Interface in Java

1. [Interface](http://www.javatpoint.com/interface-in-java)
2. [Example of Interface](http://www.javatpoint.com/interface-in-java#interfaceex)
3. [Multiple inheritance by Interface](http://www.javatpoint.com/interface-in-java#interfacemultiple)
4. [Why multiple inheritance is supported in Interface while it is not supported in case of class.](http://www.javatpoint.com/interface-in-java#interfacewhynot)
5. [Marker Interface](http://www.javatpoint.com/interface-in-java#interfacemarker)
6. [Nested Interface](http://www.javatpoint.com/nested-interface)

An **interface** is a blueprint of a class. It has static constants and abstract methods.

The interface is **a mechanism to achieve fully abstraction** in java. There can be only abstract methods in the interface. It is used to achieve fully abstraction and multiple inheritance in Java.

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

It cannot be instantiated just like abstract class.

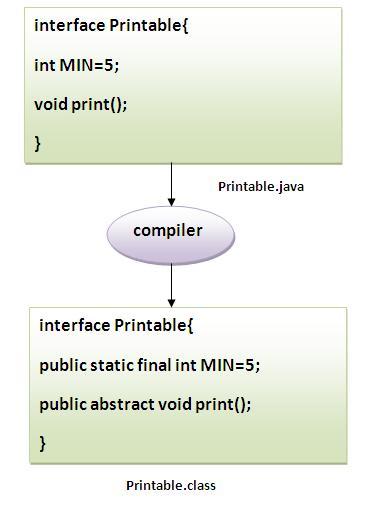
Why use Interface?

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

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

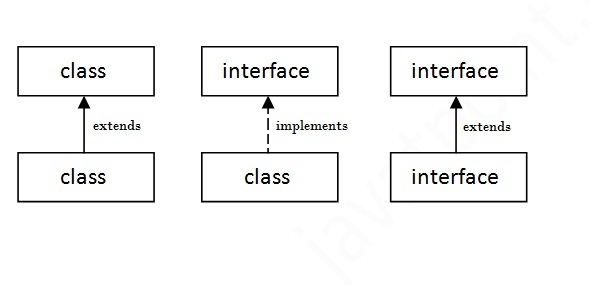
***The java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members.***

In other words, Interface fields are public, static and final bydefault, and methods are public and abstract.



**Understanding 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**.



Simple example of Interface

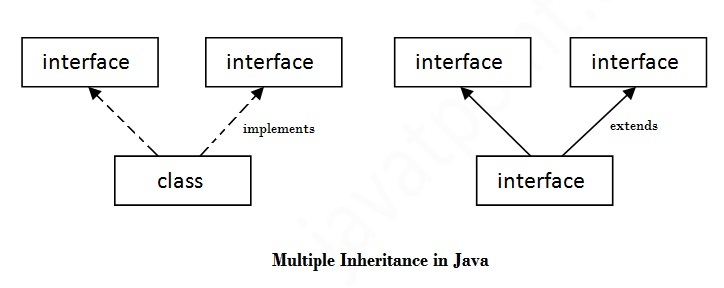
|  |
| --- |
| In this example, Printable interface have only one method, its implementation is provided in the A class. |

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

Output:Hello

Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance.



1. **interface** Printable{
2. **void** print();
3. }
5. **interface** Showable{
6. **void** show();
7. }
9. **class** A **implements** Printable,Showable{
11. **public** **void** print(){System.out.println("Hello");}
12. **public** **void** show(){System.out.println("Welcome");}
14. **public** **static** **void** main(String args[]){
15. A obj = **new** A();
16. obj.print();
17. obj.show();
18. }
19. }

Output:Hello

Welcome

**Q) Multiple inheritance is not supported in case of class but it is supported in case of interface, why?**

|  |
| --- |
| As we have explained in the inheritance chapter, multiple inheritance is not supported in case of class. But it is supported in case of interface because there is no ambiguity as implementation is provided by the implementation class. For example: |

1. **interface** Printable{
2. **void** print();
3. }
5. **interface** Showable{
6. **void** print();
7. }
9. **class** A **implements** Printable,Showable{
11. **public** **void** print(){System.out.println("Hello");}
13. **public** **static** **void** main(String args[]){
14. A obj = **new** A();
15. obj.print();
16. }
17. }

Output:Hello

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

***Note: A class implements interface but One interface extends another interface .***

1. **interface** Printable{
2. **void** print();
3. }
5. **interface** Showable **extends** Printable{
6. **void** show();
7. }
9. **class** A **implements** Showable{
11. **public** **void** print(){System.out.println("Hello");}
12. **public** **void** show(){System.out.println("Welcome");}
14. **public** **static** **void** main(String args[]){
15. A obj = **new** A();
16. obj.print();
17. obj.show();
18. }
19. }

Output:Hello

Welcome

**Que) What is marker or tagged interface ?**

|  |
| --- |
| An interface that have no member is known as 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. |

1. //How Serializable interface is written?
3. **public** **interface** Serializable{
4. }

**Nested Interface**

Note: An interface can have another interface i.e. known as nested interface. We will learn it in detail in the nested classes chapter. For example:

1. **interface** printable{
2. **void** print();
3. **interface** MessagePrintable{
4. **void** msg();
5. }
6. }

[**More about Nested Interface**](http://www.javatpoint.com/nested-interface)