SYNTAX TECHNOLOGIES

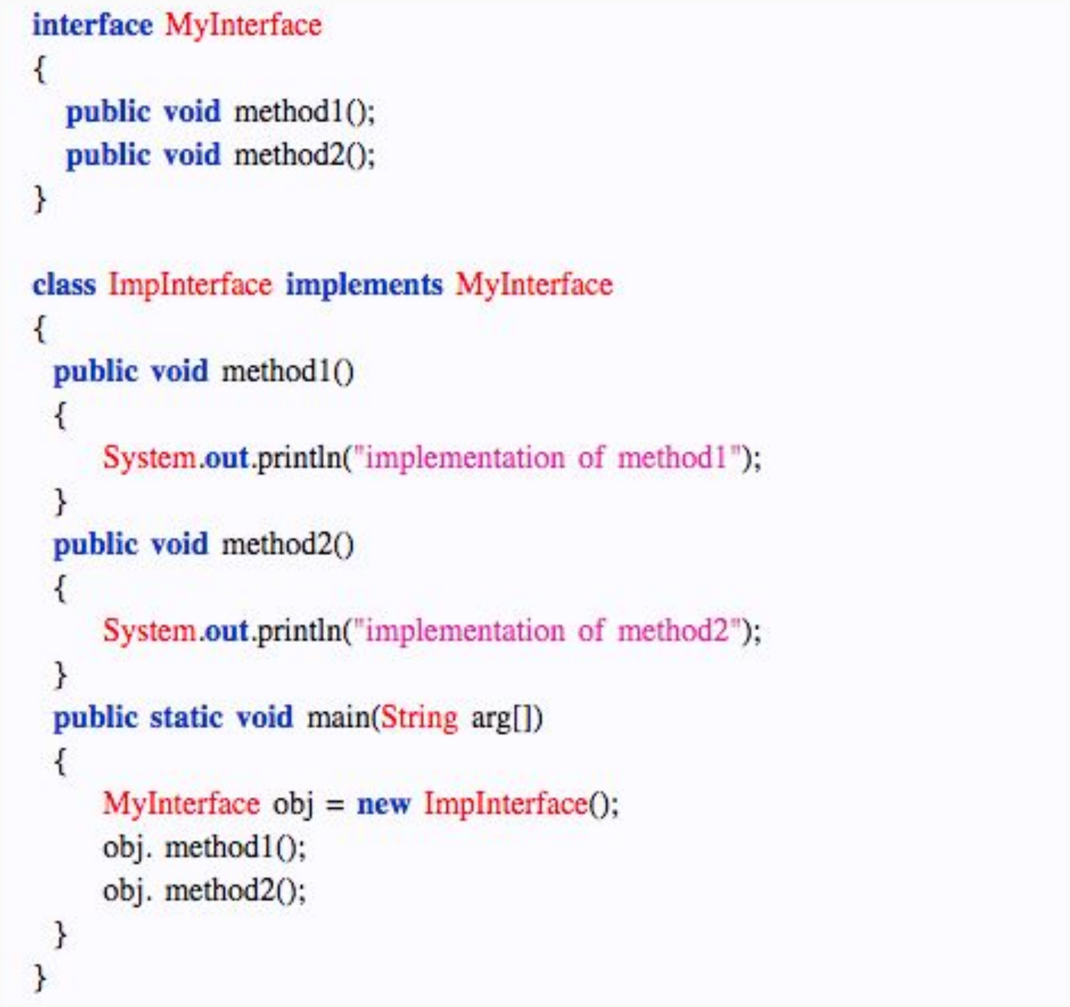
JAVA

Class 27

Agenda

Interface
Interface vs Abstract Class

Interface



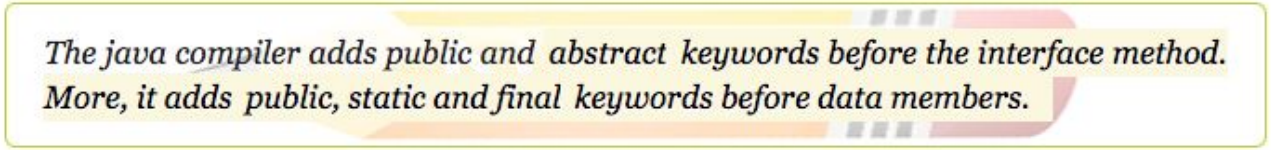
```
interface MyInterface
{
   public void method1();
   public void method2();
}

class ImpInterface implements MyInterface
{
  public void method1()
  {
      System.out.println("implementation of method1");
  }
  public void method2()
  {
      System.out.println("implementation of method2");
  }
  public static void main(String arg[])
  {
      MyInterface obj = new ImpInterface();
      obj. method1();
      obj. method2();
  }
}
```

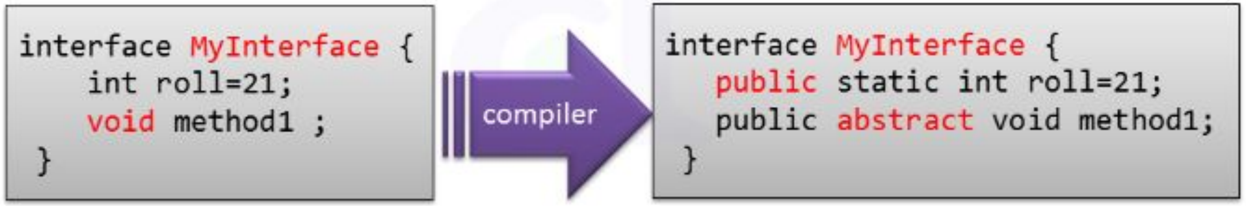
Interface

All methods of an interface are implicitly public abstract.

Interface adds public, static and final keywords before data members.



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



```
interface MyInterface {
    int roll=21;
    void method1 ;
 }
```

```
interface MyInterface {
   public static int roll=21;
   public abstract void method1;
 }
```

Implementing Interfaces

When we use abstract and when Interface?

If we do not know any things about implementation and just have requirement specification then we should be go for Interface

If we are talking about implementation but not completely (partially implemented) then we should be go for abstract

Note:
Prior to JDK 8, interface could not define implementation. Now we can add implemented **default** or **static** methods into interface.

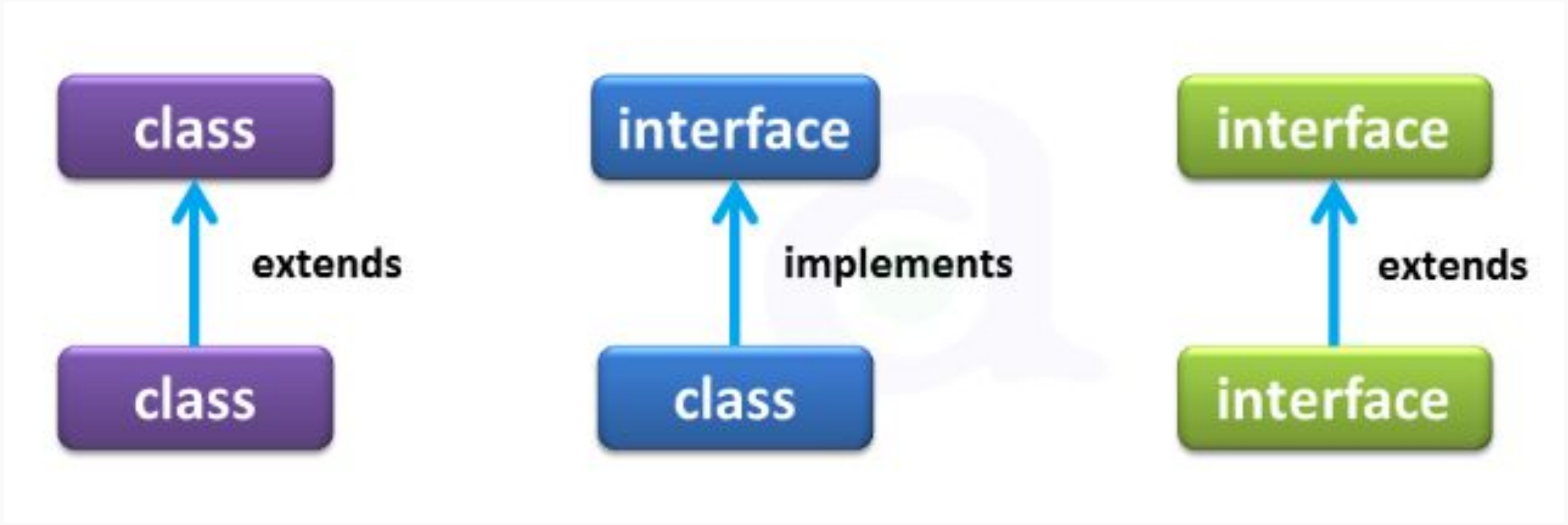
Rules to implement an interface

A class can implement more than one interface at a time.

A class can extend only one class, but implement many interfaces.

An interface can extend another interface, similarly to the way that a class can extend another class.

Relationship between class and Interface



A class uses the extends keyword to implement another class.
A class uses the implements keyword to implement an interface.
An interface uses the extends keyword to implement an interface.

Abstract Class vs Interface

Abstract class	**Interface**
The abstract keyword is used to declare abstract class	The interface keyword is used to declare interface
Abstract class does not support multiple inheritance	Interface support multiple inheritance
Abstract class contains Constructors	Interface doesn't contain Constructors
An abstract class contains both incomplete (abstract) and complete member and Abstract class can have abstract and non-abstract methods.	An interface contains only incomplete member (signature of member) and Interface can have only abstract methods (Since Java 8 we can have static and default methods)
An abstract class can contain access modifiers for the methods, properties(variables)	An interface cannot have access modifiers by default everything is assumed as public
Abstract class can have final, static and non-static variables.	Interface has only final static variables.

Abstract Class vs Interface

Abstract class	**Interface**
Abstract class may contain either variable or constants.	Interface contains only constants.
The default access specifier of abstract class methods are default.	The default access specifier of interface method are public.
These class properties can be reused in other class using extend keyword.	These properties can be reused in any other class using implements keyword.
For the abstract class there is no restriction like initialization of variable at the time of variable declaration.	For the interface it should be compulsory to initialization of variable at the time of variable declaration.
There are no any restriction for abstract class variable.	For the interface variable can not declare variable as private, protected
There are no any restriction for abstract class method modifier that means we can use any modifiers.	For the interface method cannot be declared as protected, private, final

Example of Interface

```
interface Person {

void run();  // abstract method
}
class A implements Person {
     public void run() {
     SOP("Run fast");
}
public static void main(String
args[]) {
 A obj = new A();
 obj.run();
 }
}
```

Multiple Inheritance using interface Example

```
interface Developer {
    void code();
}

interface Tester {
    void test();
}
```

```
class Employee implements
Developer, Tester {

public void code() {
   SOP("writes code");
}

public void test() {
    SOP("tests code");
}

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