sets the type of the motorcycle.

**3.6 TruckVehicle (Interface) extends Vehicle**

**3.6.1 Purpose**

The TruckVehicle interface extends the base Vehicle interface and includes additional methods for managing truck-specific details.

**3.6.2 Methods**

* **getCargoCapacity(): double**
  + Returns the cargo capacity of the truck (in tons).
* **getTransmissionType(): String**
  + Returns the transmission type of the truck (manual or automatic).
* **setCargoCapacity(double cargoCapacity): void**
  + Sets the cargo capacity of the truck.
* **setTransmissionType(String transmissionType): void**
  + Sets the transmission type of the truck.

**3.7 Truck (Class) implements TruckVehicle**

**3.7.1 Purpose**

The Truck class provides concrete implementations for the TruckVehicle interface, allowing storage and retrieval of specific attributes related to trucks.

**3.7.2 Methods**

* **getMake(): String**
  + Returns the make of the truck.
* **getModel(): String**
  + Returns the model of the truck.
* **getYear(): int**
  + Returns the year of manufacture of the truck.
* **setMake(String make): void**
  + Sets the make of the truck.
* **setModel(String model): void**
  + Sets the model of the truck.
* **setYear(int year): void**
  + Sets the year of manufacture of the truck.
* **getCargoCapacity(): double**
  + Returns the cargo capacity of the truck.
* **getTransmissionType(): String**
  + Returns the transmission type of the truck.
* **setCargoCapacity(double cargoCapacity): void**
  + Sets the cargo capacity of the truck.
* **setTransmissionType(String transmissionType): void**
  + Sets the transmission type of the truck.

**4. Main Program (VehicleInformationSystem)**

**4.1 Overview of the Main Program**

The VehicleInformationSystem class serves as the main program for user interaction. It allows users to create, input, and display details of various vehicle objects. The program incorporates a menu-driven system for ease of use.

**4.2 User Interaction and Menu System**

The main program provides a menu to users with options for entering details for a Car, Motorcycle, or Truck. Users can choose an option, and the program will execute the corresponding functionality. If the user enters an invalid option or input, the program displays an error message and prompts the user to enter a valid choice.

**4.3 Error Handling**

The program includes error handling mechanisms to manage potential exceptions, specifically **InputMismatchException**. If the user enters data of the wrong type or an invalid menu choice, the program catches the exception, displays an error message, consumes the invalid input, and continues to the menu.

**5. Code (Programming codes)**

**Main:**

// VehicleInformationSystem.java

import java.util.InputMismatchException;

import java.util.Scanner;

public class VehicleInformationSystem {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // User interaction to create and display vehicle objects

        try {

            int option;

            do {

                displayMenu();

                try {

                    option = scanner.nextInt();

                    scanner.nextLine(); // consume the newline character

                    switch (option) {

                        case 1:

                            System.out.println("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Enter details of Car:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

                            Car car = new Car();

                            inputVehicleDetails(scanner, car);

                            inputCarDetails(scanner, car);

                            displayVehicleDetails(car);

                            break;

                        case 2:

                            System.out.println("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Enter details of Motorcycle:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

                            Motorcycle motorcycle = new Motorcycle();

                            inputVehicleDetails(scanner, motorcycle);

                            inputMotorcycleDetails(scanner, motorcycle);

                            displayVehicleDetails(motorcycle);

                            break;

                        case 3:

                            System.out.println("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Enter details of Truck:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

                            Truck truck = new Truck();

                            inputVehicleDetails(scanner, truck);

                            inputTruckDetails(scanner, truck);

                            displayVehicleDetails(truck);

                            break;

                        case 0:

                            System.out.println("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Exiting the program. Goodbye!  \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

                            break;

                        default:

                            System.out.println("\n\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Invalid option. Please enter a valid option  \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

                            break;

                    }

                } catch (InputMismatchException e) {

                    System.out.println("\n\t\tInvalid input. Please enter a valid option   \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

                    scanner.nextLine(); // consume the invalid input

                    option = -1; // set option to -1 to continue the loop

                }

            } while (option != 0);

        } finally {

            scanner.close();

        }

    }

    private static void displayMenu() {

        System.out.println("\n\n\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Vehicle Information System Menu:     \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

        System.out.println("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  1. Enter details of Car              \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

        System.out.println("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  2. Enter details of Motorcycle       \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

        System.out.println("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  3. Enter details of Truck            \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

        System.out.println("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  0. Exit                              \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

        System.out.print("\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Choose an option:   ");

    }

    private static void inputVehicleDetails(Scanner scanner, Vehicle vehicle) {

        System.out.print("\t\tEnter make: ");

        vehicle.setMake(scanner.nextLine());

        System.out.print("\t\tEnter model: ");

        vehicle.setModel(scanner.nextLine());

        System.out.print("\t\tEnter year: ");

        vehicle.setYear(scanner.nextInt());

        scanner.nextLine(); // consume the newline character

    }

    private static void inputCarDetails(Scanner scanner, CarVehicle car) {

        System.out.print("\t\tEnter number of doors: ");

        car.setNumDoors(scanner.nextInt());

        scanner.nextLine(); // consume the newline character

        System.out.print("\t\tEnter fuel type: ");

        car.setFuelType(scanner.nextLine());

    }

    private static void inputMotorcycleDetails(Scanner scanner, MotorVehicle motorcycle) {

        System.out.print("\t\tEnter number of wheels: ");

        motorcycle.setNumWheels(scanner.nextInt());

        scanner.nextLine(); // consume the newline character

        System.out.print("\t\tEnter motorcycle type: ");

        motorcycle.setMotorcycleType(scanner.nextLine());

    }

    private static void inputTruckDetails(Scanner scanner, TruckVehicle truck) {

        System.out.print("\t\tEnter cargo capacity (in tons): ");

        truck.setCargoCapacity(scanner.nextDouble());

        scanner.nextLine(); // consume the newline character

        System.out.print("\t\tEnter transmission type: ");

        truck.setTransmissionType(scanner.nextLine());

    }

    private static void displayVehicleDetails(Vehicle vehicle) {

        System.out.println("\n\n\t\t\t\t$$$$$$$$$$\*\*\*\*\*\*\*\*\*\*\*\*\*\* Vehicle Details:   \*\*\*\*\*\*\*\*\*\*\*\*\*\*$$$$$$$$$$");

        System.out.println("\t\t\t\t Make: " + vehicle.getMake());

        System.out.println("\t\t\t\t Model: " + vehicle.getModel());

        System.out.println("\t\t\t\t Year: " + vehicle.getYear());

    }

}

// Vehicle.java

public interface Vehicle {

    String getMake();

    String getModel();

    int getYear();

    void setMake(String make);

    void setModel(String model);

    void setYear(int year);

}

// CarVehicle.java

public interface CarVehicle extends Vehicle {

    int getNumDoors();

    String getFuelType();

    void setNumDoors(int numDoors);

    void setFuelType(String fuelType);

}

// Car.java

public class Car implements CarVehicle {

    private String make;

    private String model;

    private int year;

    private int numDoors;

    private String fuelType;

    @Override

    public String getMake() {

        return make;

    }

    @Override

    public String getModel() {

        return model;

    }

    @Override

    public int getYear() {

        return year;

    }

    @Override

    public void setMake(String make) {

        this.make = make;

    }

    @Override

    public void setModel(String model) {

        this.model = model;

    }

    @Override

    public void setYear(int year) {

        this.year = year;

    }

    @Override

    public int getNumDoors() {

        return numDoors;

    }

    @Override

    public String getFuelType() {

        return fuelType;

    }

    @Override

    public void setNumDoors(int numDoors) {

        this.numDoors = numDoors;

    }

    @Override

    public void setFuelType(String fuelType) {

        this.fuelType = fuelType;

    }

}

// MotorVehicle.java

public interface MotorVehicle extends Vehicle {

    int getNumWheels();

    String getMotorcycleType();

    void setNumWheels(int numWheels);

    void setMotorcycleType(String motorcycleType);

}

// Motorcycle.java

public class Motorcycle implements MotorVehicle {

    private String make;

    private String model;

    private int year;

    private int numWheels;

    private String motorcycleType;

    @Override

    public String getMake() {

        return make;

    }

    @Override

    public String getModel() {

        return model;

    }

    @Override

    public int getYear() {

        return year;

    }

    @Override

    public void setMake(String make) {

        this.make = make;

    }

    @Override

    public void setModel(String model) {

        this.model = model;

    }

    @Override

    public void setYear(int year) {

        this.year = year;

    }

    @Override

    public int getNumWheels() {

        return numWheels;

    }

    @Override

    public String getMotorcycleType() {

        return motorcycleType;

    }

    @Override

    public void setNumWheels(int numWheels) {

        this.numWheels = numWheels;

    }

    @Override

    public void setMotorcycleType(String motorcycleType) {

        this.motorcycleType = motorcycleType;

    }

}

// TruckVehicle.java

public interface TruckVehicle extends Vehicle {

    double getCargoCapacity();

    String getTransmissionType();

    void setCargoCapacity(double cargoCapacity);

    void setTransmissionType(String transmissionType);

}

// Truck.java

public class Truck implements TruckVehicle {

    private String make;

    private String model;

    private int year;

    private double cargoCapacity;

    private String transmissionType;

    @Override

    public String getMake() {

        return make;

    }

    @Override

    public String getModel() {

        return model;

    }

    @Override

    public int getYear() {

        return year;

    }

    @Override

    public void setMake(String make) {

        this.make = make;

    }

    @Override

    public void setModel(String model) {

        this.model = model;

    }

    @Override

    public void setYear(int year) {

        this.year = year;

    }

    @Override

    public double getCargoCapacity() {

        return cargoCapacity;

    }

    @Override

    public String getTransmissionType() {

        return transmissionType;

    }

    @Override

    public void setCargoCapacity(double cargoCapacity) {

        this.cargoCapacity = cargoCapacity;

    }

    @Override

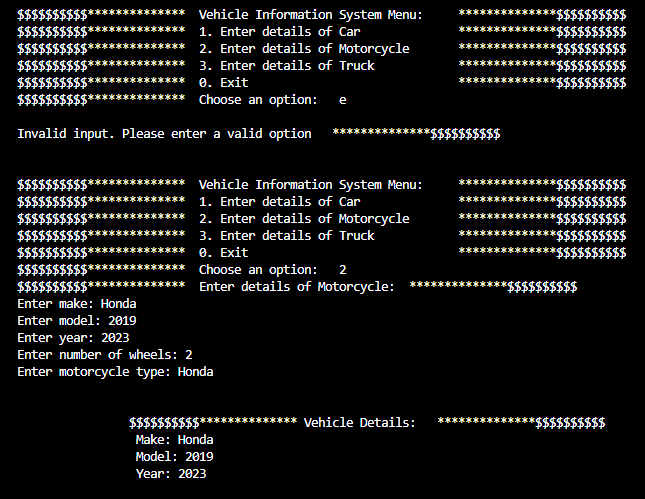
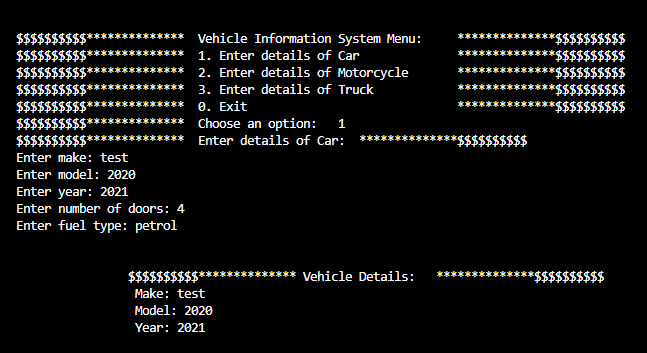
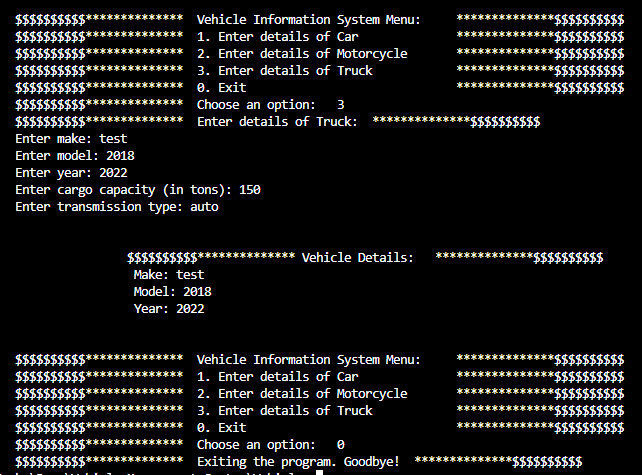
    public void setTransmissionType(String transmissionType) {

        this.transmissionType = transmissionType;

    }

}

**6. Output**

**  
  
  
  
**

**The EnD**