# Interface in Java

1. **Is-A relationship:** Whenever one class inherits another class, it is called an IS-A relationship.
2. **Has-A relationship:**Whenever an instance of one class is used in another class, it is called HAS-A relationship.

The interface in Java is a mechanism to achieve [*abstraction*](https://www.javatpoint.com/abstract-class-in-java). Interfaces can have abstract methods and variables. It cannot have a method body. It is used to achieve abstraction and multiple [inheritance in Java](https://www.javatpoint.com/inheritance-in-java).

Java Interface also **represents the IS-A relationship.**

It cannot be instantiated just like the abstract class.

## **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.( Loose coupling in Java means that the classes are independent of each other)

## **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.

1. **interface** <interface\_name>{
3. // declare constant fields
4. // declare methods that abstract
5. // by default.
6. }

## **Java 8 Interface Improvement**

Since [Java 8](https://www.javatpoint.com/java-8-features), interface can have default and static methods which is discussed later.

AD**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 the methods are public and abstract.



#### **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 A6 class.

1. **interface** printable{
2. **void** print();
3. }
4. **class** A6 **implements** printable{
5. **public** **void** print(){System.out.println("Hello");}
7. **public** **static** **void** main(String args[]){
8. A6 obj = **new** A6();
9. obj.print();
10. }
11. }

## **Java Interface Example: Bank**

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

1. **interface** Bank{
2. **float** rateOfInterest();
3. }
4. **class** SBI **implements** Bank{
5. **public** **float** rateOfInterest(){**return** 9.15f;}
6. }
7. **class** PNB **implements** Bank{
8. **public** **float** rateOfInterest(){**return** 9.7f;}
9. }
10. **class** TestInterface2{
11. **public** **static** **void** main(String[] args){
12. Bank b=**new** SBI();
13. System.out.println("ROI: "+b.rateOfInterest());
14. }}

## **Multiple inheritance in Java by interface**

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



1. **interface** Printable{
2. **void** print();
3. }
4. **interface** Showable{
5. **void** show();
6. }
7. **class** A7 **implements** Printable,Showable{
8. **public** **void** print(){System.out.println("Hello");}
9. **public** **void** show(){System.out.println("Welcome");}
11. **public** **static** **void** main(String args[]){
12. A7 obj = **new** A7();
13. obj.print();
14. obj.show();
15. }
16. }

## **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 is not supported in the case of [class](https://www.javatpoint.com/object-and-class-in-java) because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class. For example:

1. **interface** Printable{
2. **void** print();
3. }
4. **interface** Showable{
5. **void** print();
6. }
8. **class** TestInterface3 **implements** Printable, Showable{
9. **public** **void** print(){System.out.println("Hello");}
10. **public** **static** **void** main(String args[]){
11. TestInterface3 obj = **new** TestInterface3();
12. obj.print();
13. }
14. }

As you can see in the above example, Printable and Showable interface have same methods but its implementation is provided by class TestTnterface1, so there is no ambiguity.

## **Interface inheritance**

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

1. **interface** Printable{
2. **void** print();
3. }
4. **interface** Showable **extends** Printable{
5. **void** show();
6. }
7. **class** TestInterface4 **implements** Showable{
8. **public** **void** print(){System.out.println("Hello");}
9. **public** **void** show(){System.out.println("Welcome");}
11. **public** **static** **void** main(String args[]){
12. TestInterface4 obj = **new** TestInterface4();
13. obj.print();
14. obj.show();
15. }
16. }

## **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*

1. **interface** Drawable{
2. **void** draw();
3. **default** **void** msg(){System.out.println("default method");}
4. }
5. **class** Rectangle **implements** Drawable{
6. **public** **void** draw(){System.out.println("drawing rectangle");}
7. }
8. **class** TestInterfaceDefault{
9. **public** **static** **void** main(String args[]){
10. Drawable d=**new** Rectangle();
11. d.draw();
12. d.msg();
13. }}

## **Java 8 Static Method in Interface**

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

*File: TestInterfaceStatic.java*

1. **interface** Drawable{
2. **void** draw();
3. **static** **int** cube(**int** x){**return** x\*x\*x;}
4. }
5. **class** Rectangle **implements** Drawable{
6. **public** **void** draw(){System.out.println("drawing rectangle");}
7. }
9. **class** TestInterfaceStatic{
10. **public** **static** **void** main(String args[]){
11. Drawable d=**new** Rectangle();
12. d.draw();
13. System.out.println(Drawable.cube(3));
14. }}

Note: An interface can have another interface which is known as a nested interface. We will learn it in detail in the [nested classes](https://www.javatpoint.com/java-inner-class) chapter.