Creating a Class in Java

Properties and Methods

Lecture Overview

1. Introduction to Classes in Java
2. Understanding Properties (Fields)
3. Understanding Methods
4. Creating a Class: Step-by-Step Example
5. Access Modifiers and Encapsulation
6. Key Points and Best Practices
7. Q&A

Introduction to Classes

What is a Class?

A blueprint for creating objects.

Defines properties (attributes) and behaviors (methods) of objects.

Key Components:

Properties (Fields/Instance Variables)

Methods (Functions inside the class)

Example of a Class

```
class Car {
    // Properties
    String make;
    String model;
    int year;
    // Method
    void startEngine() {
        System.out.println("Engine started.");
    }
}
```

Explanation:

`Car`: Class name.

Properties: `make`, `model`, `year`.

Method: `startEngine()`.

Understanding Properties

What are Properties (Fields)?

- Variables that define the state of an object.
- Defined inside the class but outside any methods.

Example:

String color;

int speed;

Types of Fields:

- Instance Variables
- Static Variables (shared across all objects

Defining Properties in a Class

Syntax:

```
class MyClass {
    // Field Declaration
    String propertyName;
}
```

Example:

```
class Employee {
    String name;
    int employeeId;
}
```

Understanding Methods

What are Methods?

- Functions that define the behavior of objects.
- Methods operate on the object’s data.
- Methods can take parameters and return values.

Example:

```
void displayDetails() {
    System.out.println("Displaying details...");
}
```

Defining Methods in a Class

Syntax:

```
returnType methodName(parameters)
{
    // method body
}
```

Example:

```
class Calculator {
    int add(int a, int b) {
        return a + b;
    }
}
```

Creating a Class

Creating a Class: `Student`

```
class Student {
    // Properties
    String name;
    int rollNumber;

    // Method
    void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Roll Number: " +
rollNumber);
    }
}
```

Explanation:

Properties: `name`, `rollNumber`.

Method: `displayInfo()`.

The `Student` class represents a student object.

Access Modifiers and Encapsulation

Access Modifiers:

- `public`, `private`, `protected`, default (no modifier).
- Control access to fields and methods.

Encapsulation:

- Restrict direct access to fields.
- Use getter and setter methods.

```
public class Person {
    private String name;

    // Getter
    public String getName() {
        return name;
    }

    // Setter
    public void setName(String newName) {
        name = newName;
    }
}
```

Example with Encapsulation

```
class Account {
    private double balance;
    // Getter
    public double getBalance() {
        return balance;
    }
    // Setter
    public void deposit(double amount) {
        if(amount > 0) {
            balance += amount;
        }
    }
}
```

- **Explanation**:
  - `balance` is private, protecting it from direct modification.
  - Methods control access to `balance`.

Key Points and Best Practices

- Use descriptive names for properties and methods.
- Group related behaviors in methods.
- Keep fields private and use getters and setters.
- Avoid overly complex methods; break them into smaller ones if necessary.
- Leverage access modifiers for encapsulation.

Q&A

Open the floor for questions from the audience.