Interface

Monday, September 15, 2025 11:35 AM

Interface in Java – Notes

1. Definition

An **interface** in Java is a collection of **abstract methods** (and from Java 8 onward, also default and static methods) used to define a contract that classes must follow.

- It specifies **what** a class must do but **not how** to do it.
- A class implements an interface to provide the actual behavior.

2. Key Features

- Declared using the interface keyword.
- All methods in an interface are **public and abstract** by default (till Java 7).
- Variables in an interface are **public, static, and final** by default.
- A class can **implement multiple interfaces** (Java's way of multiple inheritance).
- Interfaces cannot be instantiated directly.

3. Key Features

```
interface EvCar extends Car{  2 implementations  new *
    // constants
    String modelType = "ev" ;

    // abstract methods
    void autopark();  2 implementations  new *
    void autopilot();  2 implementations  new *
}
```

4. Implementing an interface

```
class BMWiX1 implements EvCar , SportsCar{  new *

    // declaration of abstract methods
    @Override  new *
    public void autopark() {
        System.out.println("car in auto park mode !!");
    }

    @Override  new *
    public void autopilot() {
        System.out.println("car in autopilot mode !!");
    }

    @Override  new *
    public void sports() {
```

```
    @Override  new *
    public void sports() {
        System.out.println("car in offroad mode !!");
    }
}
```

5. When to Use Interfaces

- To achieve **multiple inheritance** of type.
- To define a **contract** or **API** for a group of classes.
- To implement **loose coupling** between components.

6. Difference Between Abstract Class & Interface

Feature	Abstract Class	Interface
Keyword	abstract class	interface
Methods	Can have abstract + concrete	Abstract by default (till Java 7)
Variables	Any type	public static final only
Inheritance	Single (extends)	Multiple (implements)
Constructors	Can have	Cannot have

7. Anonymous Inner Class & Interface

You can create an **anonymous inner class** implementing an interface without creating a separate named class:

```
// anonymous inner class object creation through interface
EvCar myCar2 = new EvCar() {  new *
    @Override  new *
    public void autopark() {
        System.out.println("car in auto park mode !!");
    }

    @Override  new *
    public void autopilot() {
        System.out.println("car in autopilot mode !!");
    }
};
```

8. Using it store diff. class objects

```
Car[] parkingArea = new Car[2] ;
parkingArea[0] = mycar1 ;
parkingArea[1] = myCar2 ;
```

9. Java Versions – Interface Enhancements

- **Java 8**: Default & static methods allowed.

- **Java 9**: Private methods allowed inside interfaces.

10. Important Points

- Interfaces support **polymorphism**.
- A class can implement multiple interfaces but can extend only one class.
- All interface methods are implicitly **public**.
- Cannot create object of an interface directly.
- After implementation we only get methods not the variables cause they are final & static not instance variables