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.

Java Interface also represents the IS-A relationship.

It cannot be instantiated just like the abstract class.

Uses of 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.

Declaring an interface

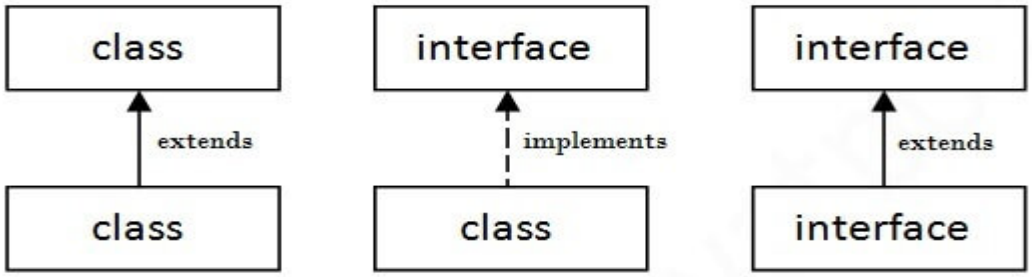
An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the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

Syntax:

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

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.



Example

```
interface A
{
   void display();
}
class B implements A
{
    public void display()
    {
        System.out.println("Hello");
     }
}
class MB
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.display();
 }
}
```

Output:

Hello

Interface Example:

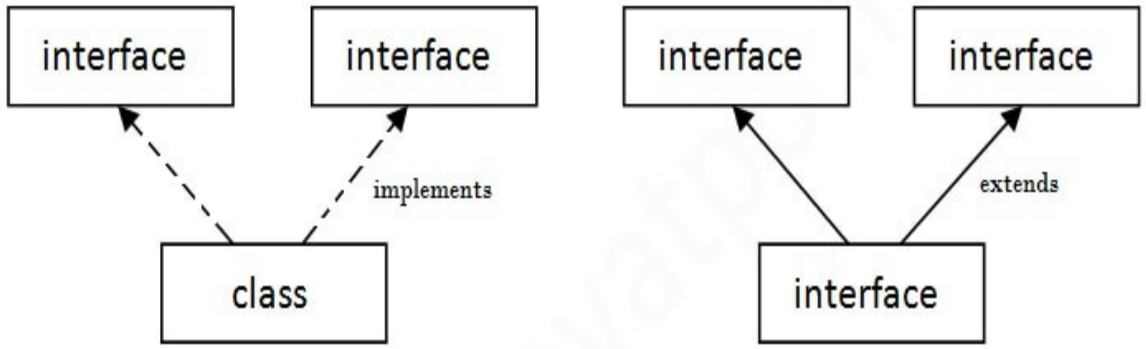
In this example, the interface A has only one method. Its implementation is provided by B and C classes. In a real scenario, an interface is defined by someone else, but its implementation is provided by different implementation providers. Moreover, it is used by someone else. The implementation part is hidden by the user who uses the interface.

```
interface A
{
   void display();
}
class B implements A
{
public void display()
{
   System.out.println("Display method in B class");
}
}
class C implements B
{
   public void display()
   {
      System.out.println("display method in C class");
    }
}
class MainClass
{
    public static void main(String args[])
    {
```

```
        D obj=new D();
    obj.draw();
}
}
```

Multiple inheritance 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 A
{
     void display();
}
interface B
{
   void show();
}
class C implements A,B
{
    public void display()
```

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

Output:

Hello

Welcome

Interface inheritance

A class implements an interface, but one interface extends another interface.

```
interface A
{
      void display();
}
interface B extends A
```

```
{
    void show();
}
class C implements B
{
    public void display()
    {
        System.out.println("Hello");
     }
     public void show()
     {
         System.out.println("Welcome");
      }
}
Class MainClass
 {
     public static void main(String args[])
     {
          C obj = new C();
          obj.display();
          obj.show();
      }
}
```

Output:

Hello

Welcome

Abstract class in Java

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

Example

```
abstract class

{  }
```

Abstract Method

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example

```
abstract void display();   //no method body and abstract
```

Example of Abstract class that has an abstract method

In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

```
abstract class Bike

{

  abstract void run();

}

class Honda extends Bike

{

    void run()

     {
```

```
        System.out.println("running safely");
    }
}
class Car
{
    public static void main(String args[])
    {
        Bike obj = new Honda();
        obj.run();
    }
}
running safely
```

Difference between abstract class and interface

Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods. Abstract class and interface both can't be instantiated.

But there are many differences between abstract class and interface that are given below.

Abstract class	Interface
1) Abstract class can have abstract and non-abstract methods.	Interface can have only abstract methods. Since Java 8, it can have default and static methods also.
2) Abstract class doesn't support multiple inheritance.	Interface supports multiple inheritance.
3) Abstract class can have final, non-final, static and non-static variables.	Interface has only static and final variables.
4) Abstract class can provide the implementation of interface.	Interface can't provide the implementation of abstract class.
5) The abstract keyword is used to declare	The interface keyword is used to declare interface.

abstract class.	
6) An abstract class can extend another Java class and implement multiple Java interfaces.	An interface can extend another Java interface only.
7) An abstract class can be extended using keyword "extends".	An interface can be implemented using keyword "implements".
8) A Java abstract class can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)Example:
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}