OOP Concepts in Java

1. Abstraction
   - Hides implementation details, shows only essential features.
   - Achieved using abstract classes and interfaces.
   Example:

```
interface Shape {
    void draw();
}
class Circle implements Shape {
    public void draw() {
        System.out.println("Drawing Circle");
    }
}
```

2. Encapsulation
   - Wrapping data (fields) and methods together, restricting direct access.
   - Achieved by making fields private and providing getters/setters.
   Example:

```
class Student {
    private String name;
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

3. Inheritance
   - One class acquires properties and behaviors of another (IS-A relationship).
   - Achieved using 'extends' or 'implements'.
   Example:

```
class Animal {
    void sound() { System.out.println("Animal makes sound"); }
}
class Dog extends Animal {
    void sound() { System.out.println("Dog barks"); }
}
```

4. Polymorphism
   - One interface, many implementations (same method behaves differently).
   - Types:
     * Compile-time (Overloading)
     * Runtime (Overriding)
   Example:

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

```
}

class Animal {
    void sound() { System.out.println("Animal sound"); }
}
class Cat extends Animal {
    void sound() { System.out.println("Cat meows"); } // Overriding
}
```

Summary:
- Abstraction → Hides 'what' & 'how'.
- Encapsulation → Protects data.
- Inheritance → Reuses code.
- Polymorphism → Many forms of behavior.