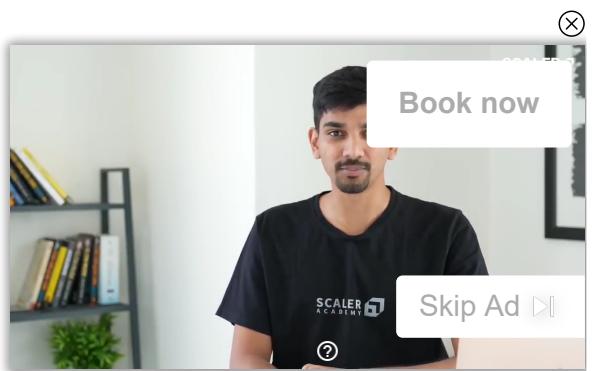


[Home](#)[Java](#)[Programs](#)[OOPs](#)[String](#)[Exception](#)[Multithreading](#)[Collections](#)[↑ SCROLL TO TOP](#)

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.

Since Java 8, we can have **default and static methods** in an interface.

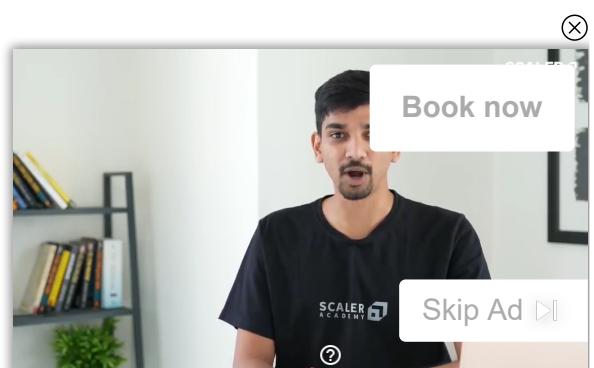
Since Java 9, we can have **private methods** in an interface.

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.

↑ SCROLL TO TOP





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.
}
```

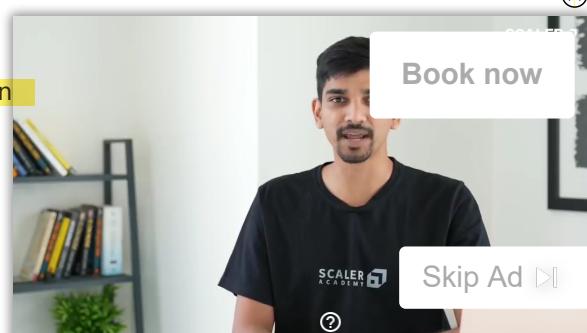
Java 8 Interface Improvement

Since **Java 8**, interface can have default and static methods which is discussed later.

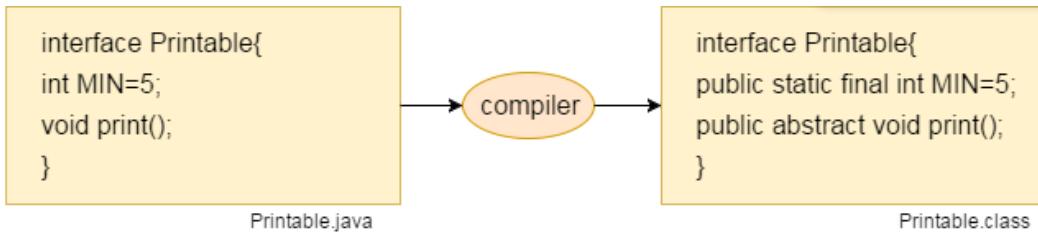
Internal addition by the compiler

The Java compiler adds `public` and `abstract` keywords before the interface method. Moreover, it adds `public`, `static` and `final` keywords before data members.

In other words, **Interface fields are public, static and final by default, and**



↑ SCROLL TO TOP



Cymulate

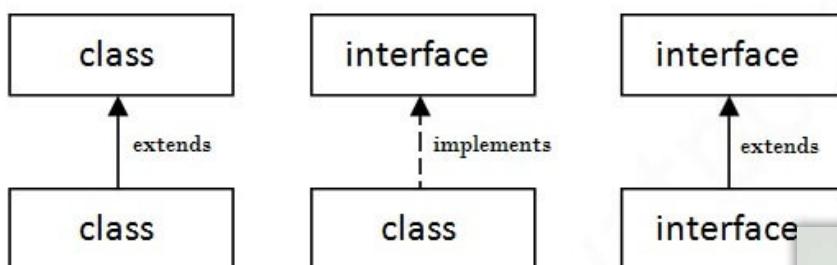


The Only Extended
Security Posture
Management Platform.

[C for yourself → Live Dem](#)

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**.



↑ SCROLL TO TOP [Example](#)

In this example, the Printable interface has only one method, and its implementation is provided in the A6 class.

```
interface printable{
void print();
}

class A6 implements printable{
public void print(){System.out.println("Hello");}
}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
}
}
```

[Test it Now](#)

Output:

Hello

Java Interface Example: Drawable

In this example, the Drawable interface has only one method. Its implementation is provided by Rectangle and Circle 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.

File: TestInterface1.java

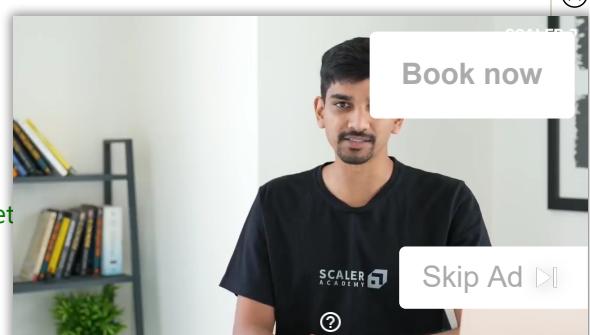
```
//Interface declaration: by first user
interface Drawable{
void draw();
}

//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(); //In real scenario, object is provided by met
d.draw();
}
}
```

[SCROLL TO TOP](#)



[Test it Now](#)

Output:

```
drawing circle
```

Java Interface Example: Bank

Let's see another example of java interface which provides the implementation of Bank interface.

File: TestInterface2.java

```
interface Bank{
    float rateOfInterest();
}

class SBI implements Bank{
    public float rateOfInterest(){return 9.15f;}
}

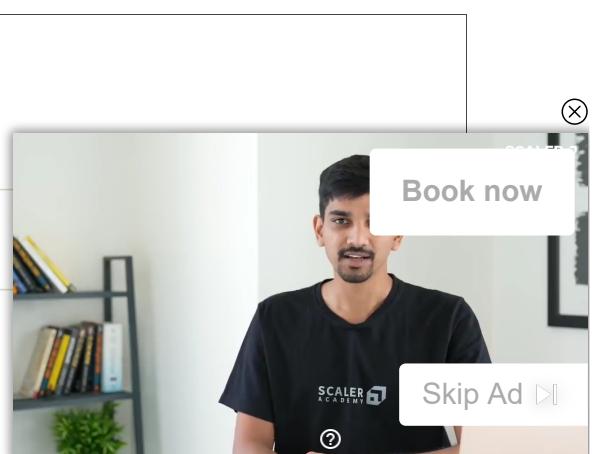
class PNB implements Bank{
    public float rateOfInterest(){return 9.7f;}
}

class TestInterface2{
    public static void main(String[] args){
        Bank b=new SBI();
        System.out.println("ROI: "+b.rateOfInterest());
    }
}
```

[Test it Now](#)

Output:

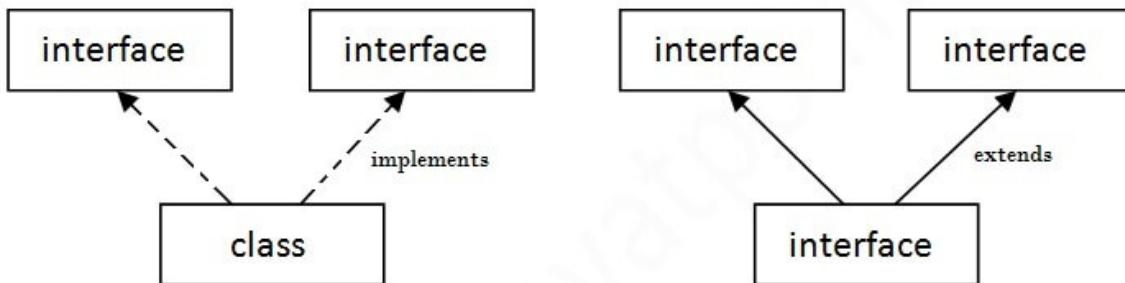
```
ROI: 9.15
```



Multiple inheritance in Java by interface

[↑ SCROLL TO TOP](#)

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 Printable,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();
}
  
```

Test it Now

Output:Hello
Welcome

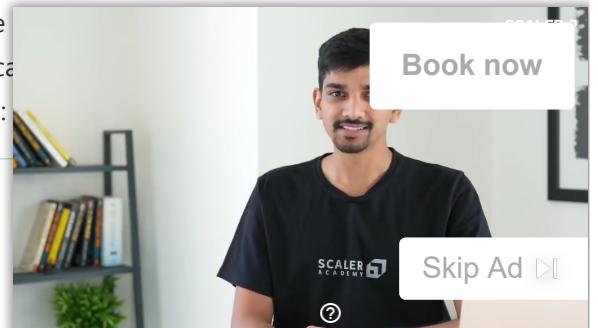
Q) Multiple inheritance is not supported through class in java, but it is possible by an interface, why?

As we have explained in the inheritance chapter, multiple inheritance leads to ambiguity. However, it is supported in case of an interface because implementation is provided by the implementation class. For example:

```

interface Printable{
    ...
}
  
```

↑ SCROLL TO TOP



```

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();
    }
}

```

[Test it Now](#)

Output:

```
Hello
```

As you can see in the above example, Printable and Showable interface have same methods but its implementation is provided by class TestInterface1, so there is no ambiguity.

Interface inheritance

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

```

interface Printable{
    void print();
}

interface Showable extends Printable{
    void show();
}

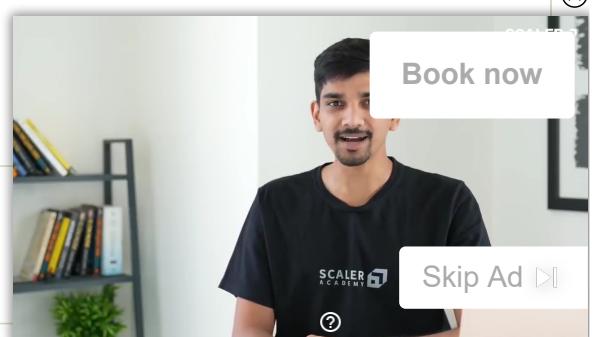
class TestInterface4 implements Showable{
    public void print(){System.out.println("Hello");}
    public void show(){System.out.println("Welcome");}
}

public static void main(String args[]){
    TestInterface4 obj = new TestInterface4();
    obj.print();
    obj.show();
}

```

[Test it Now](#)

[↑ SCROLL TO TOP](#)



Hello
Welcome

Java 8 Default Method in Interface

Since Java 8, we can have method body in interface. But we need to make it default method. Let's see an example:

File: TestInterfaceDefault.java

```
interface Drawable{
    void draw();
    default void msg(){System.out.println("default method");}
}

class Rectangle implements Drawable{
    public void draw(){System.out.println("drawing rectangle");}
}

class TestInterfaceDefault{
    public static void main(String args[]){
        Drawable d=new Rectangle();
        d.draw();
        d.msg();
    }
}
```

Test it Now

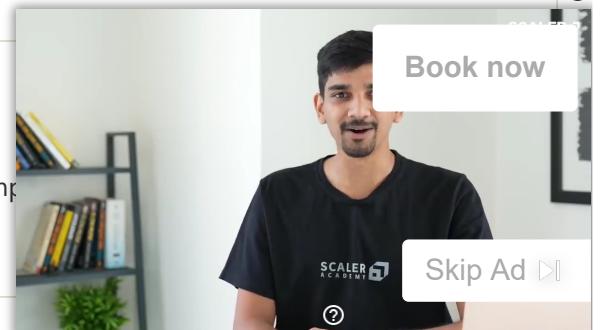
Output:

```
drawing rectangle
default method
```

Java 8 Static Method in Interface

Since Java 8, we can have static method in interface. Let's see an example:

SCROLL TO TOP



```

interface Drawable{
    void draw();
    static int cube(int x){return x*x*x;}
}

class Rectangle implements Drawable{
    public void draw(){System.out.println("drawing rectangle");}
}

class TestInterfaceStatic{
    public static void main(String args[]){
        Drawable d=new Rectangle();
        d.draw();
        System.out.println(Drawable(cube(3)));
    }
}

```

[Test it Now](#)

Output:

```

drawing rectangle
27

```

Q) What is marker or tagged interface?

An interface which has no member is known as a marker or tagged interface, for example, **Serializable**, **Cloneable**, **Remote**, etc. They are used to provide some essential information to the JVM so that JVM may perform some useful operation.

```

//How Serializable interface is written?
public interface Serializable{
}

```

Nested Interface in Java

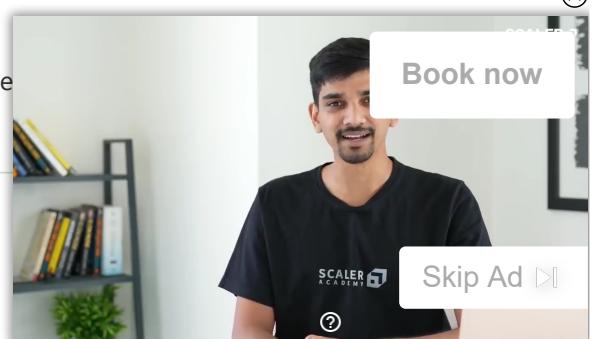
Note: An interface can have another interface which is known as a nested classes chapter. For example:

```

interface printable{
    ...
}

interface .Printable{
}

```



```
void msg();
}
}
```

More about Nested Interface

« Prev

Next »

 For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



ⓘ ×

PropertyShare.in

India's 1st & Largest CRE Platform. Tax Efficient, Safe, Free, Optio

Property Share

↑ SCROLL TO TOP

Learn Latest Tutorials

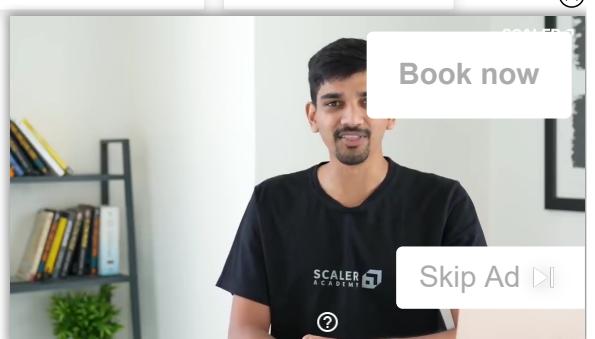
 Splunk tutorial Splunk	 SPSS tutorial SPSS	 Swagger tutorial Swagger	 T-SQL tutorial Transact-SQL	 Tumblr tutorial Tumblr
 React tutorial ReactJS	 Regex tutorial Regex	 Reinforcement learning tutorial Reinforcement Learning	 R Programming tutorial R Programming	 RxJS tutorial RxJS
 React Native tutorial React Native	 Python Design Patterns Python Design Patterns	 Python Pillow tutorial Python Pillow	 Python Turtle tutorial Python Turtle	 Keras tutorial Keras

Preparation

 Aptitude Aptitude	 Logical Reasoning Reasoning	 Verbal Ability Verbal Ability	 Interview Questions Interview Questions	 Company Interview Questions Company Questions
---	---	---	---	---

Trending Technologies

 Artificial Intelligence Tutorial Artificial Intelligence	 AWS Tutorial AWS	 Selenium tutorial Selenium	 Cloud Computing tutorial Cloud Computing	 Hadoop tutorial Hadoop
 ReactJS Tutorial ReactJS	 Data Science Tutorial Data Science	 Angular 7 Tutorial Angular 7	 Blockchain Tutorial Blockchain	 Git Tutorial Git
 Machine Learning Tutorial Machine Learning	 DevOps Tutorial DevOps			

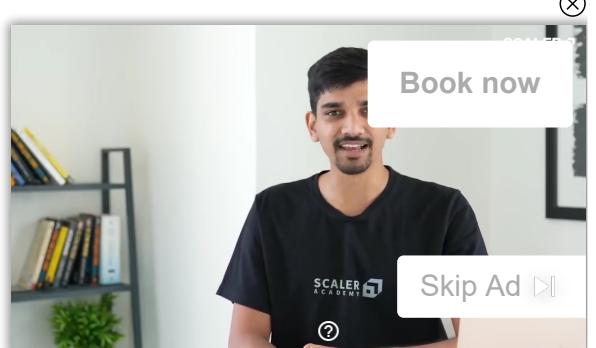


↑ SCROLL TO TOP

B.Tech / MCA

 DBMS tutorial DBMS	 Data Structures tutorial Data Structures	 DAA tutorial DAA	 Operating System tutorial Operating System	 Computer Network tutorial Computer Network
 Compiler Design tutorial Compiler Design	 Computer Organization and Architecture Computer Organization	 Discrete Mathematics Tutorial Discrete Mathematics	 Ethical Hacking Tutorial Ethical Hacking	 Computer Graphics Tutorial Computer Graphics
 Software Engineering Tutorial Software Engineering	 html tutorial Web Technology	 Cyber Security tutorial Cyber Security	 Automata Tutorial Automata	 C Language tutorial C Programming
 C++ tutorial C++	 Java tutorial Java	 .Net Framework tutorial .Net	 Python tutorial Python	 List of Programs Programs
 Control Systems tutorial Control System	 Data Mining Tutorial Data Mining	 Data Warehouse Tutorial Data Warehouse		

↑ SCROLL TO TOP



↑ SCROLL TO TOP

