```java
import java.util.UUID;

public class Vehicle {
    // Protected fields for inheritance
    protected String brand;
    protected String model;
    protected int year;
    protected String engineType;

    // Private fields requiring getter/setter access
    private String registrationNumber;
    private boolean isRunning;

    // Default constructor
    public Vehicle() {
        this.brand = "Unknown";
        this.model = "Unknown";
        this.year = 0;
        this.engineType = "Unknown";
        this.registrationNumber = "N/A";
        this.isRunning = false;
        System.out.println("Vehicle default constructor called");
    }

    // Parameterized constructor
    public Vehicle(String brand, String model, int year, String
engineType) {
        this.brand = brand;
        this.model = model;
        this.year = year;
        this.engineType = engineType;
        this.registrationNumber = UUID.randomUUID().toString();
        this.isRunning = false;
        System.out.println("Vehicle parameterized constructor called");
    }

    // Method to start the vehicle
    public void start() {
        this.isRunning = true;
        System.out.println("Vehicle started");
```

```
    }

    // Method to stop the vehicle
    public void stop() {
        this.isRunning = false;
        System.out.println("Vehicle stopped");
    }

    // Method to get vehicle information
    public String getVehicleInfo() {
        return "Brand: " + brand + ", Model: " + model + ", Year: " + year
+
               ", Engine Type: " + engineType + ", Registration Number: "
+ registrationNumber +
               ", Is Running: " + isRunning;
    }

    // Method to display technical specifications
    public void displaySpecs() {
        System.out.println("--- Vehicle Specifications ---");
        System.out.println("Brand: " + brand);
        System.out.println("Model: " + model);
        System.out.println("Year: " + year);
        System.out.println("Engine Type: " + engineType);
    }

    // Getter for registrationNumber
    public String getRegistrationNumber() {
        return registrationNumber;
    }

    // Setter for registrationNumber
    public void setRegistrationNumber(String registrationNumber) {
        this.registrationNumber = registrationNumber;
    }

    // Getter for isRunning
    public boolean isRunning() {
        return isRunning;
    }
```

```
}

public class Car extends Vehicle {
    // Car-specific fields
    private int numberOfDoors;
    private String fuelType;
    private String transmissionType;

    // Default constructor
    public Car() {
        super(); // Explicit call to parent default constructor
        this.numberOfDoors = 4;
        this.fuelType = "Gasoline";
        this.transmissionType = "Automatic";
        System.out.println("Car default constructor called");
    }

    // Parameterized constructor
    public Car(String brand, String model, int year, String engineType,
int numberOfDoors, String fuelType, String transmissionType) {
        super(brand, model, year, engineType); // Explicit call to parent
parameterized constructor
        this.numberOfDoors = numberOfDoors;
        this.fuelType = fuelType;
        this.transmissionType = transmissionType;
        System.out.println("Car parameterized constructor called");
    }

    // Override start() method
    @Override
    public void start() {
        super.start(); // Call parent's start method
        System.out.println("Car-specific startup sequence: Checking all
systems.");
    }

    // Override displaySpecs() method
    @Override
    public void displaySpecs() {
        super.displaySpecs(); // Call parent's displaySpecs method
```

```
        System.out.println("--- Car Specific Specifications ---");
        System.out.println("Number of Doors: " + numberOfDoors);
        System.out.println("Fuel Type: " + fuelType);
        System.out.println("Transmission Type: " + transmissionType);
    }

    // Car-specific method
    public void openTrunk() {
        System.out.println("Trunk opened");
    }

    // Car-specific method
    public void playRadio() {
        System.out.println("Radio playing music");
    }

    public static void main(String[] args) {
        System.out.println("--- Testing Constructor Chaining (Default)
---");
        Car defaultCar = new Car();
        System.out.println("\n--- Testing Constructor Chaining
(Parameterized) ---");
        Car customCar = new Car("Toyota", "Camry", 2023, "V6", 4,
"Gasoline", "Automatic");

        System.out.println("\n--- Testing Inheritance of Fields and
Methods ---");
        System.out.println("Accessing protected field from parent: " +
customCar.brand);
        System.out.println("Calling inherited method from parent:");
        System.out.println(customCar.getVehicleInfo());

        System.out.println("\n--- Testing super Keyword and Method
Overriding ---");
        System.out.println("Calling overridden start() method:");
        customCar.start();
        System.out.println("Is vehicle running? " +
customCar.isRunning());
        System.out.println("\nCalling overridden displaySpecs() method:");
        customCar.displaySpecs();
```

```java
        System.out.println("\n--- Testing Method Resolution ---");
        System.out.println("Calling method that only exists in Car:");
        customCar.openTrunk();
        customCar.playRadio();

        System.out.println("\n--- Polymorphic Behavior ---");
        Vehicle myVehicle = new Car("Honda", "Civic", 2022, "I4", 4,
"Petrol", "CVT");
        myVehicle.start(); // Calls the overridden start() method in Car
        myVehicle.displaySpecs(); // Calls the overridden displaySpecs()
method in Car
        // myVehicle.openTrunk(); // This would cause a compile error
because openTrunk() is not in Vehicle

        System.out.println("\n--- Final State of customCar ---");
        customCar.stop();
        System.out.println(customCar.getVehicleInfo());
    }
}
```

```
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\practice\Q1> java Car
--- Vehicle Specifications ---
Brand: Toyota
Model: Camry
Year: 2023
Engine Type: V6
--- Car Specific Specifications ---
Number of Doors: 4
Fuel Type: Gasoline
Transmission Type: Automatic

--- Testing Method Resolution ---
Calling method that only exists in Car:
Trunk opened
Radio playing music

--- Polymorphic Behavior ---
Vehicle parameterized constructor called
Car parameterized constructor called
Vehicle started
Car-specific startup sequence: Checking all systems.
--- Vehicle Specifications ---
Brand: Honda
Model: Civic
Engine Type: I4
--- Car Specific Specifications ---
Number of Doors: 4
Fuel Type: Petrol
Transmission Type: CVT

--- Final State of customCar ---
Vehicle stopped
Brand: Toyota, Model: Camry, Year: 2023, Engine Type: V6, Registration Number: 5a028605-2947-4cb7-88f8-49a7efbdd1a6, Is Running: false
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\practice\Q1>
```

```java
class Animal {
    void eat() {
        System.out.println("Animal eats.");
    }
}

class Mammal extends Animal {
    void walk() {
        System.out.println("Mammal walks.");
    }
}

class Dog extends Mammal {
    void bark() {
        System.out.println("Dog barks.");
    }
}

public class MultilevelInheritanceDemo {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();   // from Animal
        dog.walk();  // from Mammal
        dog.bark();  // from Dog
    }
}
```

```
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week6\practice\q2> java MultilevelInheritanceDemo
Animal eats.
Mammal walks.
Dog barks.
```