Advanced Programming

Interfaces in Java — Extended Edition

Instructor: Ali Najimi

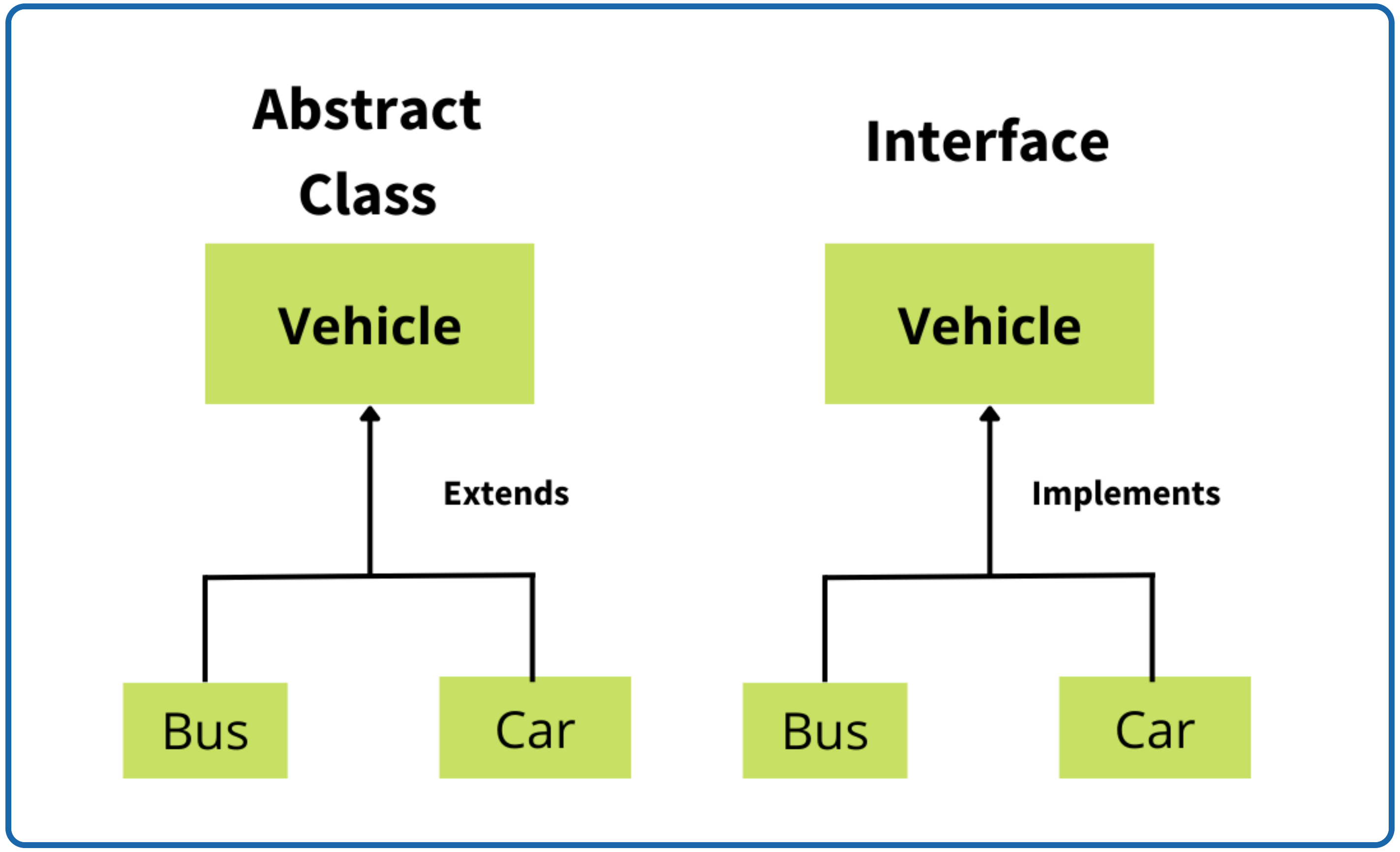
Author: Hossein Masihi

Department of Computer Engineering

Sharif University of Technology

Fall 2025

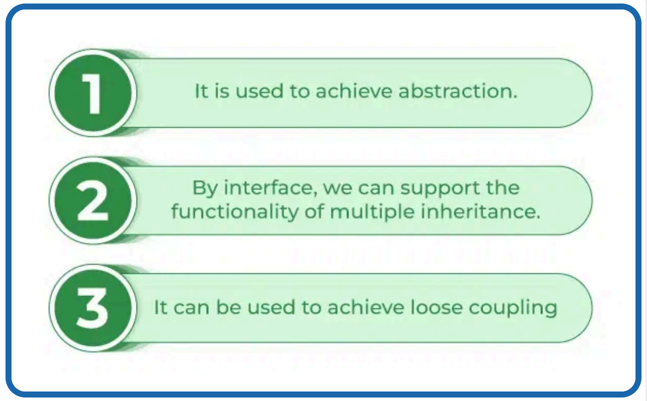
Table of Contents


Abstract Class
Vehicle
Extends
Bus
Car
Interface
Vehicle
Implements
Bus
Car

Interface — Concept

- An **interface** defines a set of **method signatures**
- Specifies *what* a class must do — not *how*
- Classes **implement** interfaces
- Enables **polymorphism** through behavior contracts

> Interfaces represent **capabilities**, not structures



1. It is used to achieve abstraction.
2. By interface, we can support the functionality of multiple inheritance.
3. It can be used to achieve loose coupling

Declaring and Implementing an Interface

```java
interface Shape {
    double area();
    double perimeter();
}
```

```java
class Circle implements Shape {
    private double r;

    Circle(double r) { this.r = r; }

    public double area() { return Math.PI * r * r; }
    public double perimeter() { return 2 * Math.PI * r; }
}
```

> A class must implement **all** interface methods

Why Use Interfaces?

Benefit	Description
Abstraction	Hide implementation details
Loose Coupling	Reduces dependency between components
Extensibility	Easy replacement of behavior
Polymorphism	Enables behavior switching at runtime
Framework Foundation	Core principle behind Spring / Jakarta EE

Default and Static Methods (Java 8+)

```java
interface Logger {
    void log(String message);
    default void info(String message) {
        log("INFO: " + message);
    }
    static void help() {System.out.println("Logger usage help");}
}
```

```java
class ConsoleLogger implements Logger {
    public void log(String message) { System.out.println(message); }
}
```

> `default` = shared behavior
> `static` = utility function at interface level

Interface vs Abstract Class

Feature	Interface	Abstract Class
State (Fields)	Only constants	Can have instance variables
Method Implementation	Only default allowed	Can contain method bodies
Constructors	Not allowed	Allowed
Multiple Inheritance	Allowed	Not allowed
Best Usage	Behavior contract	Base implementation template

Multiple Interface Implementation

```java
interface Flyable { void fly(); }
interface Swimmable { void swim(); }

class Duck implements Flyable, Swimmable {
    public void fly() { System.out.println("Duck flying"); }
    public void swim() { System.out.println("Duck swimming"); }
}
```

> Enables **multi-capability classes**

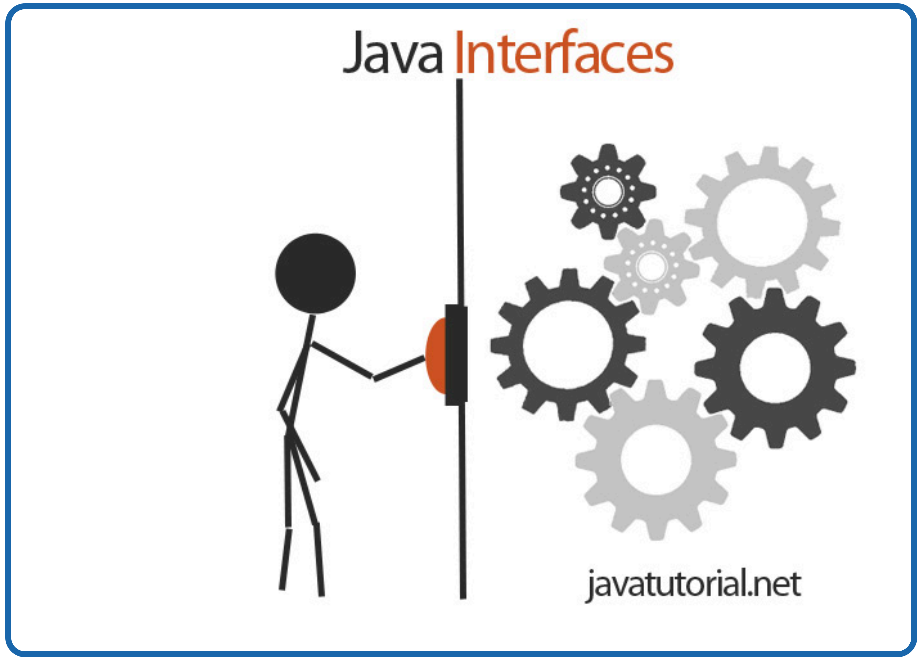
Summary

Concept	Description
Interface	Behavioral contract (what to do)
Implementation	Provided by classes (how to do it)
Default Methods	Shared optional behavior
Multiple Interfaces	Enables multiple capabilities
Goal	Flexibility, extensibility, maintainability

> Interfaces are critical for clean, scalable OOP design

Thank You!

Interfaces in Java — Extended



Advanced Programming — Fall 2025 — Sharif University of Technology