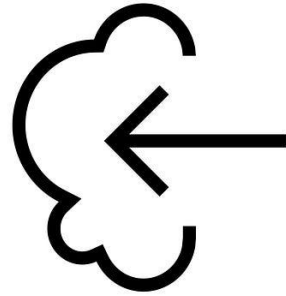


**Abstract class**



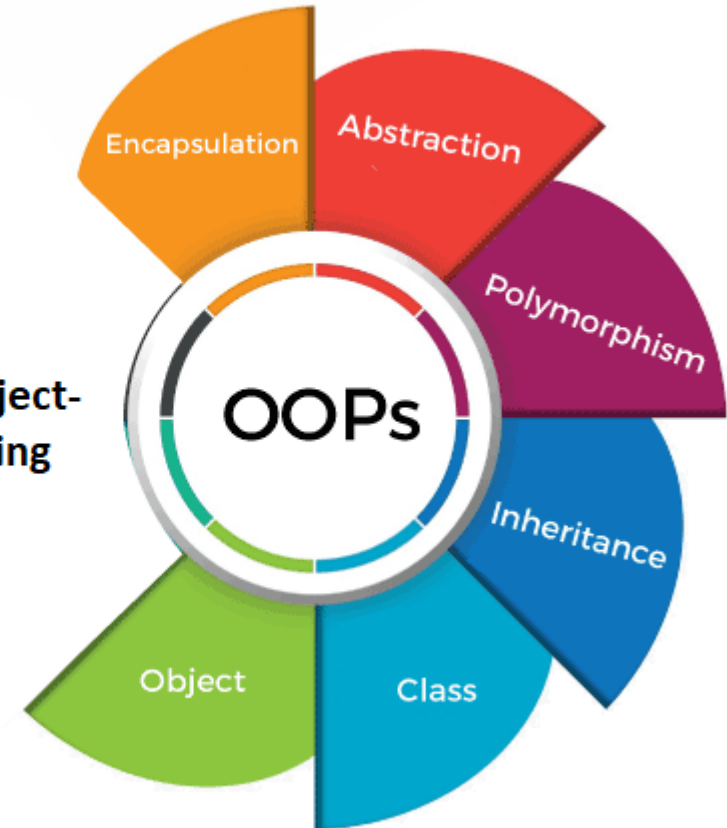
**Interface**

# **Abstract Classes & Interfaces in Java**

# Quick Recall: OOP

- Encapsulation
- Inheritance
- Polymorphism
- **Abstraction** → today's focus

Fundamentals of object-oriented programming



# What is Abstraction?

- Providing only the essential details to the user by **hiding** the unnecessary or irrelevant details of an entity.
- This helps in **reducing the operational complexity** at the user-end.
- Example: Giving the user the ability to drive the car without requiring to understand tiny details of ‘how does the engine work’.

# Abstract Classes

- Declared with '**abstract**' keyword
- Can have **abstract + non-abstract methods**
- **Cannot be instantiated – objects cannot be created**
- Must be **extended by subclasses**

# Abstract Class Syntax

```
//abstract class
abstract class Shape {

    //abstract method
    abstract double area();

    //non-abstract method
    void display() {
        System.out.println("Calculate area....");
    }
}
```

# Abstract Class Properties

- When a class contains **one or more abstract methods**, it should be **declared as abstract class**.
- The abstract methods of an **abstract class** must be defined in its subclass.
- We **cannot declare abstract constructors or abstract static methods**.
- A subclass of an abstract class can be instantiated if it overrides all abstract methods by implementation them

# Interfaces

- *Interface* is a conceptual entity similar to a Abstract class.
- Can contain only **constants (final variables)** and **abstract method** (no implementation) - Different from Abstract classes.
- An interface is basically a kind of class, but they have to be only abstract classes and final fields/variables.
- A class can implement any number of interfaces, but cannot extend more than one class at a time.

# Interface Syntax & Example

```
/*  
interface <interface name>{  
    Constant/final variables;  
    Meaningless methods - no body methods  
}  
*/
```

```
interface Playable {  
    void play();  
}
```

```
public class Piano implements Playable{  
    public void play() {  
        System.out.println("Piano play by pressing keys");  
    }  
}
```

# Interfaces & Multiple Inheritance

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

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

# Abstract Class vs Interface

Feature	Class	Interface
Instantiation	Objects can be created using new.	Cannot create objects (acts as a contract/blueprint).
Variables	Can have instance variables.	All variables are public static final (constants).
Methods	Can have concrete methods (with body).	Methods are abstract by default (Java 8+ allows default & static).
Inheritance	Supports single inheritance (extends one class).	Supports multiple inheritance (a class can implement many interfaces).
Constructors	Can have constructors.	No constructors allowed.
Access Modifiers	Supports private, protected, public, default.	Members are public by default.
Keyword	Declared using class.	Declared using interface.

**Thank you!**