Interface

Interface in Java

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.

Why use Java interface?

There are mainly three reasons to use interface. They are given below.

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

How to declare an interface?

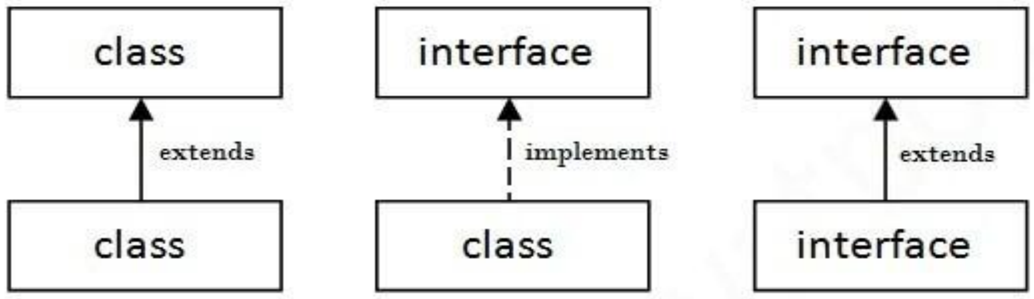
An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and **all the fields are public, static and final by default**. A class that implements an interface must implement all the methods declared in the interface.

Syntax:

```
interface <interface_name>
{
   // declare constant fields
   // declare methods that abstract
   // by default.
}
```

The relationship between classes and interfaces

As shown in the figure given below, a class extends another class, an interface extends another interface, but a **class implements an interface**.



Java Interface Example

In this example, the Printable interface has only one method, and its implementation is provided in the Example

Example

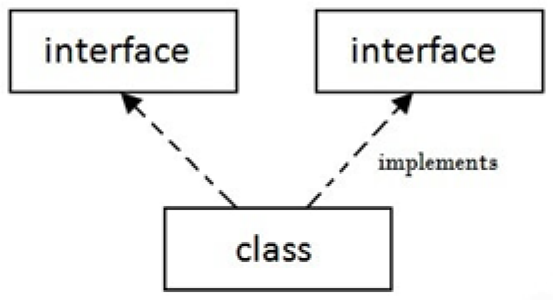
```
/*  Example :  Java Interface*/
interface printable
{
        void print();
}
class TestInterface implements printable
{
        public void print()
        {
                System.out.println("Hello");
        }
        public static void main(String args[])
        {
                TestInterfaceobj = new TestInterface();
                obj.print();
        }
}
```

Example:

```
//Interface declaration: by first user
interface Drawable
{
        void draw();
}
//Implementation: by second user
class Rectangle implements Drawable
{
        public void draw()
        {
                System.out.println("drawing rectangle");
        }
}
class Circle implements Drawable
{
        public void draw()
        {
                System.out.println("drawing circle");
        }
}
//Using interface: by third user
class TestInterface1
{
        public static void main(String args[])
        {
                Drawable d=new Circle();
                d.draw();
        }
}
```

Multiple inheritance in Java by interface

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



```
/*  Example :  Java Interface to achieve multiple inheritance*/
interface Printable
{
        void print();
}
interface Showable
{
        void show();
}
class TestInterface2 implements Printable, Showable
```

```
{
	public void print()
	{
		System.out.println("Hello");
	}
	public void show()
	{
		System.out.println("Welcome");
	}
	public static void main(String args[])
	{
		TestInterface2 obj = new TestInterface2();
		obj.print();
		obj.show();
	}
}
```

```
/*  Example :  Java Interface to achieve multiple inheritance*/
interface Printable
{
	void print();
}
interface Showable
{
	void print();
}
class TestInterface3 implements Printable, Showable
{
	public void print()
	{
		System.out.println("Hello");
	}
	public static void main(String args[])
	{
		TestInterface3 obj = new TestInterface3();
		obj.print();
	}
}
```

```
/*  Example :  Java Interface inheritance*/
interface Printable
{
	void print();
}
interface Showable extends Printable
{
	void show();
}
```

```
class TestInterface4 implements Showable
{
	public void print()
	{
		System.out.println("Hello");
	}
	public void show()
	{
		System.out.println("Welcome");
	}
	public static void main(String args[])
	{
		TestInterface4 obj = new TestInterface4();
		obj.print();
		obj.show();
	}
}
```