Interfaces in Java

- A java interface defines a blueprint of creating a class
- It has static constants and abstract methods.
- The interface in java is **a mechanism to achieve abstraction**
- It is used to achieve abstraction and multiple inheritance in Java.
- It cannot be instantiated.

An interface is similar to class in the following ways

✓The byte code of an interface appears in a .class file

✓An Interface is written in a file with .java extension

✓Interfaces may contain any number of methods

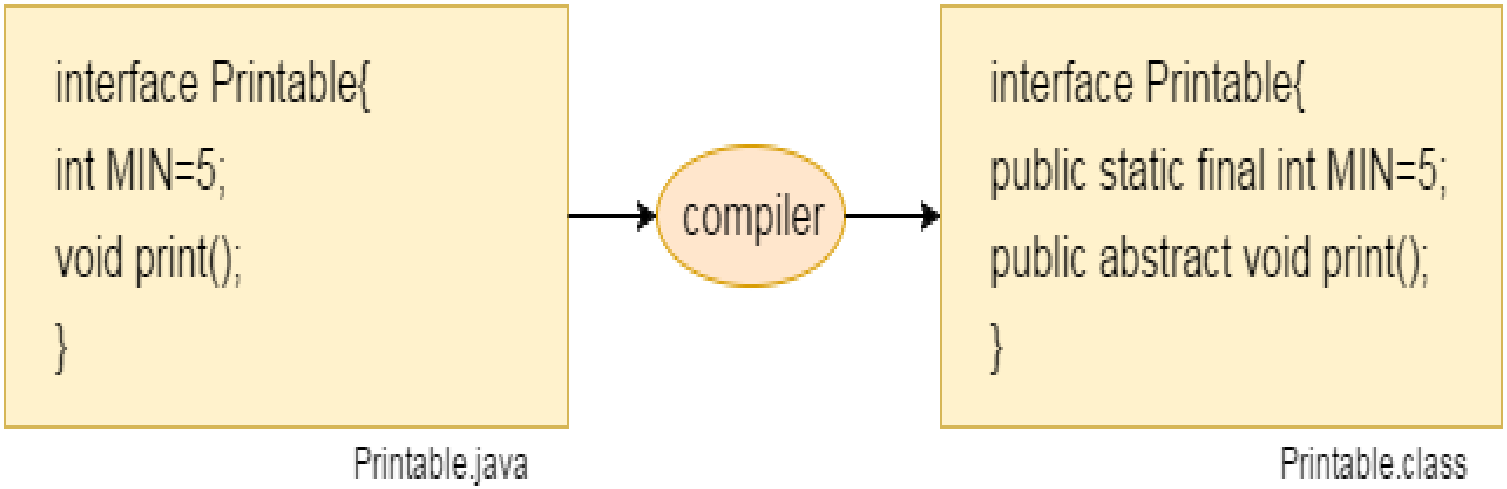
✓Interfaces can be arranged in packages and the file can be in a directory structure that matches the name of the package

Why use Java interface?

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.

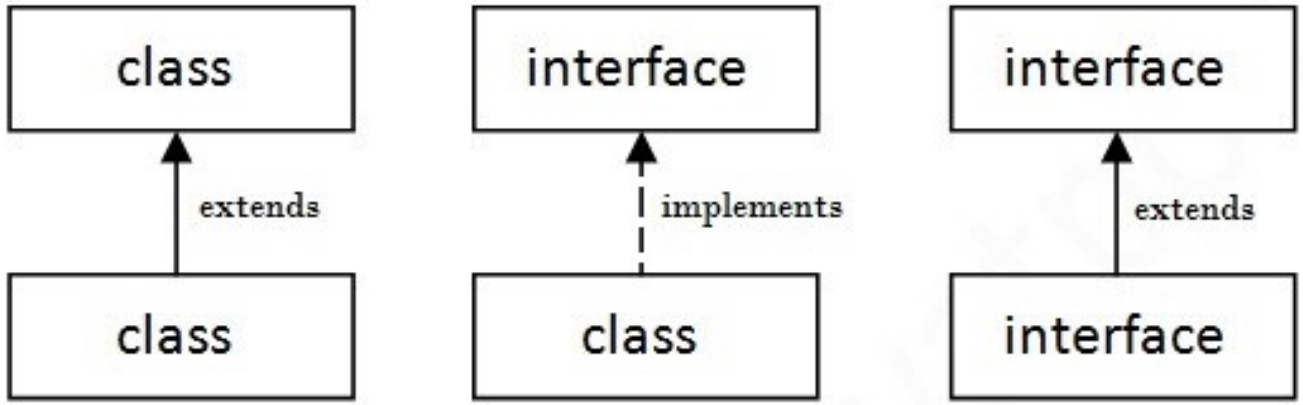
- **Java 8 Interface Improvement**
- Since Java 8, interface can have default and static methods

Interface fields are public, static and final by default, 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**.



Java Interface Example

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

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

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

Java Interface Example: Drawable

In this example, Drawable interface has only one method.
Its implementation is provided by Rectangle and Circle classes.
In real scenario, interface is defined by someone but implementation is provided by different implementation providers.
And, it is used by someone else.
The implementation part is hidden by the user which uses the interface.

```
//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();
d.draw();
}}
```

Java Interface Example: Bank

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

```
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());
}}
```

Multiple inheritance in Java by interface

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

Multiple Inheritance in Java

```
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();
 }
}
```

Interface inheritance

A class implements interface but one interface extends another interface

```
interface Printable{
void print();
}
interface Showable extends Printable{
void show();
}
class TestInterface implements Showable{
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}
public static void main(String args[]){
TestInterface obj = new TestInterface();
obj.print();
obj.show();  }}
```

Java 8 Default Method in Interface

Since Java 8, we can have method body in interface. But we need to make it default method.

```
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();
}}
```

Java 8 Static Method in Interface

Since Java 8, we can have static method in interface.

```
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));
}}
```

What is marker or tagged interface?

Marker interface in Java are interfaces with no field or methods or in simple word **empty interface in java is called marker interface**. Example of marker interface is Serializable, Clonnable and Remote interface

They are used to provide some essential information to the JVM so that JVM may perform some useful operation.

if JVM sees one Class is implementing Clonnable it performs some operation to support cloning.

In short Marker interface indicate, signal or a command to JVM.

- **Points to remember**
- We can’t instantiate an interface in java. That means we cannot create the object of an interface
- implements keyword is used by classes to implement an interface.
- While providing implementation in class of any method of an interface, it needs to be mentioned as public.
- Class that implements any interface must implement all the methods of that interface
- The interface body can contain abstract methods, default methods, and static methods
- Default methods are defined with the default modifier, and static methods with the static keyword.
- All abstract, default, and static methods in an interface are implicitly public, so you can omit the public modifier.
- Variables declared in interface are **public, static and final** by default.
- Interface variables must be initialized at the time of declaration otherwise compiler will throw an error.

- Inside any implementation class, you cannot change the variables declared in interface because by default, they are public, static and final.
- An interface can extend any interface but cannot implement it. Class implements interface and interface extends interface.
- A **class** can implement any **number of interfaces**.
- If there are **two or more same methods** in two interfaces and a class implements both interfaces, implementation of the method once is enough.

```
interface A {
public void aaa();
 }
interface B {
public void aaa();
}
class Central implements A,B {
 public void aaa() { //Any Code here }

 public static void main(String args[]) {
//Statements
 } }
```

Variable names conflicts can be resolved by interface name.

```
interface A {
int x=10;
}
interface B {
int x=100;
}
class Hello implements A,B {
 public static void main(String args[]) {
 System.out.println(A.x);
 System.out.println(B.x);
}
 }
```