# JAVA

Class 28

## Agenda

Interface in Java

Interface vs Abstract Class

# Interface in Java

Interface is similar to class which is collection of public static final variables (constants) and public abstract methods.

The interface is a mechanism to achieve fully abstraction in java.

There can be only abstract methods in the interface. It is used to achieve fully abstraction and multiple inheritance in Java.

An interface is a collection of methods that have no implementation - they are just created, but have no functionality.

An interface is a bit like a class, except you can only declare methods and variables in the interface. You cannot actually implement the methods.

## Interface in Java

By using Interface, we can achieve multiple inheritance in java.

It is implicitly abstract. So we no need to use the abstract keyword when declaring an interface.

Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.

Methods in an interface are implicitly public.

All the data members of interface are implicitly public static final. Interface can not contain instance fields. Interface only contains public static final variables. The java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members.

# Interface in Java

Interface cannot be extended by a class; it is implemented by a class.

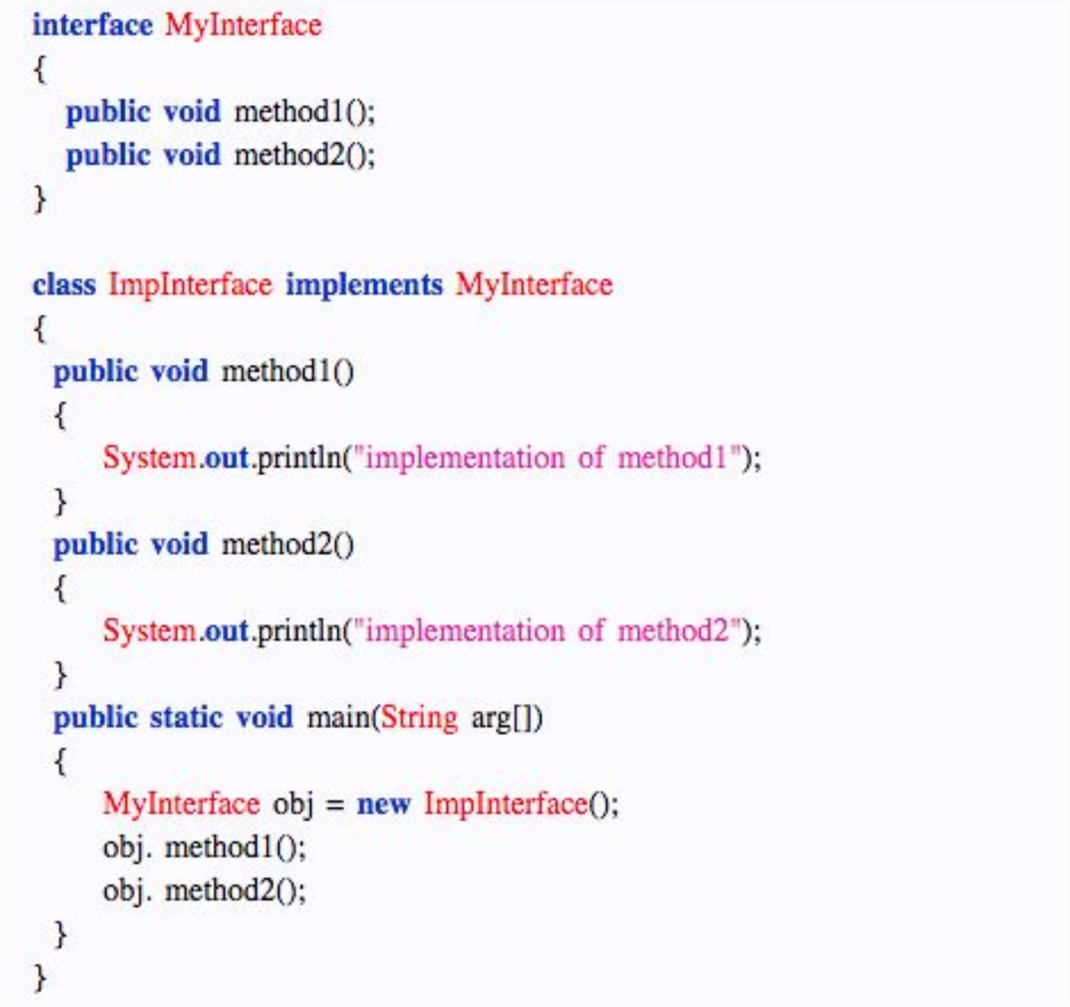
Interface can extend multiple interfaces. It means interface support multiple inheritance

An interface reference can point to objects of its implementing classes

An interface is not a class. Writing an interface is similar to writing a class, but they are two different concepts. A class describes the attributes and behaviors of an object. An interface contains behaviors that a class implements.

Unless the class that implements the interface is abstract, all the methods of the interface need to be defined in the 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();
  }
}
```

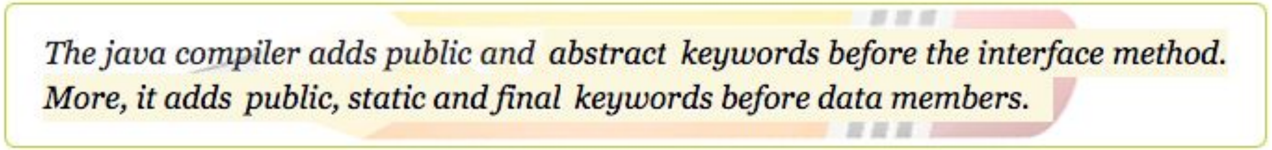
## Task

1. Create a WebDriver Interface that will have the following unimplemented behaviour: openBrowser(), closeBrowser(), maximizeWindow(), findElement(). Create 2 classes that implements WebDriver interface: ChromeDriver and FirefoxDriver.

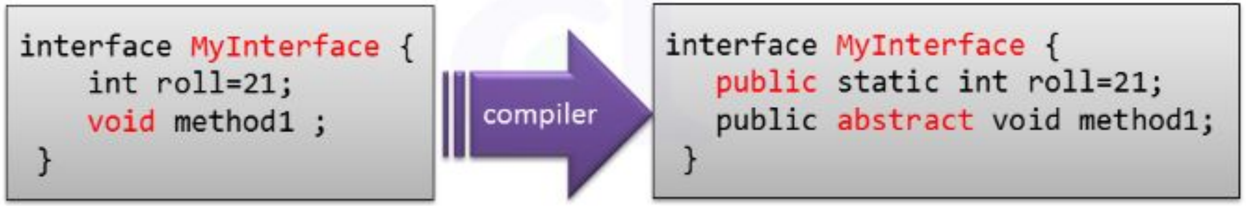
# Interface

**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

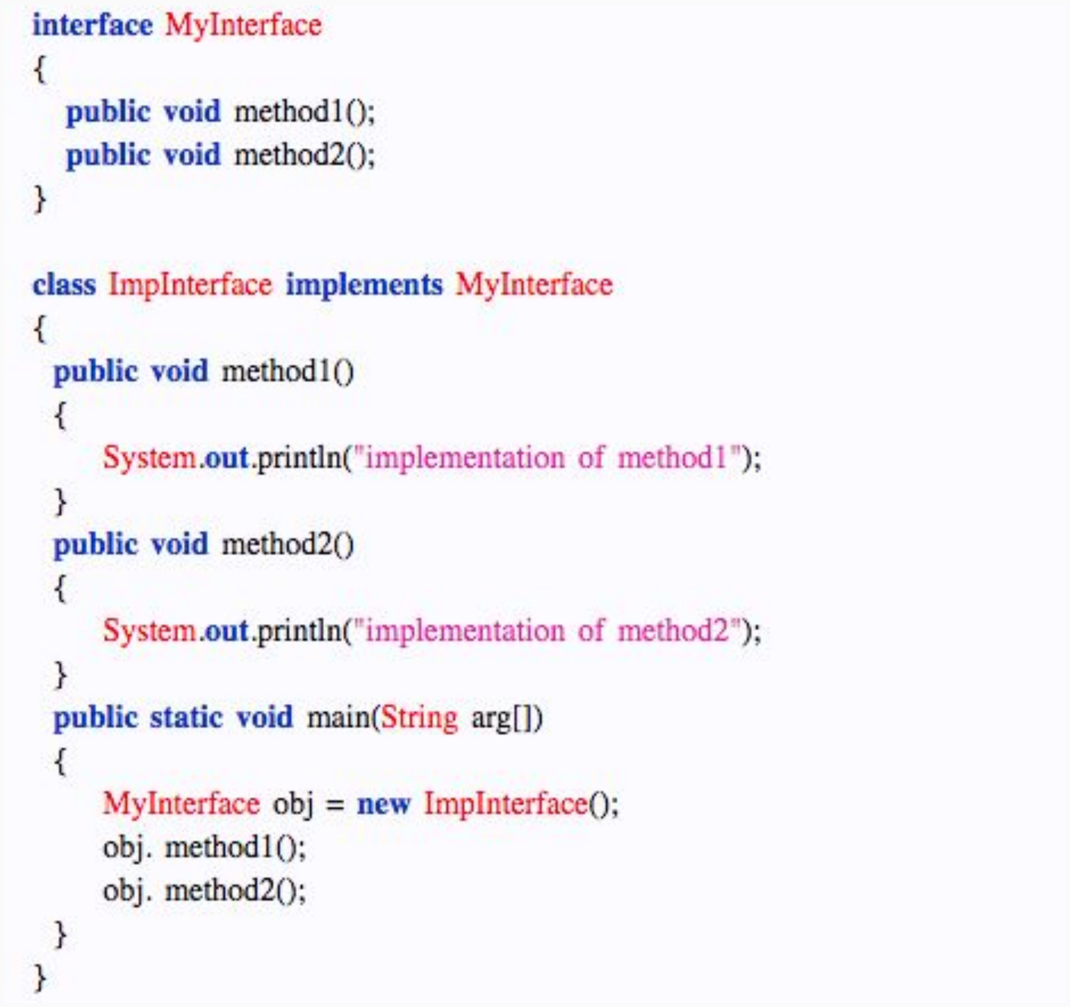
## Rules to implement an interface

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.

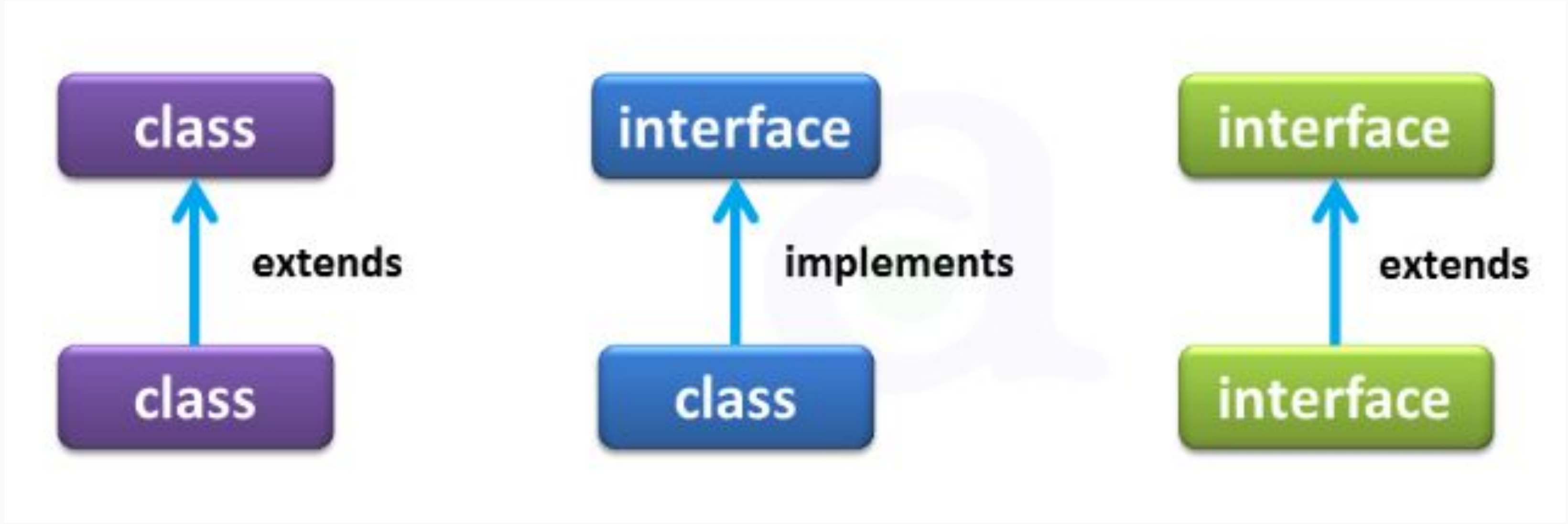
## Interface



```
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();
  }
}
```

## 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. |
| 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** |
|---|---|
| There properties can be reused commonly in a specific application. | There properties commonly usable in any application of java environment. |
| Abstract class may contain either variable or constants. | Interface should 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 declare method as protected, static, 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 disp();
}
interface Manager
{
void show();
}

class Employee implements
Developer, Manager
{
public void disp()
{
SOP("Hello Good Morning");
}
public void show()
{
SOP("How are you ?");
}
public static void main(String
args[])
{
Employee obj=new Employee();
obj.disp();
obj.show();
}
}
```