## INTERFACE IN JAVA

- An **interface** is a blueprint of a class.
- The interface is **a mechanism to achieve full abstraction** in Java.
- There can be only abstract methods in the interface.
- It is used to achieve full abstraction and multiple inheritances in Java.
- It cannot be instantiated just like an abstract class.

## WHY USE INTERFACE?

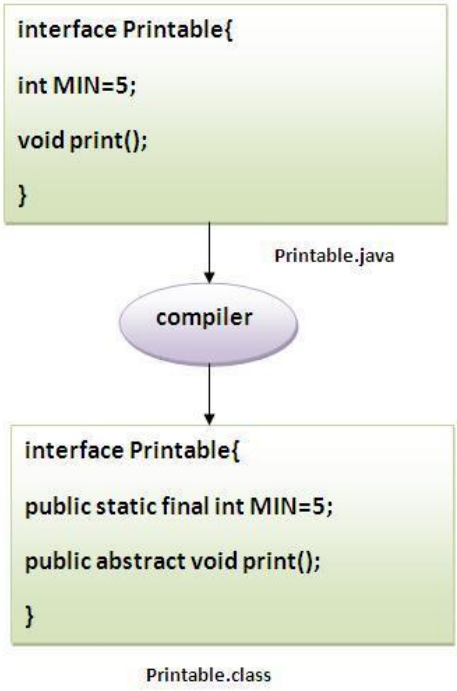
There are the following reasons to use an interface. They are given below.

- It is used to achieve full (100%) abstraction.
- By interface, we can support the functionality of multiple inheritances.

### Interface fields(data members) and Methods( member functions):

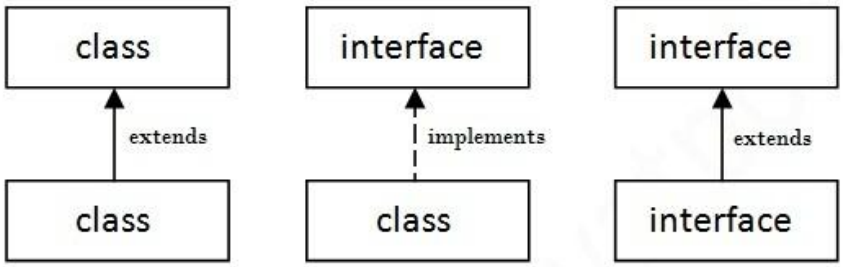
In other words, Interface fields are public, static, and final by default, and methods are public and abstract.

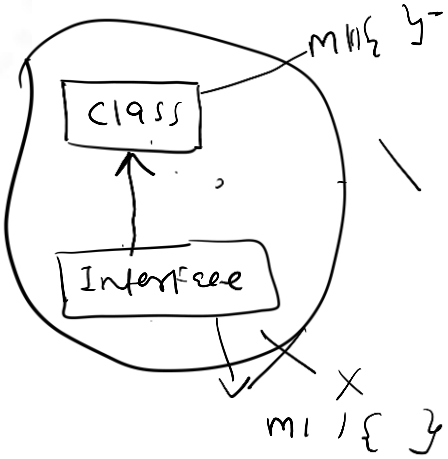
The java compiler adds public, static, and final keywords before data members and adds public and abstract keywords before the interface method.




**Understanding relationship between classes and interfaces**

- A class extends another class.
- An interface extends another interface
- but a **class implements an interface**.



**extends Vs implements**

A class can extend only one class at a time.

class A extends B ✓

class A extends B, C ☒

A class can implement any number of interfaces at a time.

class A implements B, C ✓

A class can extend a class and implement any number of interfaces simultaneously.

class A extends B implements C,D ✓

class A implements B,C extends D ☒

An interface can extend any number of interfaces at a time.

interface A extends B,C ✓

Which of the following is true??

1. A class can extend any number of class simultaneously.
2. A class implement only one interface at a time.
3. A class extend a class OR implement an interface but not both simultaneously.
4. An interface can implement any number of classes.
5. An interface can extend only one interface at a time.
6. None of the above.

**Consider the expressions:**

**X extends Y**

Which of the following properties is true?

1. Both should be classes
2. Both should be interfaces
3. No restriction

4. Both can be either classes or interfaces.

**X extends Y,Z**

1. X,Y,Z should be classes.
2. X should be class and Y,Z should be interfaces.
3. X,Y,Z should be interfaces.

**X extends Y implements Z**

X,Y should be classes, Z should be interface ✓

**X implements Y extends Z** -Compile Time Error

**Example: A class implements one interface**

```
interface printable{
void print();  //by default public and abstract
}

class A implements printable{
public void print()//overridden method
{
System.out.println("Hello");
}

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

**OUTPUT:HELLO**

**EXAMPLE: TWO CLASSES IMPLEMENT ONE INTERFACE**

```
interface Drawable {
void draw();
}
class Rectangle implements Drawable{
public void draw()//overridden
{
System.out.println("drawing rectangle");
}
}
class Circle implements Drawable
{
public void draw()//overridden
{
System.out.println("drawing circle");
}
}
class TestInterface
{
public static void main(String args[])
{
Drawable d=new Circle();
Drawable e =new Rectangle();
d.draw();
```

```
24. e.draw();
25. }
26. }
```

**Output:**
drawing circle
drawing rectangle

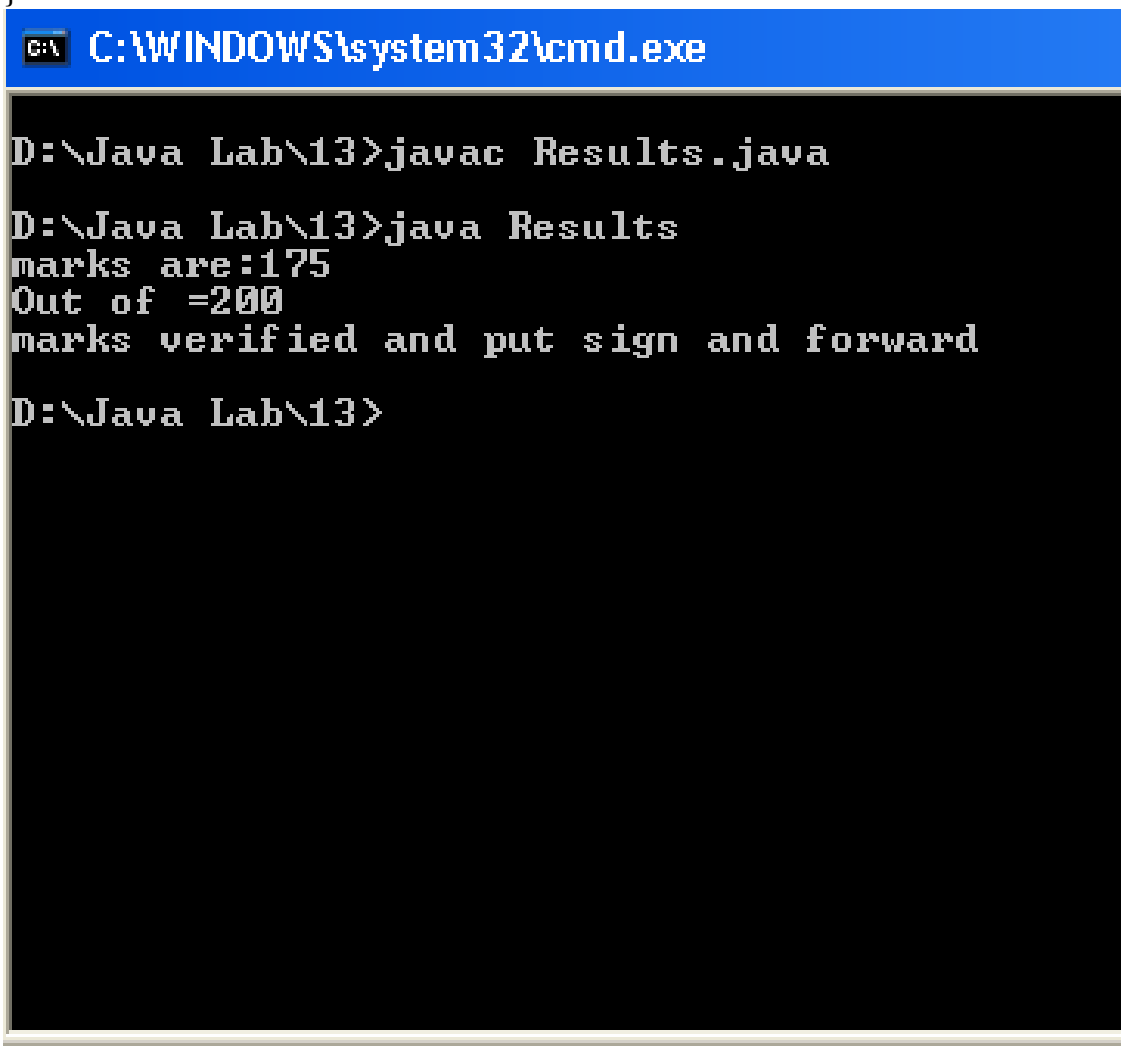
**EXAMPLE: A CLASS EXTENDS ONE CLASS AND IMPLEMENTS ONE INTERFACE (MULTIPLE INHERITANCE)**

```
class Teacher
{
int marks;
void setMark(int m)
{
marks=m;
}
void getMark()
{
System.out.println("marks are:"+marks);
}
}

interface Hod
{
int total=200;
void putSign();
}

class Results extends Teacher implements Hod
{
public void putSign()
{
System.out.println("marks verified and put sign and forward");
}
void display()
{
System.out.println("Out of ="+total);
}
public static void main(String args[])
{
Results r=new Results();
r.setMark(175);
r.getMark();
r.display();
```

```
r.putSign();
}
}
```


C:\WINDOWS\system32\cmd.exe
D:\Java Lab\13>javac Results.java
D:\Java Lab\13>java Results
marks are:175
Out of =200
marks verified and put sign and forward
D:\Java Lab\13>

**//A Class implements multiple interfaces (Multiple inheritance)**

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

interface Showable{
void show();
}

class A implements Printable,Showable{

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

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

**Output:**
Hello
Welcome

**//No ambiguity in multiple inheritance.**

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

interface Showable{
void print();
}

class A implements Printable,Showable{

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

```
}
```

Output:Hello

**MULTILEVEL INHERITANCE**

```
interface Printable{          0
void print();
}
interface Showable extends Printable{        1
void show();
}
class A implements Showable{              2
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}

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

Output:  Hello

Welcome