**Lewis Ombaka Programming Assignment Unit 6**

1. **Define the** Vehicle **Interface**

java

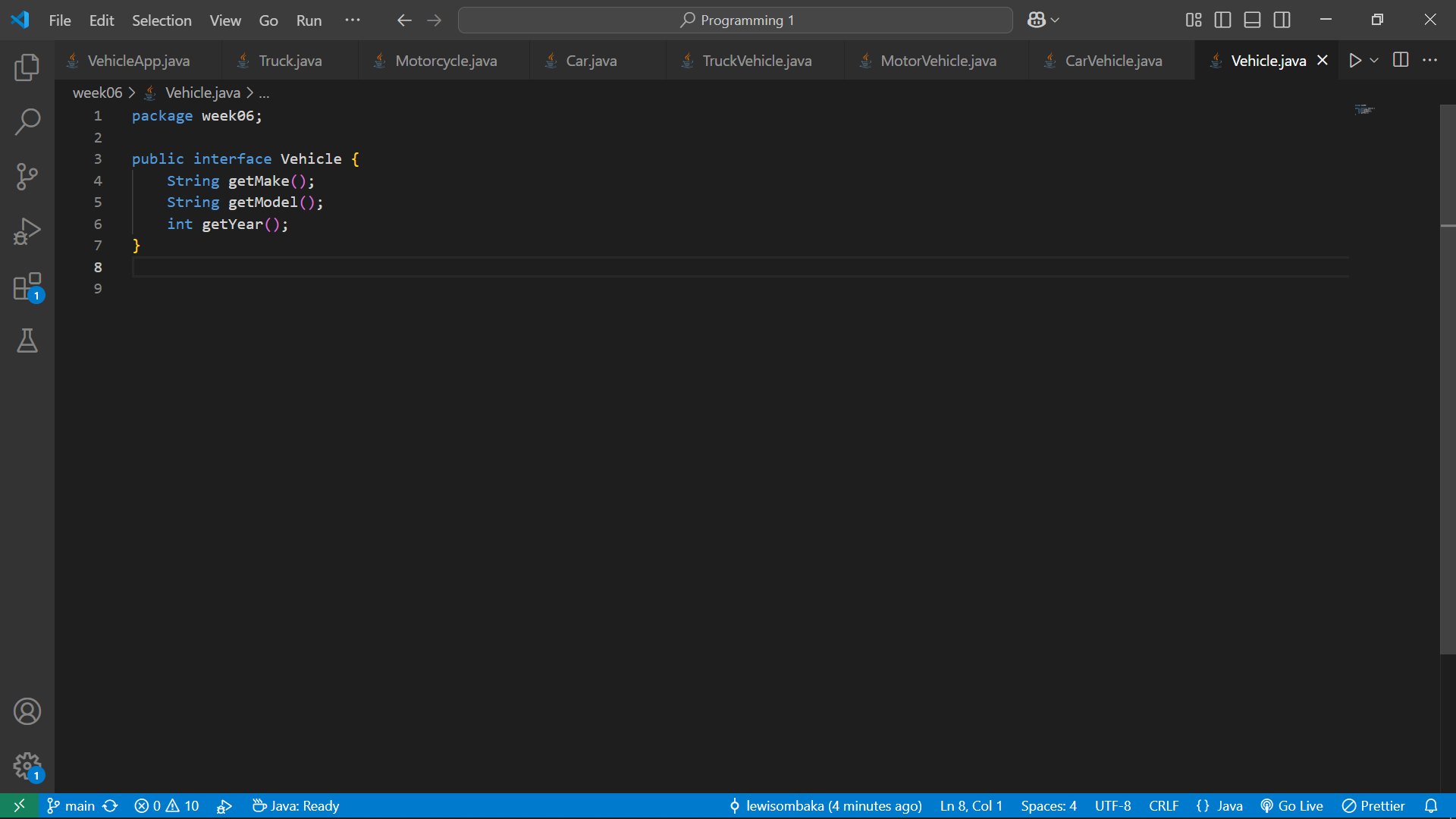
public interface Vehicle {

String getMake();

String getModel();

int getYear();

}



1. **Define the** CarVehicle **Interface**

java

public interface CarVehicle extends Vehicle {

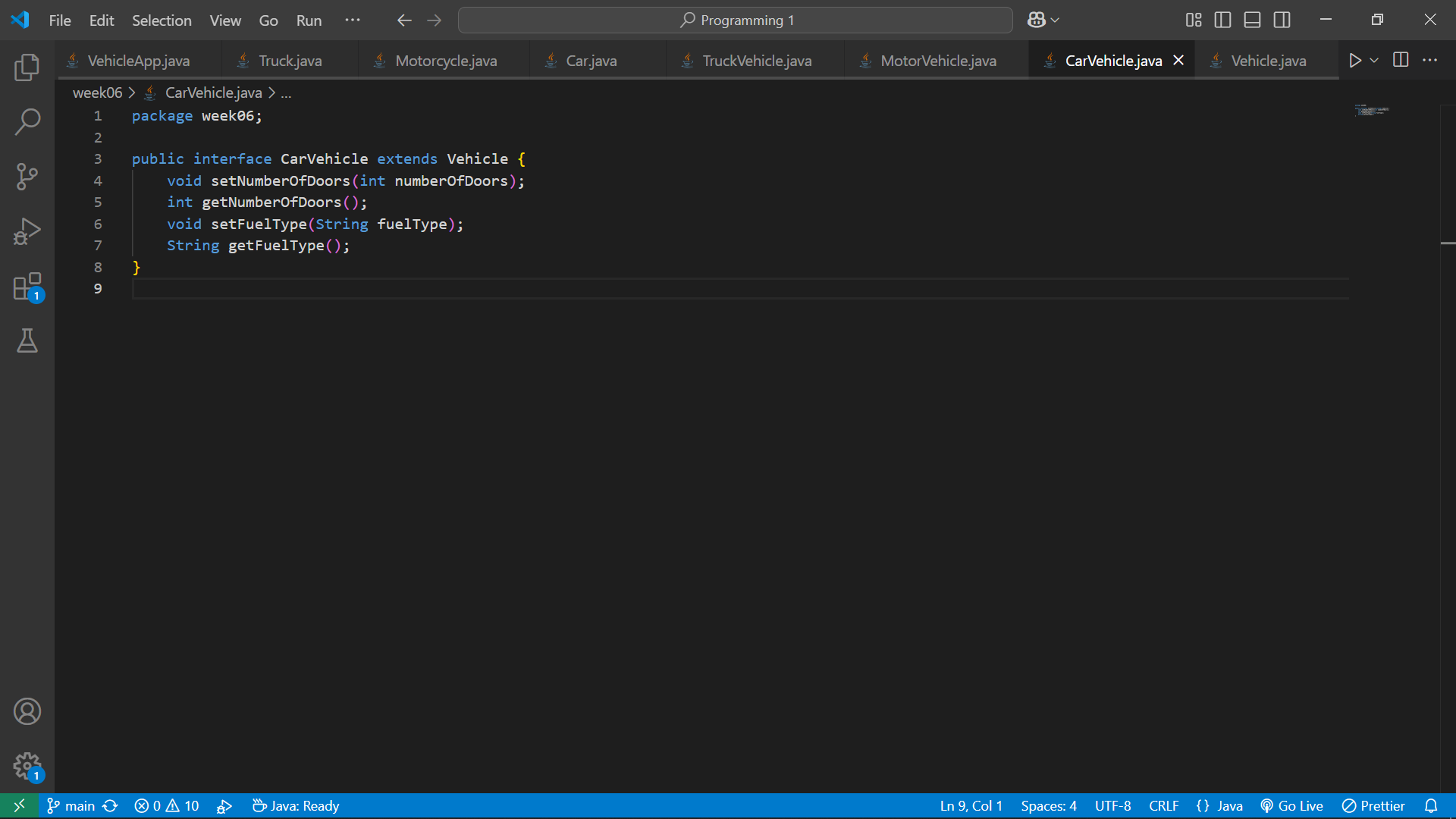
void setNumberOfDoors(int numberOfDoors);

int getNumberOfDoors();

void setFuelType(String fuelType);

String getFuelType();

}



1. **Define the** MotorVehicle **Interface**

java

public interface MotorVehicle extends Vehicle {

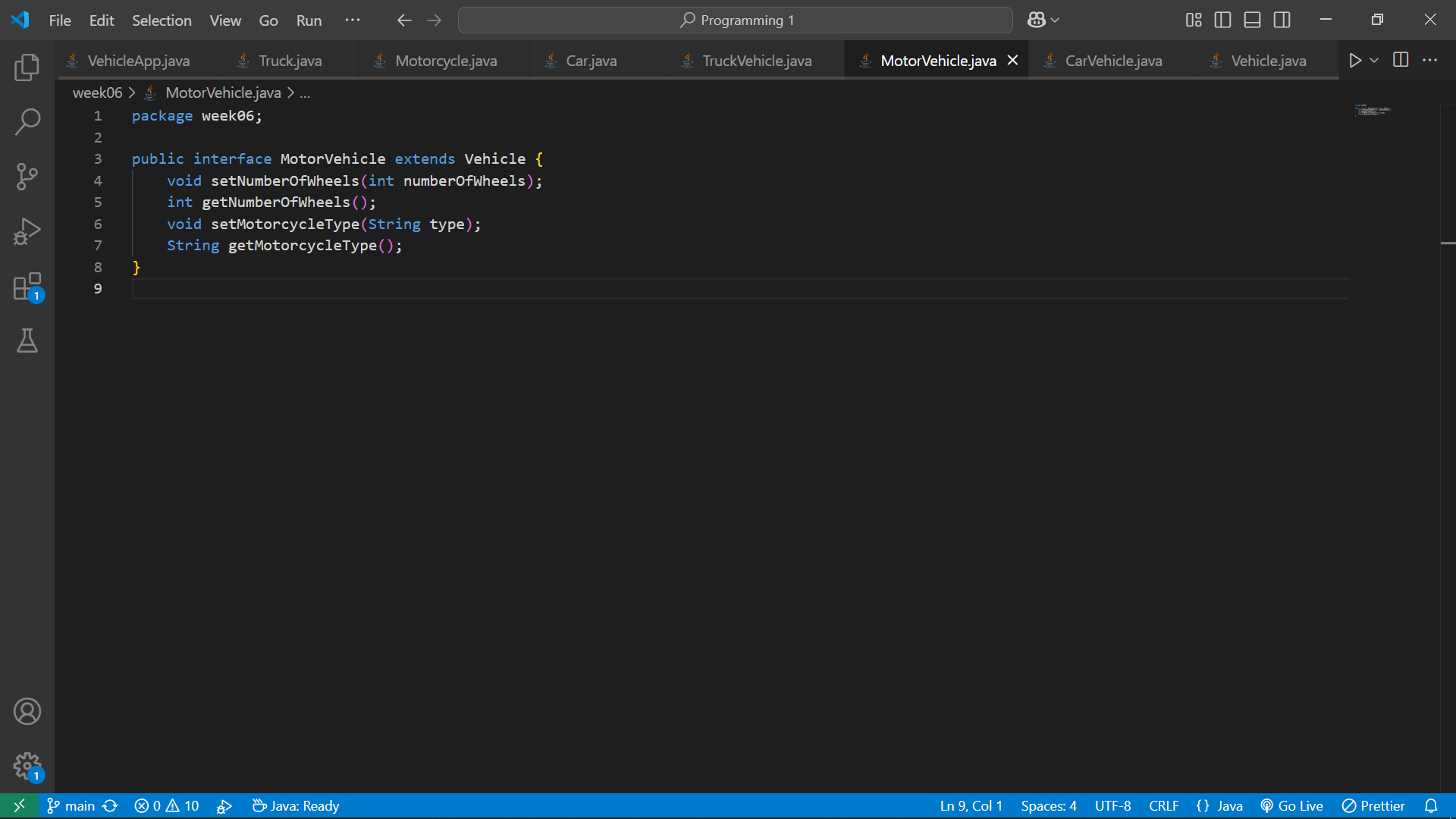
void setNumberOfWheels(int numberOfWheels);

int getNumberOfWheels();

void setMotorcycleType(String type);

String getMotorcycleType();

}



1. **Define the** TruckVehicle **Interface**

java

public interface TruckVehicle extends Vehicle {

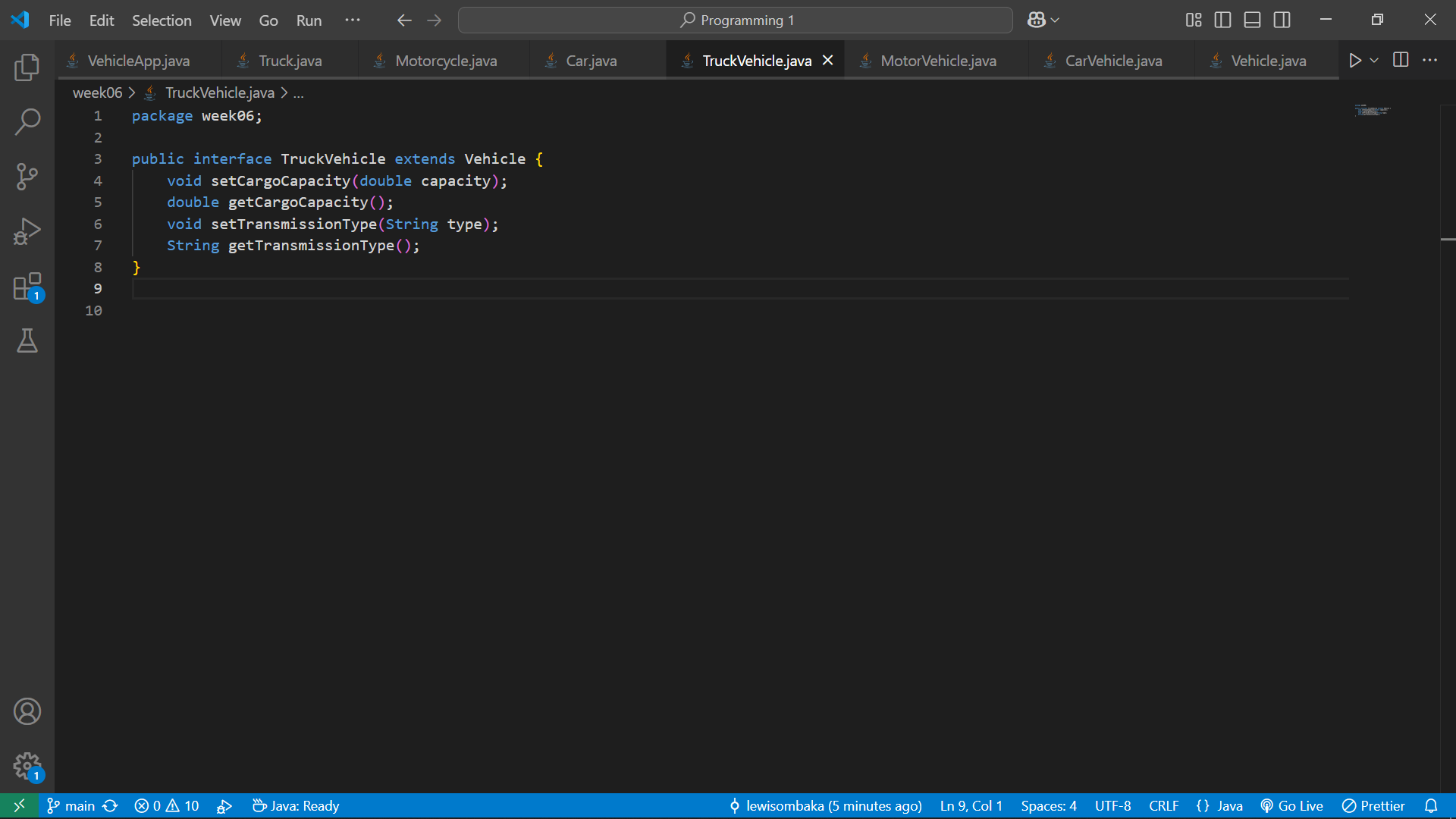
void setCargoCapacity(double capacity);

double getCargoCapacity();

void setTransmissionType(String type);

String getTransmissionType();

}



1. **Implement the** Car **Class**

java

public class Car implements 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 void setNumberOfDoors(int numberOfDoors) {

this.numberOfDoors = numberOfDoors;

}

public int getNumberOfDoors() {

return numberOfDoors;

}

public void setFuelType(String fuelType) {

this.fuelType = fuelType;

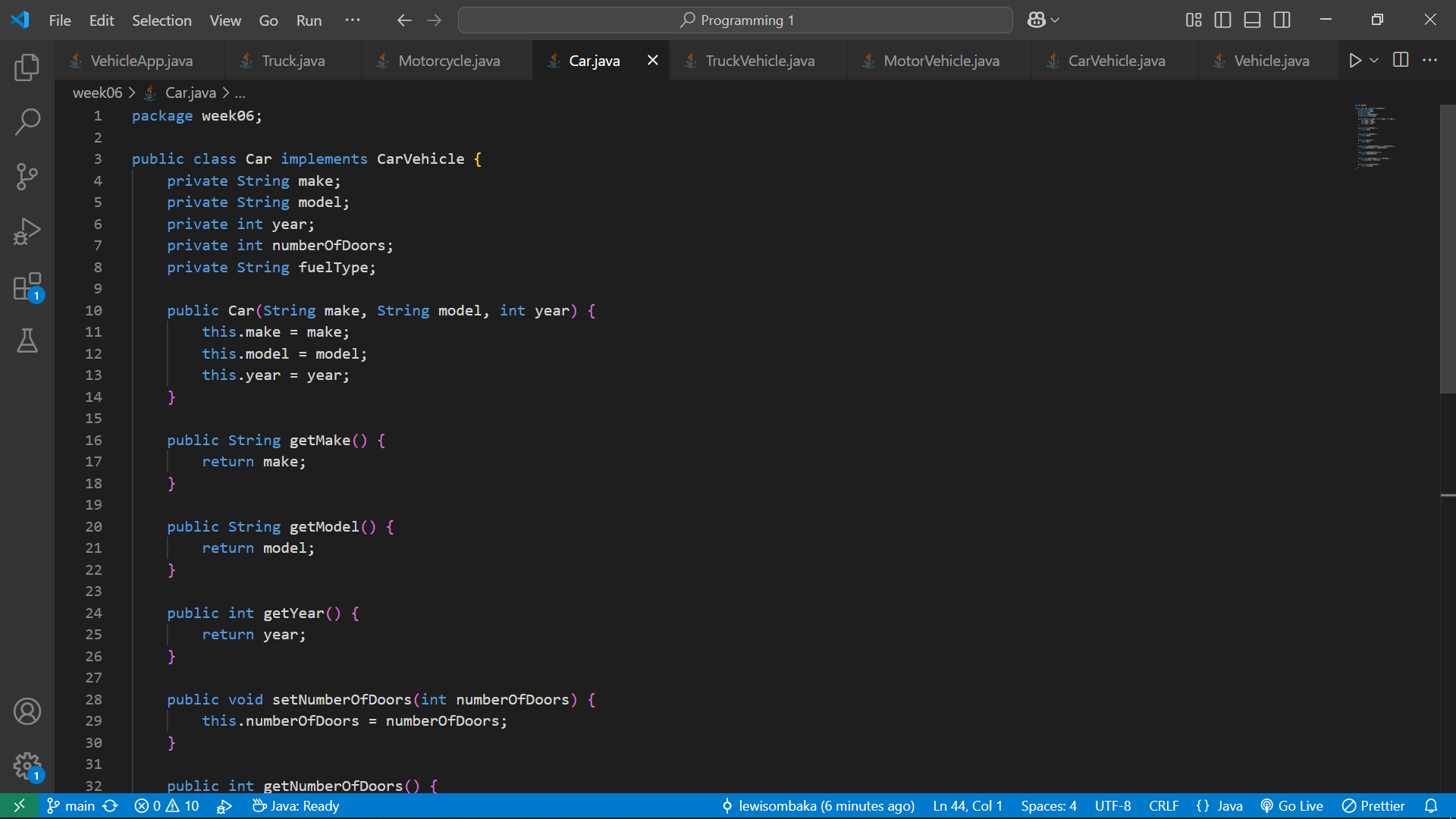
}

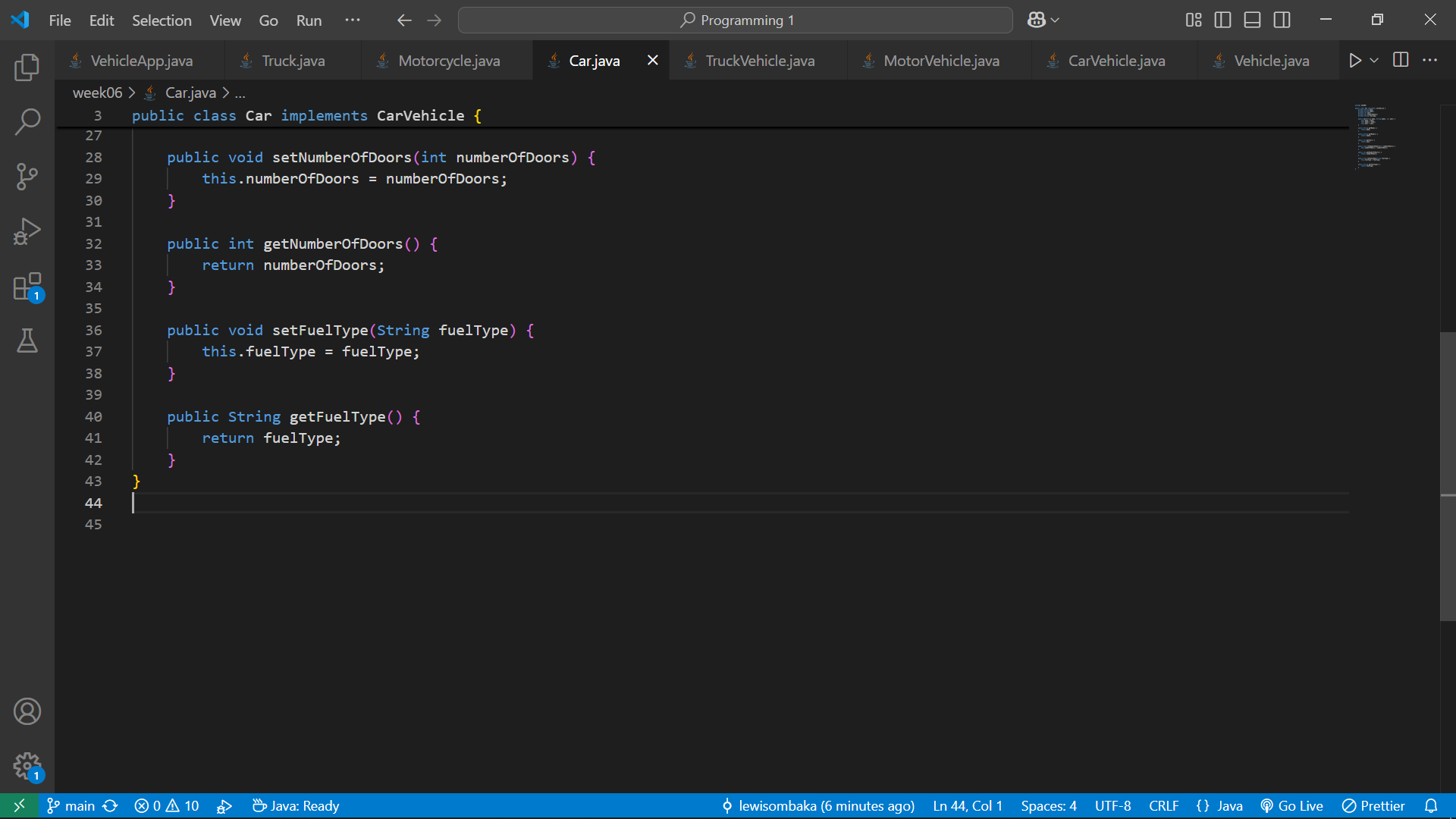
public String getFuelType() {

return fuelType;

}

}





1. **Implement the** Motorcycle **Class**

java

public class Motorcycle implements 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 void setNumberOfWheels(int numberOfWheels) {

this.numberOfWheels = numberOfWheels;

}

public int getNumberOfWheels() {

return numberOfWheels;

}

public void setMotorcycleType(String motorcycleType) {

this.motorcycleType = motorcycleType;

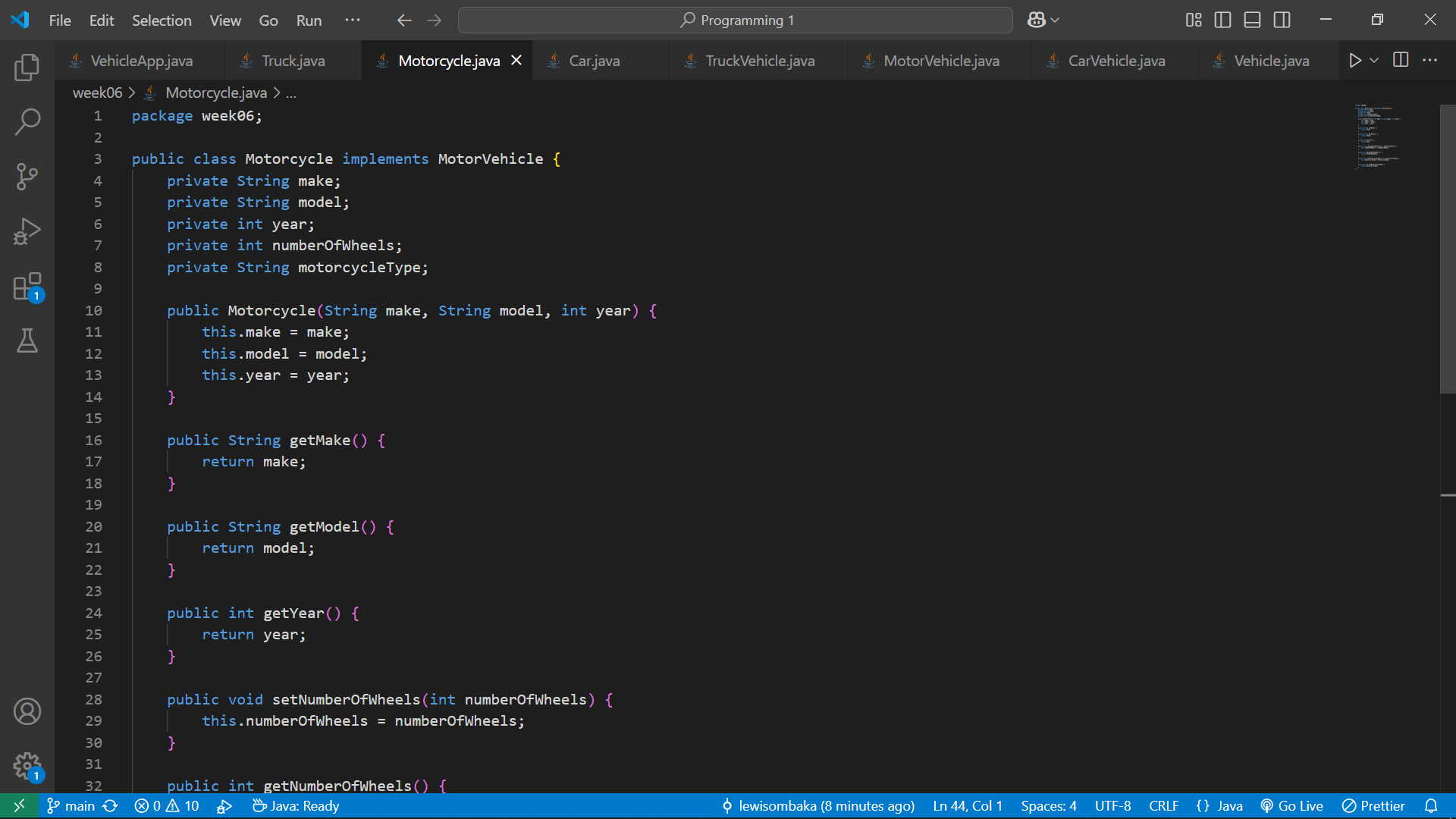
}

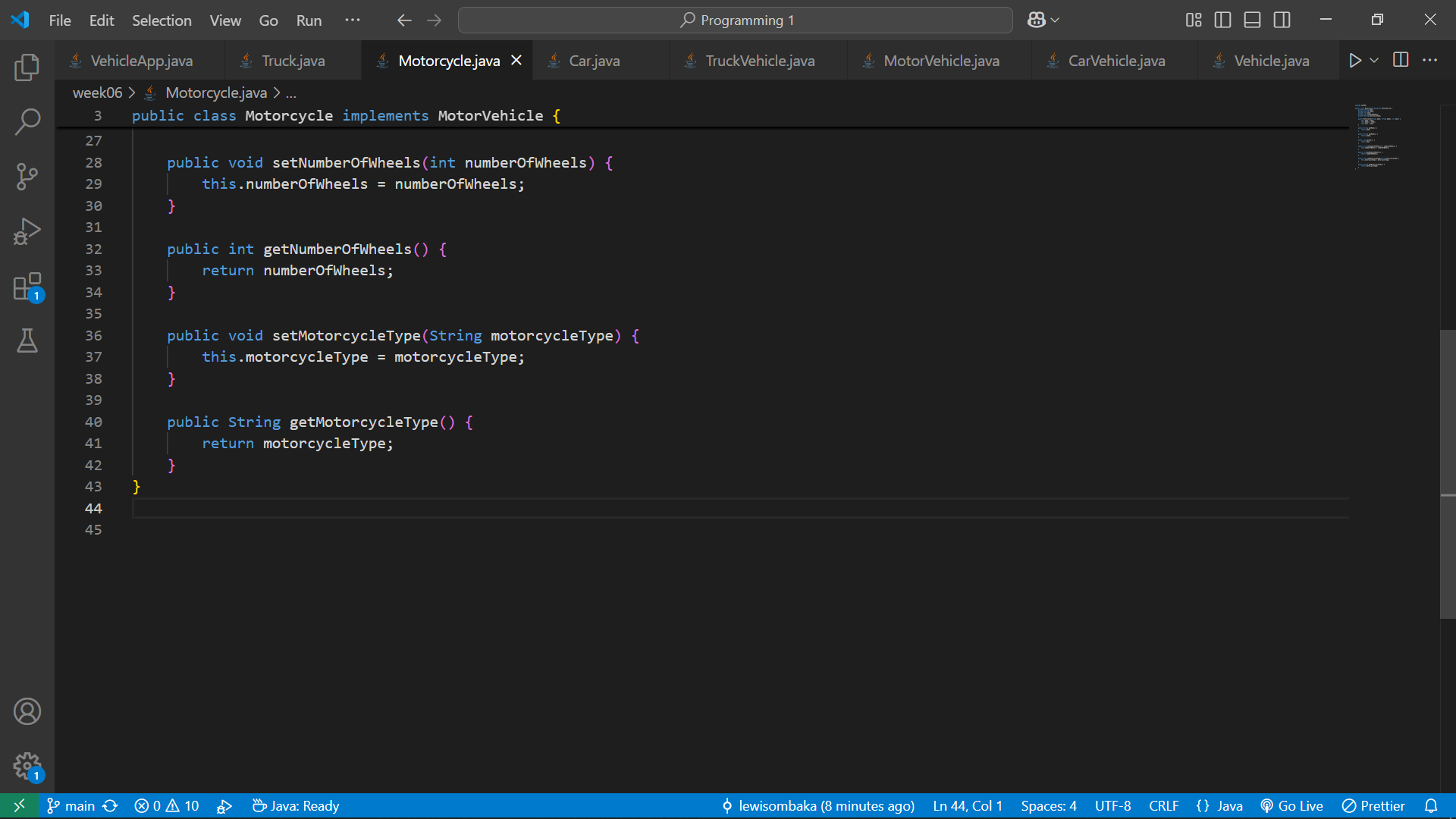
public String getMotorcycleType() {

return motorcycleType;

}

}





1. **Implement the** Truck **Class**

java

public class Truck implements 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 void setCargoCapacity(double cargoCapacity) {

this.cargoCapacity = cargoCapacity;

}

public double getCargoCapacity() {

return cargoCapacity;

}

public void setTransmissionType(String transmissionType) {

this.transmissionType = transmissionType;

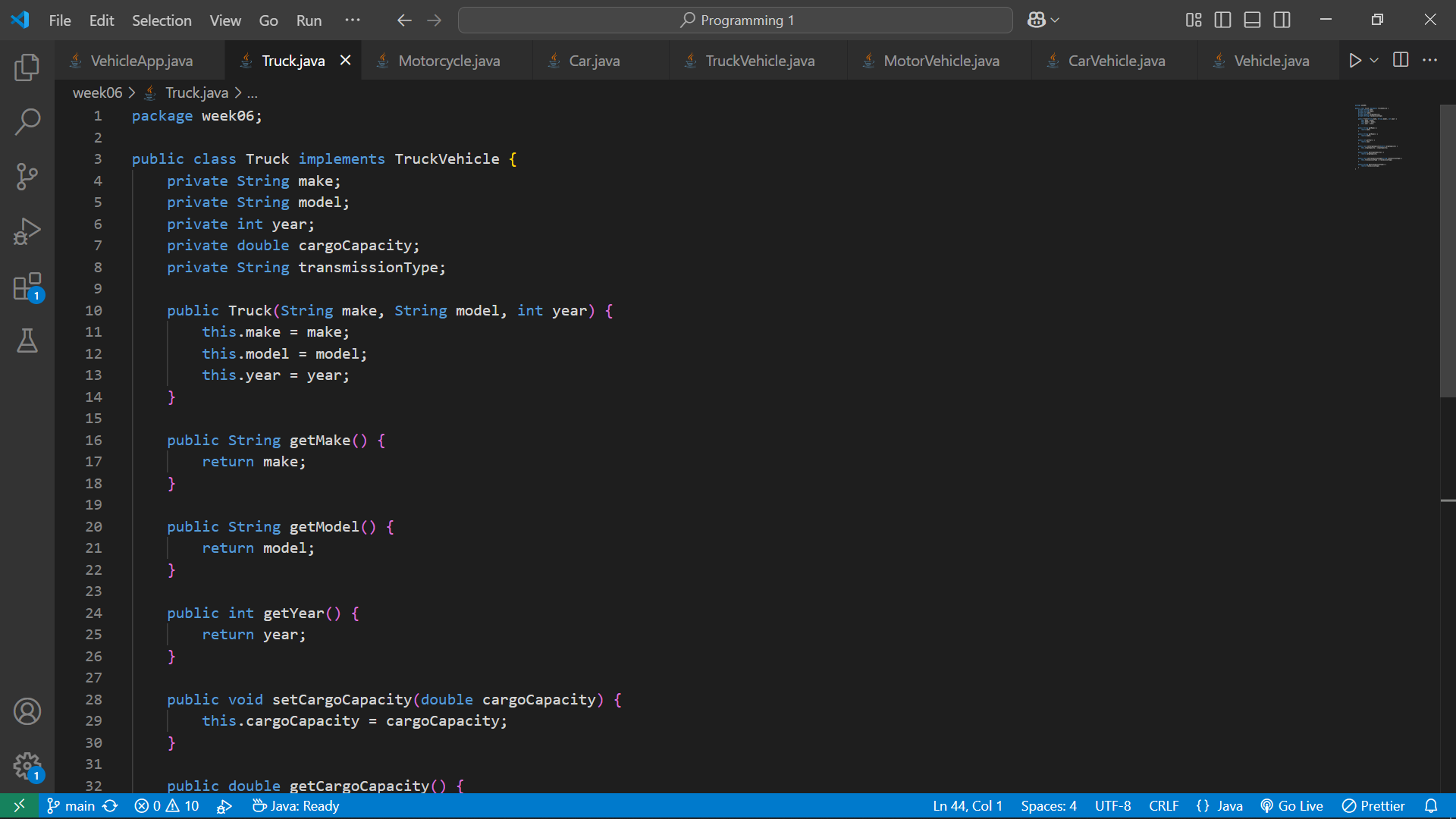
}

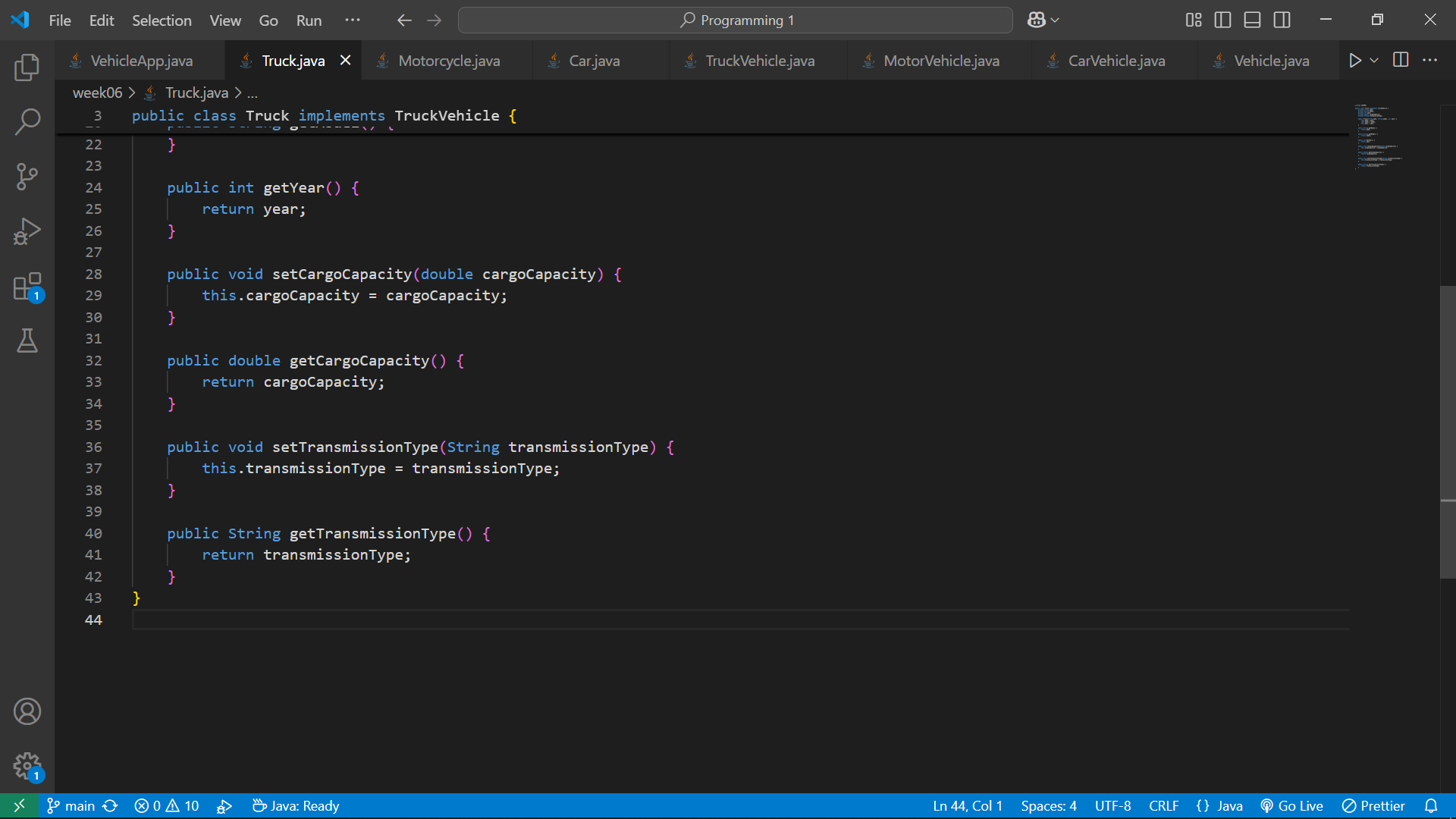
public String getTransmissionType() {

return transmissionType;

}

}





1. **Create the Main Program**

java

import java.util.Scanner;

public class VehicleApp {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Choose a vehicle type (car/motorcycle/truck): ");

String vehicleType = scanner.nextLine().toLowerCase();

Vehicle vehicle = null;

switch (vehicleType) {

case "car":

vehicle = createCar(scanner);

break;

case "motorcycle":

vehicle = createMotorcycle(scanner);

break;

case "truck":

vehicle = createTruck(scanner);

break;

default:

System.out.println("Invalid vehicle type");

System.exit(0);

}

if (vehicle != null) {

displayVehicleDetails(vehicle);

}

scanner.close();

}

private static Car createCar(Scanner scanner) {

System.out.println("Enter make: ");

String make = scanner.nextLine();

System.out.println("Enter model: ");

String model = scanner.nextLine();

System.out.println("Enter year: ");

int year = Integer.parseInt(scanner.nextLine());

Car car = new Car(make, model, year);

System.out.println("Enter number of doors: ");

int doors = Integer.parseInt(scanner.nextLine());

car.setNumberOfDoors(doors);

System.out.println("Enter fuel type (petrol/diesel/electric): ");

String fuelType = scanner.nextLine();

car.setFuelType(fuelType);

return car;

}

private static Motorcycle createMotorcycle(Scanner scanner) {

System.out.println("Enter make: ");

String make = scanner.nextLine();

System.out.println("Enter model: ");

String model = scanner.nextLine();

System.out.println("Enter year: ");

int year = Integer.parseInt(scanner.nextLine());

Motorcycle motorcycle = new Motorcycle(make, model, year);

System.out.println("Enter number of wheels: ");

int wheels = Integer.parseInt(scanner.nextLine());

motorcycle.setNumberOfWheels(wheels);

System.out.println("Enter motorcycle type (sport/cruiser/off-road): ");

String type = scanner.nextLine();

motorcycle.setMotorcycleType(type);

return motorcycle;

}

private static Truck createTruck(Scanner scanner) {

System.out.println("Enter make: ");

String make = scanner.nextLine();

System.out.println("Enter model: ");

String model = scanner.nextLine();

System.out.println("Enter year: ");

int year = Integer.parseInt(scanner.nextLine());

Truck truck = new Truck(make, model, year);

System.out.println("Enter cargo capacity (in tons): ");

double capacity = Double.parseDouble(scanner.nextLine());

truck.setCargoCapacity(capacity);

System.out.println("Enter transmission type (manual/automatic): ");

String transmissionType = scanner.nextLine();

truck.setTransmissionType(transmissionType);

return truck;

}

private static void displayVehicleDetails(Vehicle vehicle) {

System.out.println("\nVehicle Details:");

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

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

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

if (vehicle instanceof CarVehicle) {

CarVehicle car = (CarVehicle) vehicle;

System.out.println("Number of doors: " + car.getNumberOfDoors());

System.out.println("Fuel type: " + car.getFuelType());

} else if (vehicle instanceof MotorVehicle) {

MotorVehicle motorcycle = (MotorVehicle) vehicle;

System.out.println("Number of wheels: " + motorcycle.getNumberOfWheels());

System.out.println("Motorcycle type: " + motorcycle.getMotorcycleType());

} else if (vehicle instanceof TruckVehicle) {

TruckVehicle truck = (TruckVehicle) vehicle;

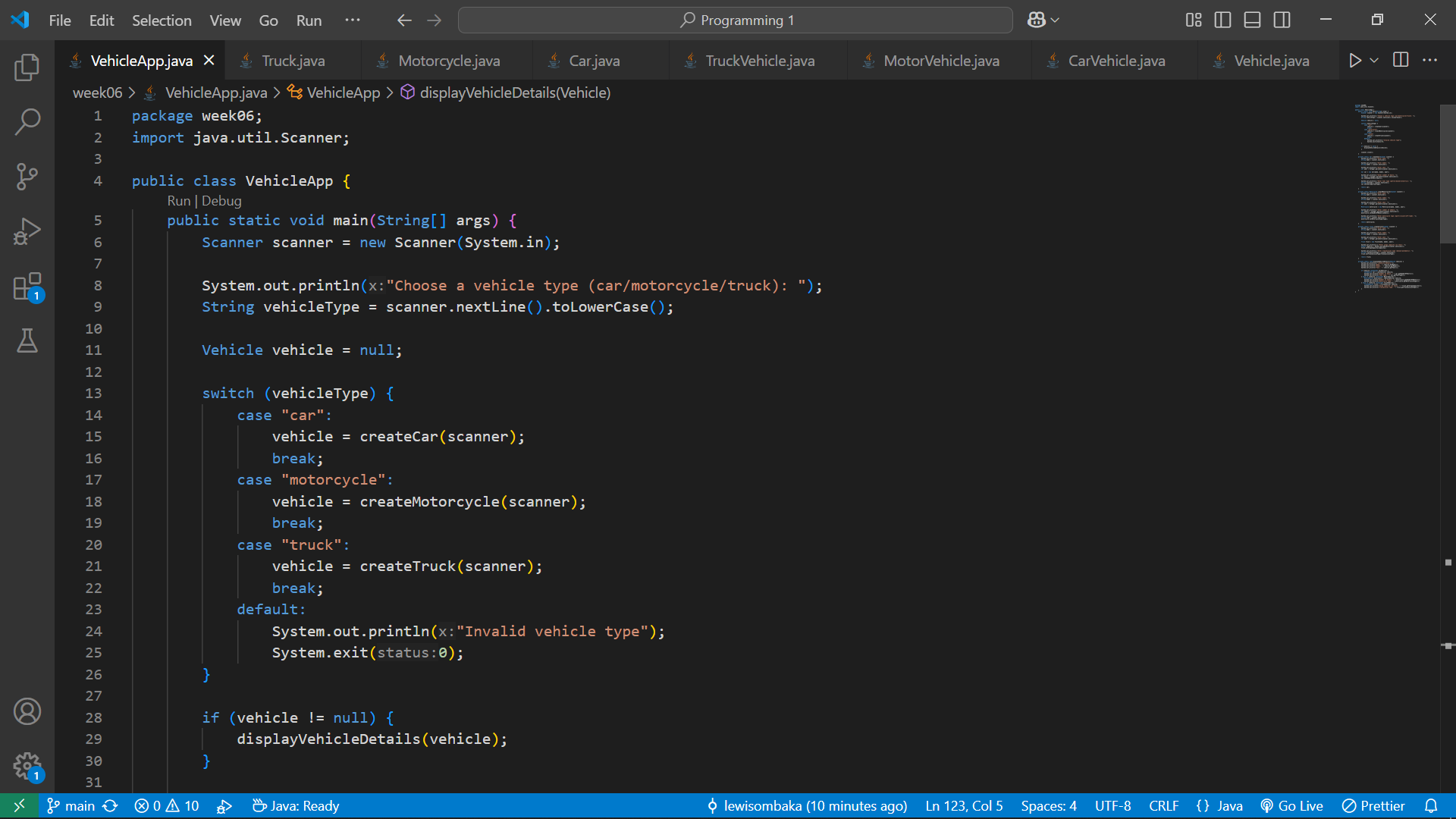
System.out.println("Cargo capacity (in tons): " + truck.getCargoCapacity());

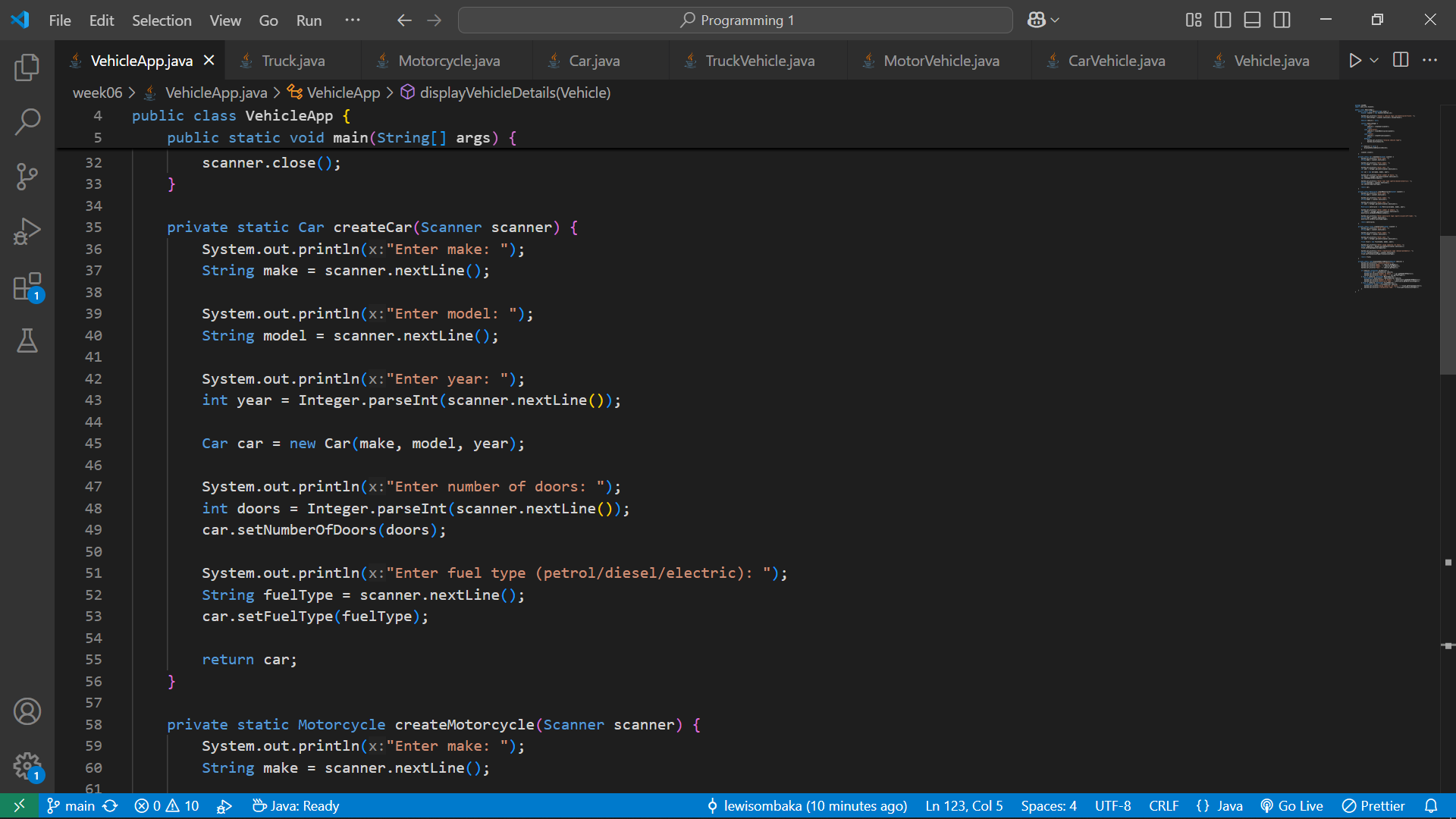
System.out.println("Transmission type: " + truck.getTransmissionType());

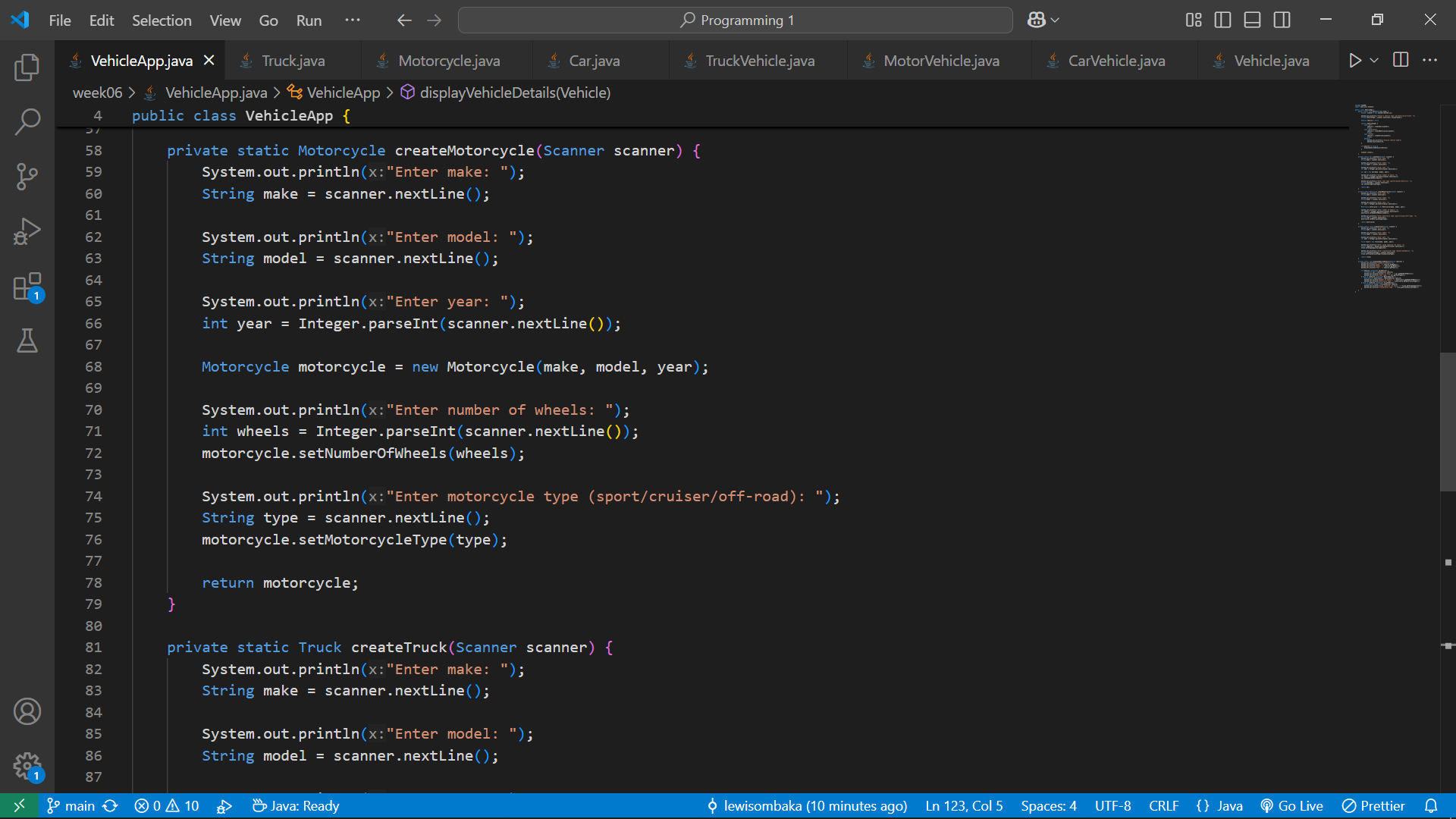
}

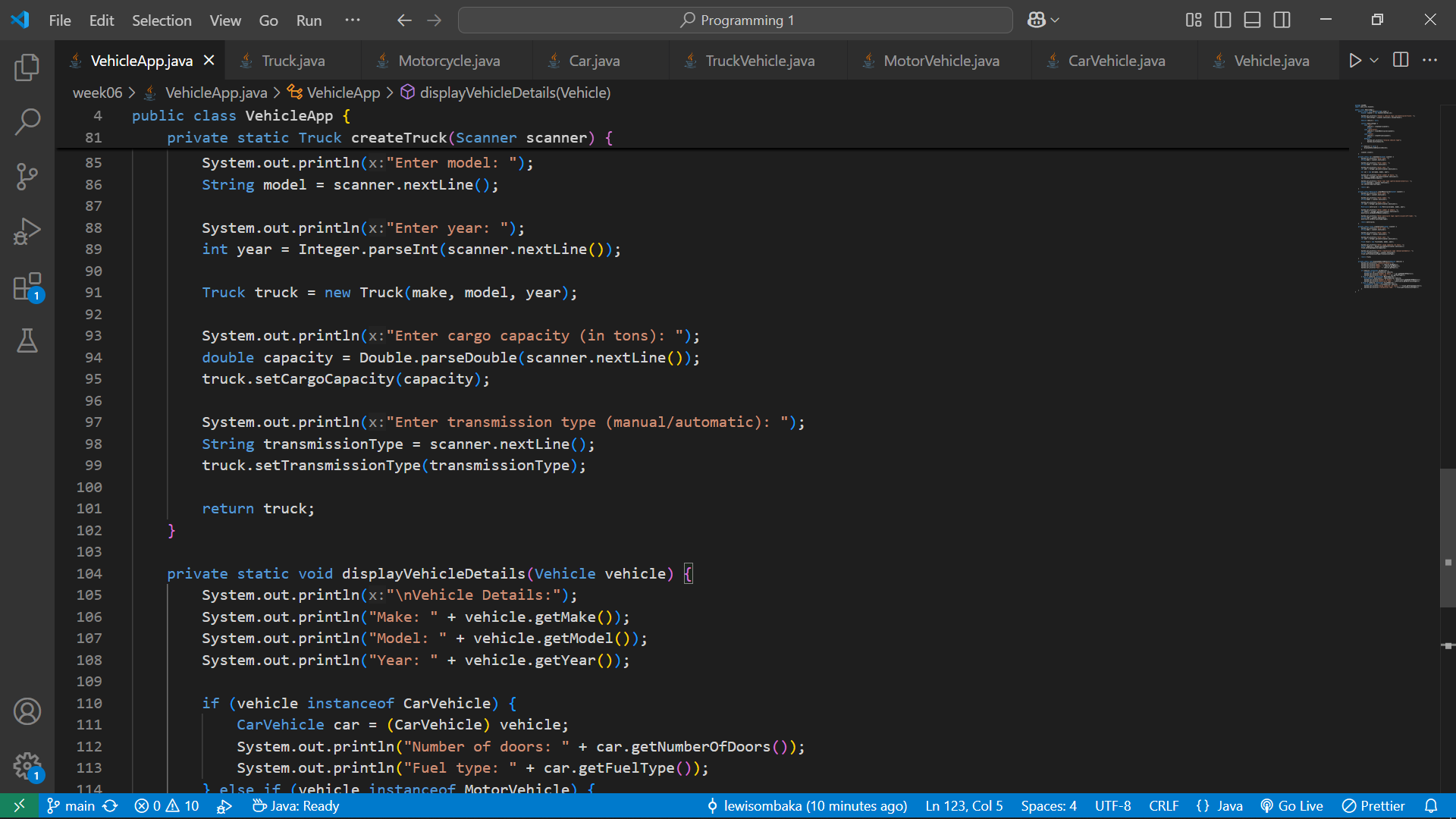
}

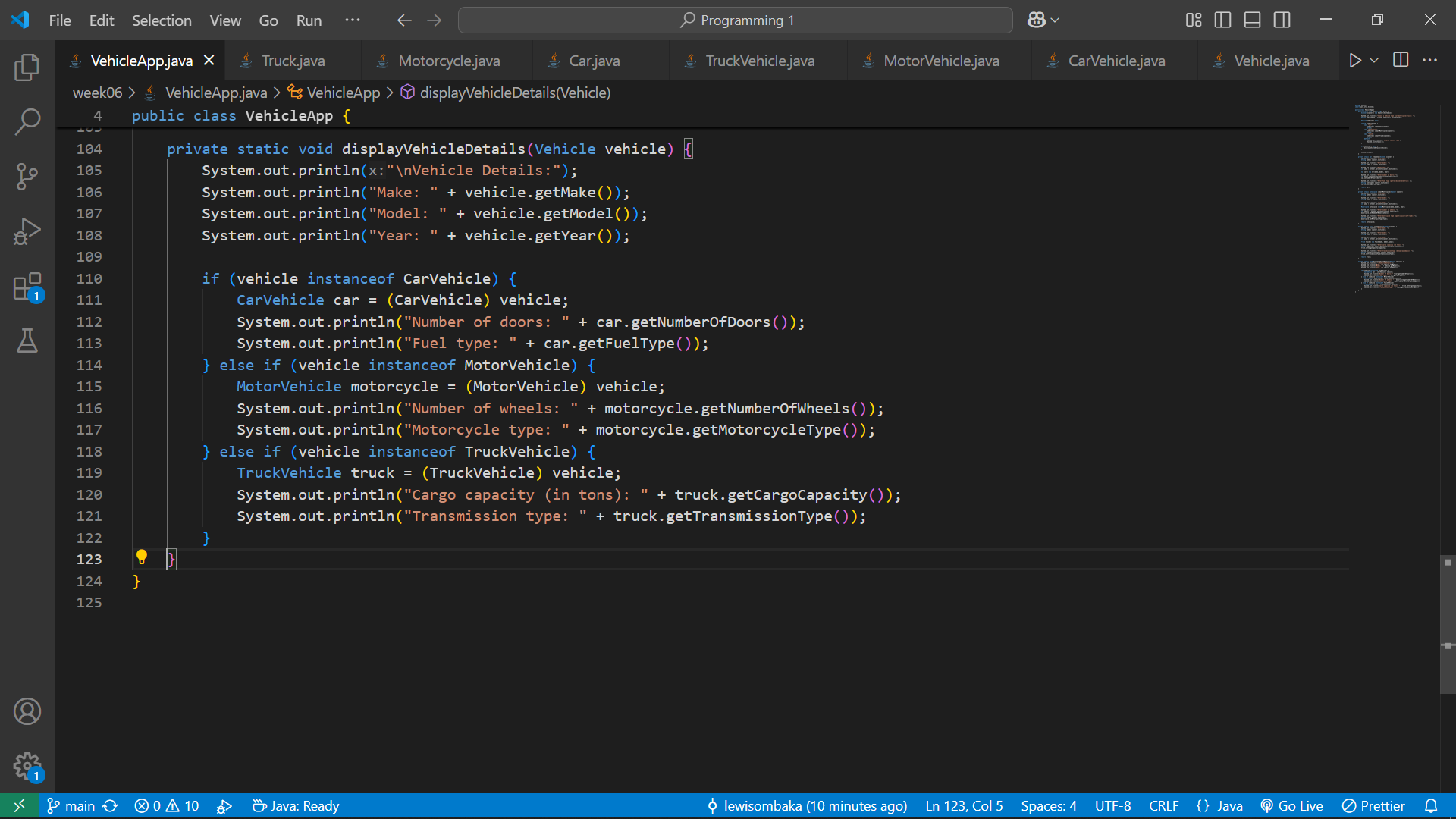
}











### **Explanation**

* **Interface Design**: The Vehicle interface establishes a contract specifying the methods for retrieving vehicle details. Additional interfaces like CarVehicle, MotorVehicle, and TruckVehicle ensure specific functionalities for each vehicle type.
* **Class Implementation**: The Car, Motorcycle, and Truck classes translate these specifications into concrete implementations, allowing storage and retrieval of specific attributes.
* **Main Program**: The VehicleApp class allows users to create and interact with different vehicle types, input relevant information, and display details of each vehicle. This ensures a comprehensive interface for managing vehicle information.
* **Code Quality**: Proper coding conventions, meaningful variable names, and appropriate indentation are used to enhance readability and maintainability.
* **Error Handling**: The program handles invalid user inputs gracefully using simple checks and exits if an invalid vehicle type is entered.
* **Documentation**: Clear class/method names are provided to aid in understanding the code.

### **Output Screenshot**

When running the program, the user is prompted to enter information for the selected vehicle type, and the details are displayed. Here’s an example output for creating a car:

Choose a vehicle type (car/motorcycle/truck): car

Enter make: Toyota

Enter model: Corolla

Enter year: 2020

Enter number of doors: 4

Enter fuel type (petrol/diesel/electric): petrol

Vehicle Details:

Make: Toyota

Model: Corolla

Year: 2020

Number of doors: 4

Fuel type: petrol

