Phase 6: Abstraction in Java

What is Abstraction?

Abstraction is the process of hiding the internal implementation details and showing only the functionality to the user.

It helps reduce complexity and increase efficiency in system design.

Abstract Class

- Defined using the 'abstract' keyword
- Can contain both abstract and non-abstract methods
- Cannot be instantiated directly
- Subclasses must implement all abstract methods unless they are abstract themselves

Example: Abstract Class

```
abstract class Appliance {
    abstract void turnOn();
    abstract void turnOff();
}

class WashingMachine extends Appliance {
    void turnOn() { System.out.println("Washing Machine is turned on."); }
    void turnOff() { System.out.println("Washing Machine is turned off."); }
}
```

Interface

- An interface is a completely abstract class
- Contains only abstract methods (Java 7) or also default/static methods (Java 8+)
- Supports multiple inheritance
- Cannot have constructors

Phase 6: Abstraction in Java

Example: Interface

```
interface Remote {
    void turnOn();
    void turnOff();
}

class SmartTV implements Remote {
    public void turnOn() { System.out.println("Smart TV is turned on."); }
    public void turnOff() { System.out.println("Smart TV is turned off."); }
}
```

Abstract Class vs Interface

- Abstract class can have state (instance variables), interface cannot (before Java 8)
- Interface supports multiple inheritance; abstract class doesn't
- Use interface to define capabilities, abstract class to share common base functionality