# Interface in Java - Complete Notes (Interview Ready)

## 1. Defining an Interface

- We create an interface using the interface keyword.
- Interfaces can extend other interfaces.
- Interfaces cannot be declared as final because they are meant to be implemented.

Example:

```
interface I1 {
    void method1(); // abstract method
}
interface I2 extends I1 {
    void method2();
}
```

## 2. Interface Variables

- All variables in an interface are implicitly public static final.
- You cannot make them private or protected.
- They must be initialized at the time of declaration.

All the following are same:

```
interface I1 {
    int a = 10;                    // public static final
    public static final int b = 20;
    static final int c = 30;
    final int d = 40;
}
```

Variables are accessed using the interface name:

```
System.out.println(I1.a);
```

3. Interface Methods

a) Abstract Methods

- Methods without a body.
- By default, they are public and abstract.

```
void display(); // abstract method
```

b) Non-abstract Methods (allowed in 3 ways)

1. Static methods
   - Must be called using the interface name.
   - Not inherited by implementing classes.

```
static void show() {
    System.out.println("Static method");
}
```

2. Default methods
   - Introduced in Java 8.
   - Can be overridden in child classes.

```
default void print() {
    System.out.println("Default method");
}
```

3. Private methods (Java 9+)

   - Can only be used inside the interface.

   - Usually called from default methods.

```
private void helper() {
    System.out.println("Private method");
}

default void callHelper() {
    helper(); // allowed
}
```

4. Important Points

- Static methods are not inherited by child classes. We call them using the interface name.
- Default methods can be called from the child class (or overridden).
- Private methods cannot be accessed outside the interface.
   - If you want to access private methods, you must call them from a default method (not from a static method).

5. Interfaces and Objects

- We cannot create objects of an interface.
- But we can use an interface as a reference to a child class object.

```
class Test implements I1 {
    public void method1() {
        System.out.println("Implemented method");
    }
}

public class Main {
    public static void main(String[] args) {
        I1 obj = new Test(); // interface reference
        obj.method1();
    }
}
```

6. Summary for Interview

- Variables -> public static final by default.
- Methods -> abstract by default, but we can have static, default, and private methods too.
- Interfaces can extend multiple interfaces.
- Cannot be declared final or instantiated.
- Interface references can hold objects of implementing classes.

This is enough for most interviews, but adding 1-2 working code examples will make your answer stand out.