FAST

National University of Computer and Emerging Sciences Peshawar

Lecture # 16

Software Construction and Development (Java Programming)

Instructor: Muhammad Abdullah Orakzai

DEPARTMENT OF COMPUTER SCIENCE

الذى علم بالقلم۔ علم الانسان ما لم يعلم۔

Abstraction in Java
(Interfaces in Java)

Contents

Interface in Java

- Used to implement common functionality.
- An **interface in Java** is a blueprint of a class.
- It has static and final field variables 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.

Interface in Java...

- It is used to achieve abstraction and multiple inheritance in Java.

- In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

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

Interface in Java…

- It cannot be instantiated just like the abstract class.

- **interface keyword** is used to declare an interface.

Why use Java interface?

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

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

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.

- A class that implements an interface must implement all the methods declared in the interface.

How to declare an interface?...

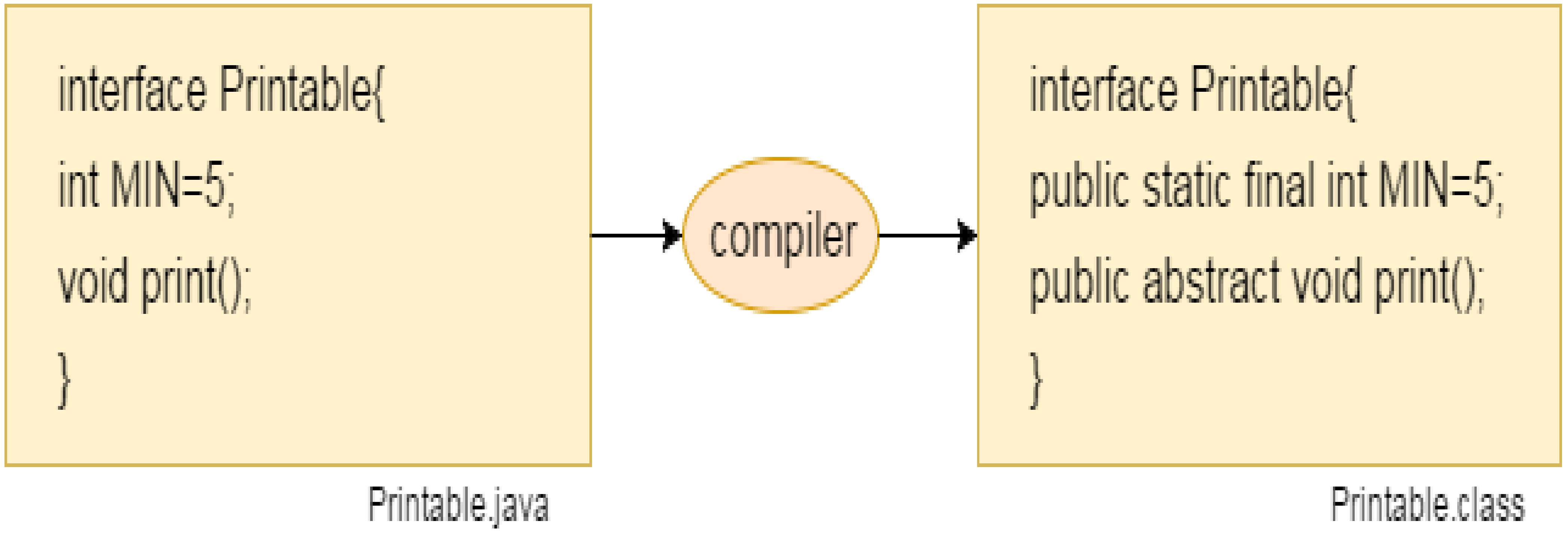
Syntax

```
interface <interface_name>
{
   // declare constant fields
   // declare methods that is abstract by default.
}
```

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.

Internal addition by the compiler



```
interface Printable{
int MIN=5;
void print();
}
```

Printable.java

```
interface Printable{
public static final int MIN=5;
public abstract void print();
}
```

Printable.class

Field variable declaration in interface

public static field datatype variableName = value;

OR

Datatype variableName = value;

Example

public static final int COUNT = 300;

OR

int COUNT   = 300;

Method declaration in interface

```
        public abstract datatype methodName(parameter list);
OR      public returntype methodName(parameter list);
OR      returntype methodName(parameter list);
```

Example

```
        public abstract int add(int a, int b);
OR      public  int add(int a, int b);
OR      int add(int a, int b);
```

implements keyword

Implements keyword is used when a class implements an interface.

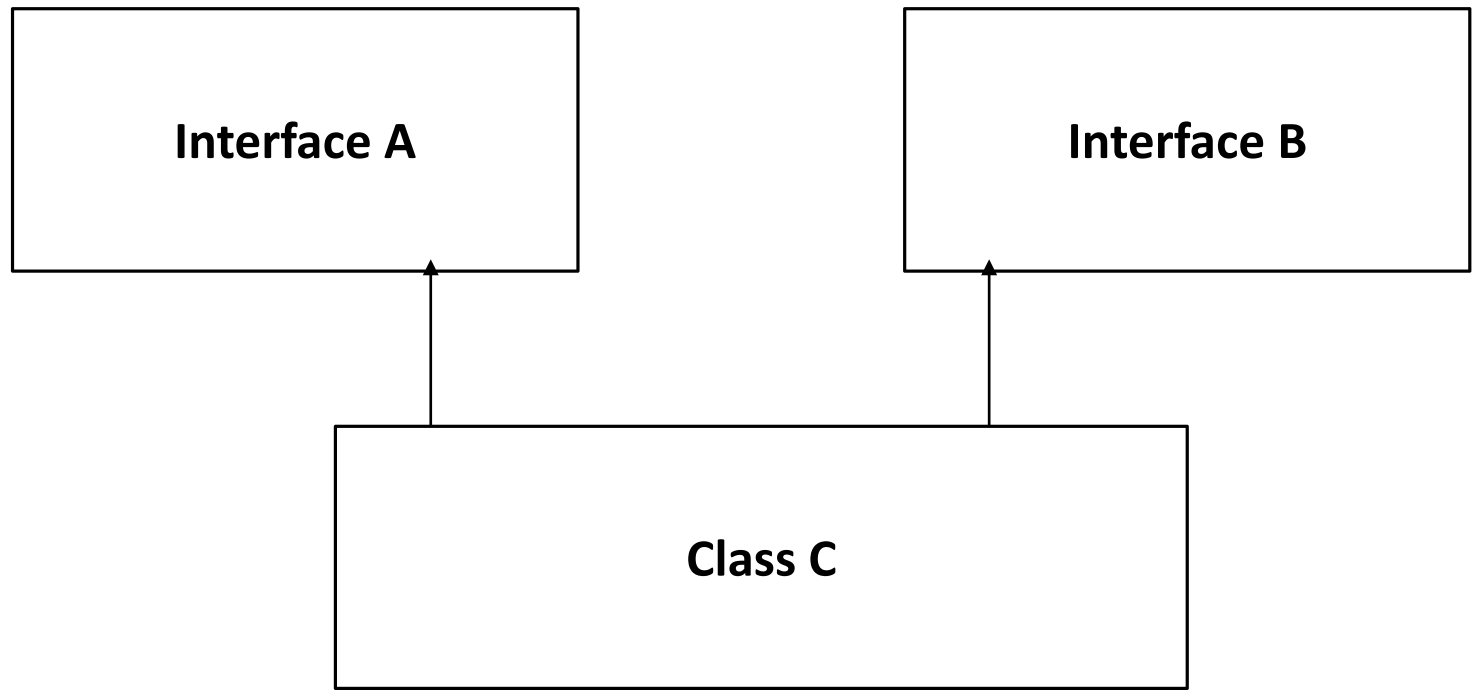
Syntax

class ClassName implements InterfaceName

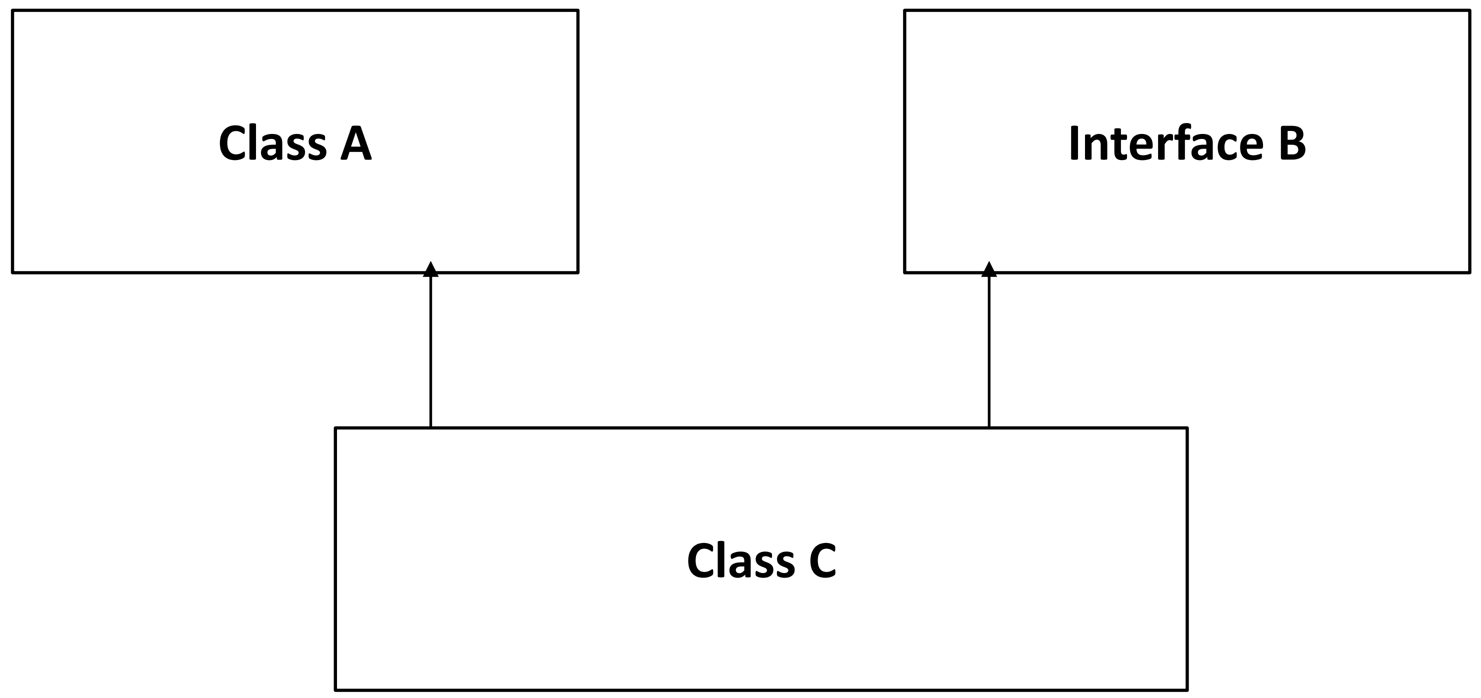
Note:

One class can implements more than one interfaces (multiple inheritance)

Multiple inheritance using interfaces



Multiple inheritance using interfaces…



Difference between class and interface

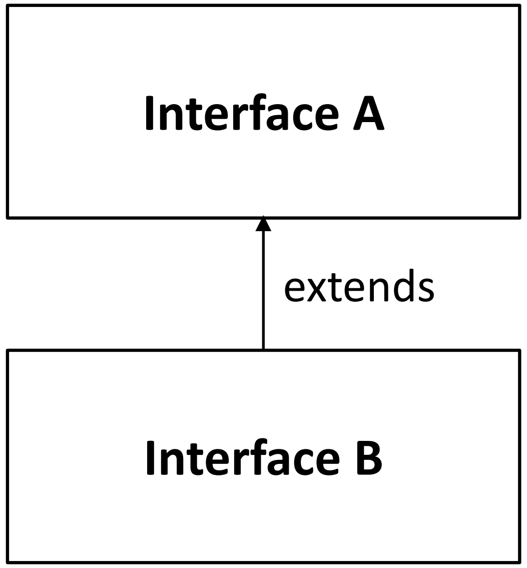
Class	Interface
1) Field variables may be normal, static or final.	Field variables must be static and final.
2) Methods have body in class.	Interface cannot be instantiated. It is implemented by other class and then object of that class is created for use.
3) Class can be instantiated.	Interface has **only static and final variables**.
4) **Access Specifier:** may be public, private and protected.	**Access Specifier:** only public is used.

Note

- One interface can inherit or extend another interface.
- The keyword extends is used.

Example

Interface B extends A

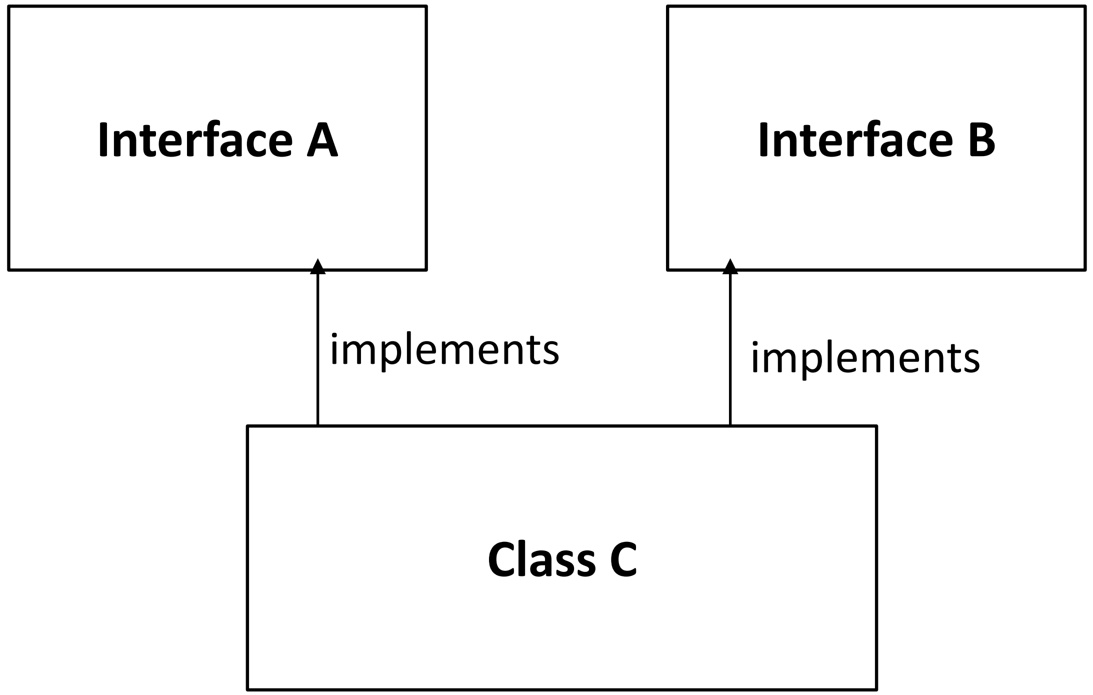


Note

One interface can inherit or extend more than one interfaces.

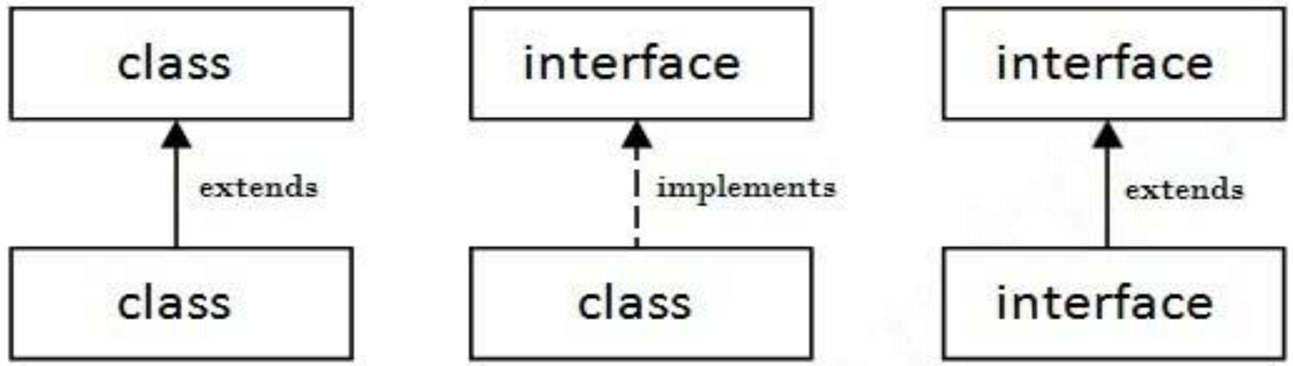
Example

Interface C extends A, B



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 A class.

```
interface Printable{

void print();

}    // Printable interface body end
```

Java Interface Example

```
class A implements Printable{
 public void print()
 {
 System.out.println("Hello");
 }
 }   // end of class A
```

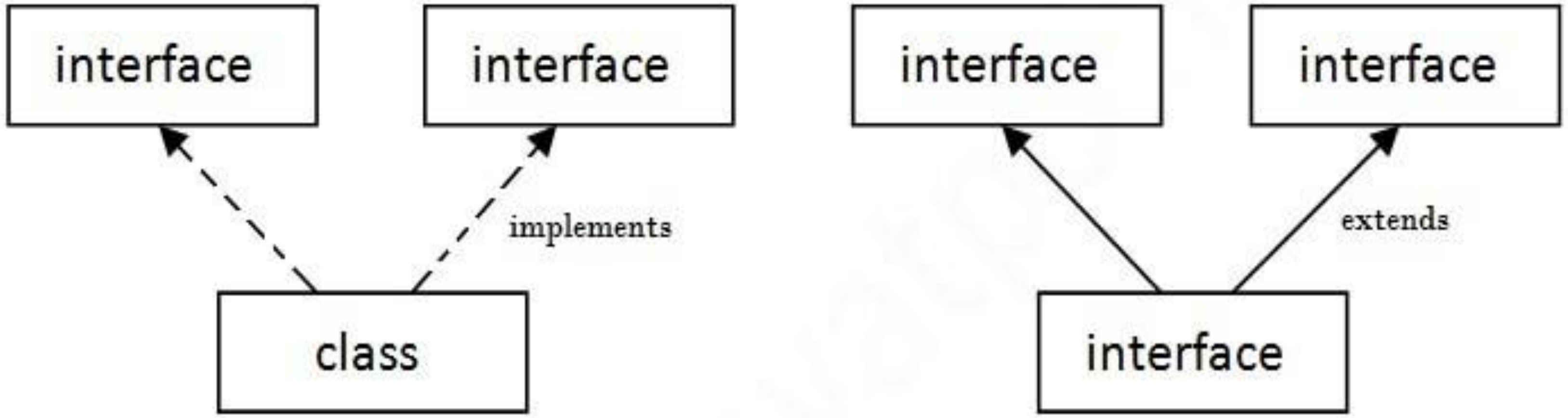
Java Interface Example…

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

Output: Hello

Multiple inheritance in Java by interface

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



Multiple Inheritance in Java

Multiple inheritance in Java by interface…

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

Multiple inheritance in Java by interface…

```
class B implements   Printable , Showable  {
public void print()   {
System.out.println("Hello");
  }
public void show()  {
System.out.println("Welcome");
 }
}    // end of B class
```

Multiple inheritance in Java by interface…

```
class  Test       // main class
{
public static void main(String args[])
{
B obj = new B();
obj.print();
obj.show();
 }
}    // end of Test class
```

Output

```
Hello
Welcome
```

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 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.

Multiple inheritance is not supported through class in java, but it is possible by an interface, why?...

For example:

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

Multiple inheritance is not supported through class in java, but it is possible by an interface, why?...

```
class TestInterface3 implements Printable, Showable{

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

public static void main(String args[]){

TestInterface3 obj = new TestInterface3();

obj.print();

 }

}
```

Output: Hello

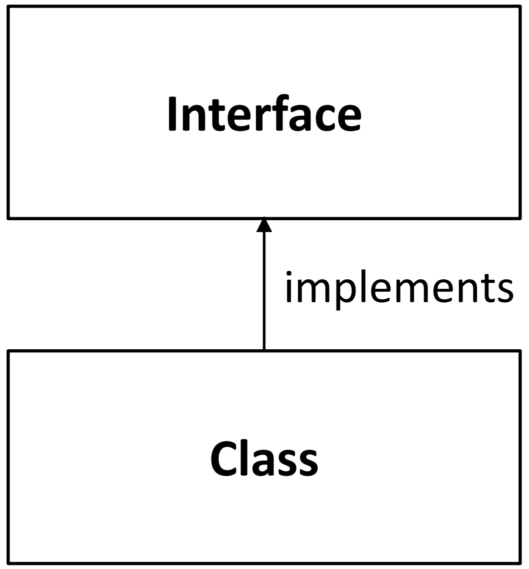
As you can see in the above example, Printable and Showable interface have same methods but its implementation is provided by class TestTnterface1, so there is no ambiguity.

Implementing interfaces

When a class implements interface

Syntax 1

```
class ClassName implements InterfaceName
{
Body of class
}
```



Implementing interfaces…

When a class implements multiple interfaces

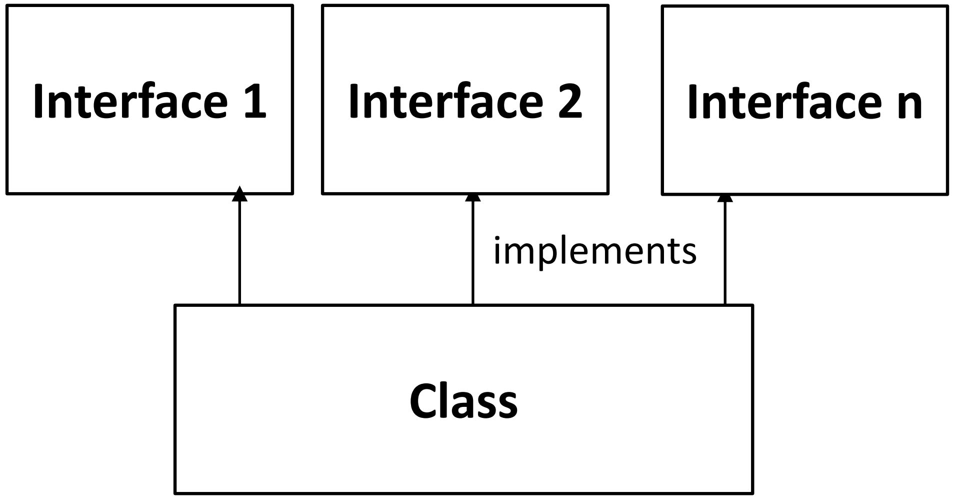
Syntax 2

class ClassName implements Interface 1, interface 2, …n

{

Body of class

}

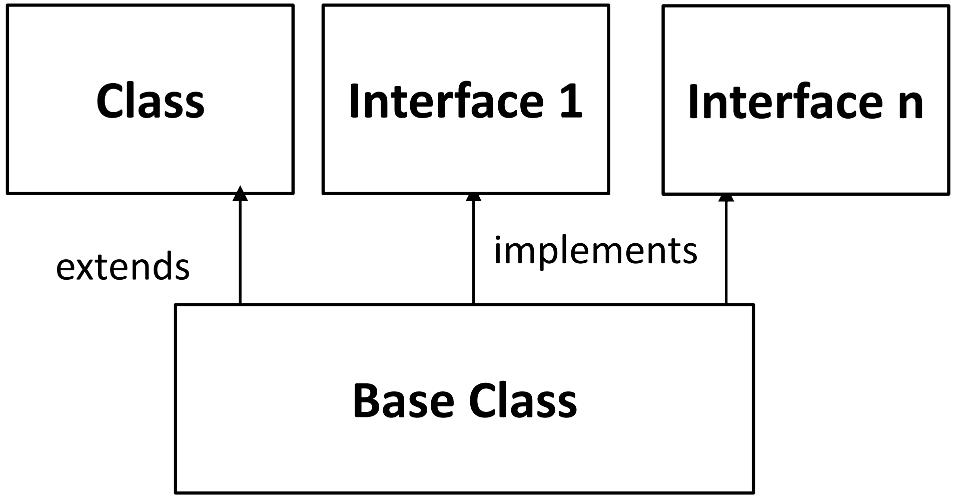


Implementing interfaces…

When a class extends another class and implements multiple interfaces.

Syntax 3

```
class ClassName extends superClassName implements Interface 1, interface 2, …n
{
Body of class
}
```



Interface inheritance

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

```
interface Printable{
void print();
}
interface Showable extends Printable{
void show();
}
```

Interface inheritance…

```
class TestInterface  implements Showable{
public void print()    {  System.out.println("Hello");   }
public void show()   {   System.out.println("Welcome");   }
}
class Test3 {      // main class
public static void main(String args[]){
TestInterface4 obj = new TestInterface4();
obj.print();
obj.show();
 }
}
```

Output

```
Hello
Welcome
```

Difference between abstract class and interface

- Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods.
- Abstract class and interface both can't be instantiated.

- Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).
- But there are many differences between abstract class and interface that are given below.

Difference between abstract class and interface…

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.

Difference between abstract class and interface…

6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

THANK YOU