Interface

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 inheritances in Java.

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

It cannot be instantiated just like abstract class.

A class implements an interface, thereby inheriting the abstract methods of the interface.

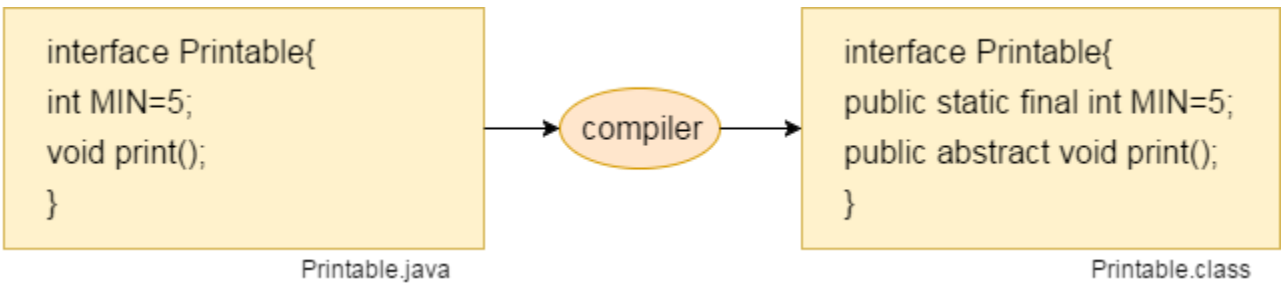
Why use Java interface?

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 inheritances.
- It can be used to achieve loose coupling.

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

A class uses the **implements** keyword to implement an interface. The implements keyword appears in the class declaration following the extends portion of the declaration.



Java Interface Example-1

```
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();

 }

}
```

Output:

```
Hello
```

Java Interface Example-2

```
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 method e.g.
//getDrawable()
d.draw();
}}
```

Output:

```
drawing circle
```

Interfaces supports Multiple Inheritance

```
interface Moveable
{
 boolean isMoveable();
}
interface Rollable
{
 boolean isRollable
}
class Tyre implements Moveable, Rollable
{
 int width;

 boolean isMoveable()
 {
  return true;
 }

 boolean isRollable()
 {
  return true;
 }
 public static void main(String args[])
 {
  Tyre tr=new Tyre();
  System.out.println(tr.isMoveable());
  System.out.println(tr.isRollable());
 }
}
```

Output :

true

true

Interface Inherits An Interface

```
public class Main {



   public static void main(String[] args) {

        shapeA circleshape=new circle();

          circleshape.Draw();
          circleshape.Draw();
   }
}

 interface shapeA
{
    public   String baseclass="shape";
    public void Draw();
}
 interface shapeB extends shapeA
{
    public   String baseclass="shape2";
    public void Draw2();
}
  class circle implements shapeB
{
   public   String baseclass="shape3";
   public void Draw() {
      System.out.println("Drawing Circle here:"+baseclass);
   }
   @Override
   public void Draw2() {
       System.out.println("Drawing Circle here:"+baseclass);
   }
}
```

```
Drawing Circle here:shape3
Drawing Circle here:shape3
```

Difference between an interface and an abstract class?

Abstract class	Interface
Abstract class is a class which contain one or more abstract methods, which has to be implemented by its sub classes.	Interface is a Java Object containing method declaration but no implementation. The classes which implement the Interfaces must provide the method definition for all the methods.
Abstract class is a Class prefix with an abstract keyword followed by Class definition.	Interface is a pure abstract class which starts with interface keyword.
Abstract class can also contain concrete methods.	Whereas, Interface contains all abstract methods and final variable declarations.
Abstract classes are useful in a situation that Some general methods should be implemented and specialization behavior should be implemented by child classes.	Interfaces are useful in a situation that all properties should be implemented.

Practice Example

Que. To solve the below Question.

Calculate the area of Triangle (formula ½*base*height)

Calculate the area of Rectangle (formula width*height)

Calculate the area of Square (formula a*a [length of height])

Calculate the area of Circle (formula pi*r*r [radius])

Solve this question using interface inheritance and abstract class