Anu Arora, Assistant Professor, CSE , GNA University

INTERFACES IN JAVA

Introduction

- An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.
- The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.
- In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Need for Interface in Java

We need an Interface in Java for the following reasons:

- Total Abstraction
- Multiple Inheritance
- Loose-Coupling

Total Abstraction

- Abstraction is the critical concept of Object-Oriented programming techniques.
- An interface only stores the method signature and not the method definition.
- Method Signatures make an Interface achieve complete Abstraction by hiding the method implementation from the user.

Multiple Inheritance

- Without Interface, the process of multiple inheritances is impossible as the conventional way of inheriting multiple parent classes results in profound ambiguity.
- This type of ambiguity is known as the Diamond problem.
- Interface resolves this issue.

Loose-Coupling

- The term Coupling describes the dependency of one class for the other.
- So, while using an interface, we define the method separately and the signature separately.
- This way, all the methods, and classes are entirely independent and archives Loose Coupling.


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

Similarities with Classes

An interface is similar to a class in the following ways –

- An interface can contain any number of methods.
- An interface is written in a file with a **.java** extension, with the name of the interface matching the name of the file.
- The byte code of an interface appears in a **.class** file.
- Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

Dissimilarities with Classes

However, an interface is different from a class in several ways, including −

- You cannot instantiate an interface.
- An interface does not contain any constructors.
- All of the methods in an interface are abstract.
- An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
- An interface is not extended by a class; it is implemented by a class.
- An interface can extend multiple interfaces.

Declaring Interfaces

- The **interface** keyword is used to declare an interface. Here is a simple example to declare an interface −

```
public interface NameOfInterface {
// Any number of final, static fields
// Any number of abstract method declarations
 }
```

Syntax of an Interface

Syntax of an Interface in Java is written as shown below.

```
Interface <Interface Name> {
//Declare Constant Fields;
//Declare Methods;
//Default Methods;
 }
//Example through NETBEANS
```

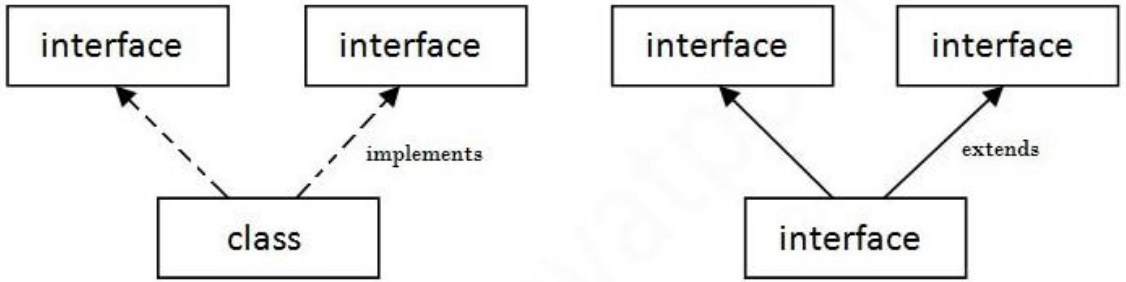
Properties of Interfaces:

Interfaces have the following properties –

- An interface is implicitly abstract. You do not need to use the **abstract** keyword while declaring an interface.
- Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.
- Methods in an interface are implicitly public.

Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.



Multiple Inheritance in Java

```
interface Printable{
void print();
}
interface Showable{
void show();
}
class A7 implements Printa
    ble,Showable{
public void print(){System.out.println
    ("Hello");}
public void show(){System.out.println
    ("Welcome");}
public static void main(String args[])
    {
A7 obj = new A7();
obj.print();
obj.show();   }}
```

Rules to override methods

When overriding methods defined in interfaces, there are several rules to be followed −

- Checked exceptions should not be declared on implementation methods other than the ones declared by the interface method or subclasses of those declared by the interface method.
- The signature of the interface method and the same return type or subtype should be maintained when overriding the methods.
- An implementation class itself can be abstract and if so, interface methods need not be implemented.

Rules to implement Interfaces

When implementation interfaces, there are several rules –

- A class can implement more than one interface at a time.
- A class can extend only one class, but implement many interfaces.
- An interface can extend another interface, in a similar way as a class can extend another class.

Extending Interfaces

- An interface can extend another interface in the same way that a class can extend another class.
- The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

Default Methods in Java Interfaces

- With the release of Java 8, we can now add methods with implementation inside an interface. These methods are called default methods.
- To declare default methods inside interfaces, we use the default keyword. For example,

```
public default void getSides() {
// body of getSides()
}
```

- E.Balaguruswamy, Programming with JAVA, A primer, 3e, TATA McGraw-Hill Company.
- https://www.tutorialspoint.com/java/java_interfaces.htm
- https://www.javatpoint.com/interface-in-java
- https://www.programiz.com/java-programming/interfaces
- https://www.simplilearn.com/tutorials/java-tutorial/java-interface