**Programming Assignment Unit 6**

Java Code:

``

import java.util.Scanner;

// Interface for basic vehicle information

interface Vehicle {

String getMake();

String getModel();

int getYear();

}

// Interface for additional car-specific information

interface CarVehicle {

int getNumberOfDoors();

void setNumberOfDoors(int numberOfDoors);

String getFuelType();

void setFuelType(String fuelType);

}

// Interface for additional motorcycle-specific information

interface MotorVehicle {

int getNumberOfWheels();

void setNumberOfWheels(int numberOfWheels);

String getMotorcycleType();

void setMotorcycleType(String motorcycleType);

}

// Interface for additional truck-specific information

interface TruckVehicle {

double getCargoCapacity();

void setCargoCapacity(double cargoCapacity);

String getTransmissionType();

void setTransmissionType(String transmissionType);

}

// Car class implementing Vehicle and CarVehicle interfaces

class Car implements Vehicle, CarVehicle {

private String make;

private String model;

private int year;

private int numberOfDoors;

private String fuelType;

public Car(String make, String model, int year) {

this.make = make;

this.model = model;

this.year = year;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public int getYear() {

return year;

}

public int getNumberOfDoors() {

return numberOfDoors;

}

public void setNumberOfDoors(int numberOfDoors) {

this.numberOfDoors = numberOfDoors;

}

public String getFuelType() {

return fuelType;

}

public void setFuelType(String fuelType) {

this.fuelType = fuelType;

}

public String toString() {

return "Car [Make=" + make + ", Model=" + model + ", Year=" + year + ", Doors=" + numberOfDoors + ", Fuel=" + fuelType + "]";

}

}

// Motorcycle class implementing Vehicle and MotorVehicle interfaces

class Motorcycle implements Vehicle, MotorVehicle {

private String make;

private String model;

private int year;

private int numberOfWheels;

private String motorcycleType;

public Motorcycle(String make, String model, int year) {

this.make = make;

this.model = model;

this.year = year;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public int getYear() {

return year;

}

public int getNumberOfWheels() {

return numberOfWheels;

}

public void setNumberOfWheels(int numberOfWheels) {

this.numberOfWheels = numberOfWheels;

}

public String getMotorcycleType() {

return motorcycleType;

}

public void setMotorcycleType(String motorcycleType) {

this.motorcycleType = motorcycleType;

}

public String toString() {

return "Motorcycle [Make=" + make + ", Model=" + model + ", Year=" + year + ", Wheels=" + numberOfWheels + ", Type=" + motorcycleType + "]";

}

}

// Truck class implementing Vehicle and TruckVehicle interfaces

class Truck implements Vehicle, TruckVehicle {

private String make;

private String model;

private int year;

private double cargoCapacity;

private String transmissionType;

public Truck(String make, String model, int year) {

this.make = make;

this.model = model;

this.year = year;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public int getYear() {

return year;

}

public double getCargoCapacity() {

return cargoCapacity;

}

public void setCargoCapacity(double cargoCapacity) {

this.cargoCapacity = cargoCapacity;

}

public String getTransmissionType() {

return transmissionType;

}

public void setTransmissionType(String transmissionType) {

this.transmissionType = transmissionType;

}

public String toString() {

return "Truck [Make=" + make + ", Model=" + model + ", Year=" + year + ", Capacity=" + cargoCapacity + " tons, Transmission=" + transmissionType + "]";

}

}

// Main program to demonstrate the functionality

public class VehicleRentalSystem {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Choose a vehicle type to create: ");

System.out.println("1. Car");

System.out.println("2. Motorcycle");

System.out.println("3. Truck");

int choice = scanner.nextInt();

scanner.nextLine(); // consume newline

switch (choice) {

case 1:

Car car = new Car("Toyota", "Corolla", 2020);

car.setNumberOfDoors(4);

car.setFuelType("Petrol");

System.out.println(car);

break;

case 2:

Motorcycle motorcycle = new Motorcycle("Yamaha", "MT-09", 2021);

motorcycle.setNumberOfWheels(2);

motorcycle.setMotorcycleType("Sport");

System.out.println(motorcycle);

break;

case 3:

Truck truck = new Truck("Ford", "F-150", 2019);

truck.setCargoCapacity(5.0);

truck.setTransmissionType("Automatic");

System.out.println(truck);

break;

default:

System.out.println("Invalid choice.");

break;

}

scanner.close();

}

}

``

**Screenshot:**

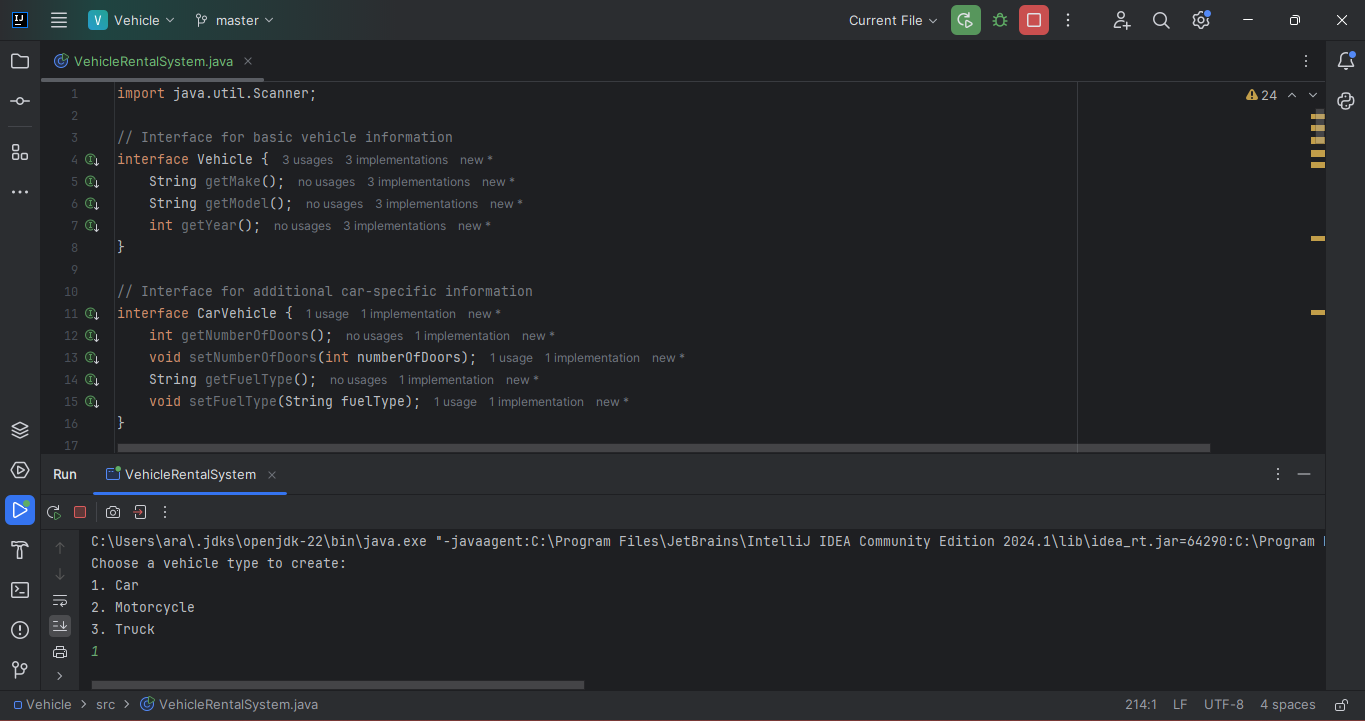
****

Figure 1

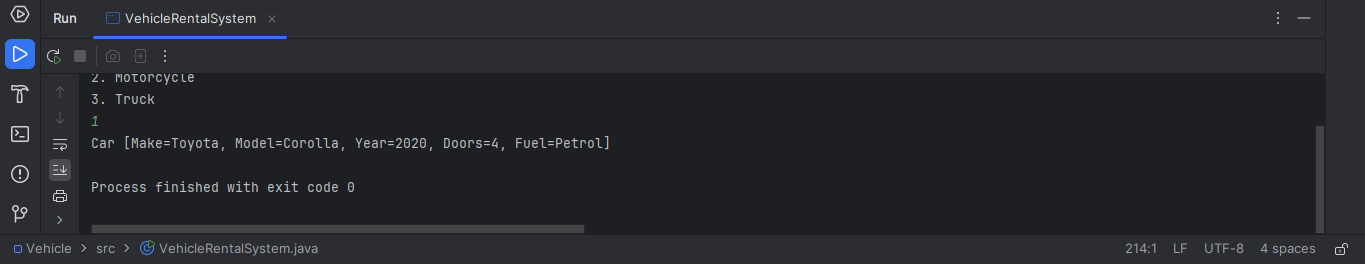
****

Figure 2: Code Output

**Explanation:**

1. Vehicle Interface: Defines methods to get the make, model, and year of a vehicle.

2. CarVehicle Interface: Extends `Vehicle` with methods specific to cars, such as getting and setting the number of doors and fuel type.

3. MotorVehicle Interface: Extends `Vehicle` with methods specific to motorcycles, such as getting and setting the number of wheels and type of motorcycle.

4. TruckVehicle Interface: Extends `Vehicle` with methods specific to trucks, such as getting and setting the cargo capacity and transmission type.

5. Car Class: Implements `Vehicle` and `CarVehicle`, providing concrete implementations of the required methods.

6. Motorcycle Class: Implements `Vehicle` and `MotorVehicle`, providing concrete implementations of the required methods.

7. Truck Class: Implements `Vehicle` and `TruckVehicle`, providing concrete implementations of the required methods.

8. Main Program: Allows the user to create instances of `Car`, `Motorcycle`, or `Truck` and display their details.

**References:**

1. Horstmann, C. S., & Cornell, G. (2013). Core Java Volume I--Fundamentals (9th ed.). Prentice Hall.

2. Sierra, K., & Bates, B. (2014). Head First Java (2nd ed.). O'Reilly Media.

3. Eckel, B. (2006). Thinking in Java (4th ed.). Prentice Hall.