

Course: Object-Oriented Programming (OOP) with Java

Lab: Classes & Objects

Released Date: January 13, 2026 @ 1:00PM

Due Date: January 14, 2026 @ 2:00PM

Lecturer: Julius Aries Kanneh, Jr

Email: juliuskanneh7@gmail.com

Lab: Classes & Objects

Objective:

- Understand how to define and instantiate classes in Java.
 - Learn to create objects using constructors.
 - Get hands-on experience working with instance variables and methods.
-

Part 1: Create a Class - **Car**

Step 1: Define the **Car** Class

1. Open your Java IDE (IntelliJ IDEA, Eclipse, or any other editor you are using).
2. Create a new Java class named **Car**.
3. Inside the **Car** class, define three instance variables (fields) to store:
 - **brand** (type: String)
 - **model** (type: String)
 - **year** (type: int)
4. Your code should look like this:

```
Java
public class Car {
    // Instance variables
    String brand;
    String model;
    int year;
}
```

Step 2: Add a Constructor

1. Now, we will add a **constructor** to the **Car** class. The constructor will initialize the **brand**, **model**, and **year** variables when a new **Car** object is created.
2. The constructor will look like this:

Java

```
public Car (String brand, String model, int year) {  
    this.brand = brand;  
    this.model = model;  
    this.year = year;  
}
```

Here, **this** refers to the instance of the object being created. It is used to distinguish between the instance variables and the parameters.

Step 3: Add a Method to the **Car** Class

1. Next, we will add a method called **start()**. This method will print a message to the console indicating that the car is starting.
2. The method definition will look like this:

Java

```
// Method to start the car  
public void start() {  
    System.out.println(brand + " " + model + " is starting.");  
}
```

Step 4: Create a **Main** Class to Test Your **Car** Class

1. Now, let's create a **Main** class where we will instantiate the **Car** objects and call the **start()** method.
2. Create a new class called **Main** and inside it, write the following code:

Java

```
public class Main {  
    public static void main(String[] args) {  
        // Creating a new Car object  
        Car myCar = new Car("Toyota", "Camry", 2020);  
    }  
}
```

```
        // Calling the start() method
        myCar.start();
    }
}
```

This code:

- Creates a new object `myCar` using the `Car` constructor.
- Calls the `start()` method of the `Car` class, which prints out a message.

Part 2: Modify and Expand the `Car` Class

Step 1: Add More Methods to `Car`

1. Now that the basic `Car` class works, let's add a method to display the details of the car (brand, model, and year). This method will be called `displayDetails()`.
2. Add the following method to the `Car` class:

Java

```
// Method to display car details
public void displayDetails() {
    System.out.println("Car Details: ");
    System.out.println("Brand: " + brand);
    System.out.println("Model: " + model);
    System.out.println("Year: " + year);
}
```

Step 2: Modify `Main` Class to Test `displayDetails()`

1. Now, modify the `Main` class to call the `displayDetails()` method after calling `start()`:

Java

```
public class Main {
    public static void main(String[] args) {
        // Creating a new Car object
        Car myCar = new Car("Toyota", "Camry", 2020);
    }
}
```

```

        // Calling the start() method
        myCar.start();

        // Calling the displayDetails() method
        myCar.displayDetails();
    }
}

```

Part 3: Additional Challenge (Optional)

1. **Task:** Modify the **Car** class to include a **fuel efficiency** field (miles per gallon) and a **fuel level** field (in gallons).
2. **New Method:** Add a method **drive(int miles)** that decreases the fuel level based on the miles driven and fuel efficiency. If the fuel level reaches 0, display a message saying the car is out of fuel.

Example Method:

```

Java
public void drive(int miles) {
    double fuelConsumed = miles / (double) fuelEfficiency;
    if (fuelLevel >= fuelConsumed) {
        fuelLevel -= fuelConsumed;
        System.out.println("You drove " + miles + " miles.");
        System.out.println("Remaining fuel: " + fuelLevel + "
gallons.");
    } else {
        System.out.println("Not enough fuel to drive that far.");
    }
}

```

Key Concepts Covered in the Lab:

- **Instance Variables:** How to define and use instance variables.
- **Constructor:** Initializing object state with the constructor.
- **Methods:** Defining and using methods within a class.
- **Object Instantiation:** Creating objects from a class and interacting with them.

Submission Instructions:

- Submit your Java code (`Car.java` and `Main.java`) after completing all the tasks.
- For the additional challenge, submit the code with the fuel efficiency and drive method implemented.

Resources Needed:

- Java Development Kit (JDK).
- IDE (IntelliJ IDEA, Eclipse, or NetBeans).
- Access to the internet (for research and learning materials).