Java: Class and Object

1. What is a Class?
A **class** is a blueprint or template that defines the structure (attributes) and behavior (methods) of objects. It provides a way to group related data and functions together.

- A **class** doesn't consume memory until objects are created from it.
- It is defined using the `class` keyword.

Syntax:

```
class ClassName {
    // Attributes (fields)
    DataType attributeName;

    // Methods (behavior)
    returnType methodName(parameters) {
        // Method body
    }
}
```

Example:

```
class Car {
    // Attributes
    String brand;
    int speed;

    // Constructor
    Car(String brand, int speed) {
        this.brand = brand;
        this.speed = speed;
    }

    // Method
    void displayDetails() {
         System.out.println("Brand: " + brand + ", Speed: " + speed + "
km/h");
    }
}
```

---

2. What is an Object?

An **object** is an instance of a class that represents a real-world entity. It has:

- **State**: Defined by the values of its attributes.
- **Behavior**: Defined by the methods of the class.

Syntax:

```
ClassName objectName = new ClassName(parameters);
```

Example:

```
public class Main {
    public static void main(String[] args) {
        // Creating an object of the Car class
        Car myCar = new Car("Tesla", 200);
          myCar.displayDetails();   // Output: Brand: Tesla, Speed: 200
km/h
    }
}
```

Class vs. Object

Aspect	Class	Object
Definition	A blueprint or template for creating objects.	An instance of a class.
Memory	Does not occupy memory.	Occupies memory when created.
Usage	Defines attributes and behaviors.	Represents specific attributes and behaviors.
Example	`class Car {}`	`Car myCar = new Car();`

Key Concepts

Attributes (Fields):

- Represent the data or state of the object.
- Example: `String brand`, `int speed`.

Methods:

- Define the behavior or actions of the object.
- Example: `void displayDetails()`.

Constructor:

- A special method used to initialize the attributes of an object.

Example:

```
Car(String brand, int speed) {
    this.brand = brand;
    this.speed = speed;
}
```

Memory Allocation:

- When an object is created, memory is allocated for its attributes in the heap.

Best Practices

Use meaningful class names:

```
class Car {}  // Good
class C {}    // Bad
```

1. Encapsulate data by marking attributes as `private` and providing getter and setter methods.
2. Follow proper naming conventions for attributes and methods (`camelCase`).
3. Always provide a default or parameterized constructor.

Programming Practice

Program 1: Displaying the Details of a Student

```
class Student {
    String name;
    int rollNumber;
    double marks;

    // Constructor
    Student(String name, int rollNumber, double marks) {
        this.name = name;
        this.rollNumber = rollNumber;
        this.marks = marks;
    }

    // Method to display student details
    void displayDetails() {
            System.out.println("Name: " + name + ", Roll Number: " +
rollNumber + ", Marks: " + marks);
    }
}

public class Main {
    public static void main(String[] args) {
        Student student = new Student("Alice", 101, 89.5);
            student.displayDetails();   // Output: Name: Alice, Roll
Number: 101, Marks: 89.5
    }
}
```

Program 2: Travel Details

```
class TravelDetails {
    String fromCity, toCity;
    double distance;
    // Constructor
    TravelDetails(String fromCity, String toCity, double distance) {
        this.fromCity = fromCity;
        this.toCity = toCity;
        this.distance = distance;
    }

    // Method to display travel information
    void displayTravelInfo() {
           System.out.println("Traveling from " + fromCity + " to " +
toCity + " covers " + distance + " km.");
    }
}

public class Main {
    public static void main(String[] args) {
                TravelDetails travel = new TravelDetails("Chennai",
"Bangalore", 345.6);
         travel.displayTravelInfo();  // Output: Traveling from Chennai
to Bangalore covers 345.6 km.
    }
}
```