Interfaces

An interface declares a set of methods and their signatures. The methods that are declared in an interface should not have implementation codes. A class that makes use of an interface should provide the codes for the methods declared in that interface.

Important points

1. An interface in java is a blueprint of a class. it has static constants and abstract methods.
2. Methods should be only declared in an interface. i.e they should not have implementation codes.
3. All methods that are declared in an interface are automatically public.
4. An interface does not have instance variables. But can have constants. Such constants will be inherited by the class that implements the interface.
5. A class that implements an interface should provide the implementation codes for the methods declared in that interface.
6. An interface may be implemented by different classes in different ways.
7. Every variable in an interface is implicitly public, static and final.
8. Every method in an interface is implicitly public and abstract.
9. This provides dynamic method resolution or polymorphism.

An interface is similar to a class in the following ways −

* An interface can contain any number of methods.
* An interface is written in a file with a **.java** extension, with the name of the interface matching the name of the file.
* The byte code of an interface appears in a **.class** file.
* Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

However, an interface is different from a class in several ways, including −

* You cannot instantiate an interface.
* An interface does not contain any constructors.
* All of the methods in an interface are abstract.
* An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
* An interface is not extended by a class; it is implemented by a class.
* An interface can extend multiple interfaces.

Structure of an interface

General syntax

interface name

{

type variable-name1=value1;

:

type variable-nameN=valueN;

return-type method-name1 (parameter-list);

:

return-type method-nameN (parameter-listN);

}

Implementation of an interface

A class shall make use of an interface by implementing the same.

General syntax:

**class** class-name **implements** interface-name

{

:

}

If a class implements an interface, it should implement all the methods declared in that interface. Else the compiler will generate an error. A class can implement more than one interfaces.

**class** class-name **implements** interface1, interface2,..

{

:

}

A class shall extend another class and at the same time , it shall also implement one or more interfaces.

Example:

**class** Circle**extends** Shape **implements** shape2D

{

:

}

Multiple inheritance can be achieved as follows:

**class** Circle**extends** Shape **implements** shape2D, shape3D

{

:

}

Another example

Implementation of an Interface

A class shall make use of an interface by the keyword **implements**. If a class implements an interface, it should implement all the methods declared in that interface. Other wise the complier will generate an error message.

Interface example 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Interface

interface A

{

void method1();

void method2();

}

interface B extends A

{

void method3();

}

class myclass implements B

{

public void method1()

{

System.out.println("this is interface A: method1()");

}

public void method2()

{

System.out.println("this is interface A: method2()");

}

public void method3()

{

System.out.println("this is interface B: method3()");

}

}

public class interface1 {

public static void main(String args[])

{

myclass ob=new myclass();

ob.method1();

ob.method2();

ob.method3();

}

}

Another example

Class E extends A implements B, C, D

{

:

}

|  |
| --- |
| interface Shape2D  {  double getArea();  }  class Circle implements Shape2D  {  int radius;  Circle(int r)  {  this.radius=r;  }  public double getArea()  {  return Math.PI\*radius\*radius;  }  }  class Circledemo  {  public static void main(String args[])  {  Circle c1=new Circle(5);  double area;  area=c1.getArea();  System.out.println("area of circle is "+area);  }  } |

Output

|  |
| --- |
| E:\javaprgs\interfaces>javac Circledemo.java  E:\javaprgs\interfaces>java -cp .Circledemo  area of circle is 78.53981633974483 |

Example 2:

This program inherit a class and implement an interface

|  |
| --- |
| interface Shape2D  {  double getArea();  }  class Shape  {  void display()  {  System.out.println("Name of the shape is circle");  }  }  class Circle extends Shape implements Shape2D  {  int radius;  Circle(int r)  {  this.radius=r;  }  public double getArea()  {  return Math.PI\*radius\*radius;  }  }  class Interfacedemo2  {  public static void main(String args[])  {  Circle c1=new Circle(5);  double area;  area=c1.getArea();  c1.display();  System.out.println("area of circle is "+area);  }  } |

Output

|  |
| --- |
| E:\javaprgs\interfaces>javac interfacedemo2.java  E:\javaprgs\interfaces>java -cp . Interfacedemo2  Name of the shape is circle  area of circle is 78.53981633974483 |

Example 3:

Multiple use of an interface

|  |
| --- |
| interface Shape2D  {  double getArea();  }  class Shape  {  void display(String s)  {  System.out.println("Name of the shape is circle "+s);  }  }  class Circle extends Shape implements Shape2D  {  int radius;  Circle(int r)  {  this.radius=r;  }  public double getArea()  {  return Math.PI\*radius\*radius;  }  }  class Square extends Shape implements Shape2D  {  int side;  Square(int s)  {  this.side=s;  }  public double getArea()  {  return side\*side;  }  }  class Interfacedemo3  {  public static void main(String args[])  {  Circle c1=new Circle(5);  double area;  area=c1.getArea();  c1.display("Circle");  System.out.println("area of circle is "+area);  Square s1=new Square(10);  s1.display("squre");  area=s1.getArea();  System.out.println("area of square is "+area);  }  } |

Output

|  |
| --- |
| E:\javaprgs\interfaces>javac Interfacedemo3.java  E:\javaprgs\interfaces>java -cp . Interfacedemo3  Name of the shape is circle Circle  area of circle is 78.53981633974483  Name of the shape is circle squre  area of square is 100.0 |

Interface inheritance

Syntax

Interface name-1 extends name-2,name-3..

Example

Interface shape2d extends shape3d

{

:

}

Here shape2d and shape3d are interfaces

Interface threewheeler extends vehicle, twowheeler

{

:

}

Here threewheeler, vehicle, twowheeler are interfaces

Example

|  |
| --- |
| // program to illustrate interface inheritance concept  interface intface1  {  int j=20;  int j1();  }  interface intface2  {  double k1();  }  interface intface3 extends intface1,intface2  {  boolean l1();  }  class intimplement implements intface3  {  public int j1()  {  return 10;  }  public double k1()  {  return 20.9;  }  public boolean l1()  {  return true;  }  }  public class intfacedemo1  {  public static void main(String args[])  {  intimplementobj=new intimplement();  System.out.println(obj.j);  System.out.println(obj.j1());  System.out.println(obj.k1());  System.out.println(obj.l1());  }  } |

Output

|  |
| --- |
| E:\javaprgs>javac intfacedemo1.java  E:\javaprgs>java -cp . intfacedemo1  20  10  20.9  true |

Filename:wheeler.java

|  |
| --- |
| interface Vehicle  {  void changeGear(int a);  void speedUp(int a);  void applyBrakes(int a);  }  class Bicycle implements Vehicle  {  int speed;  int gear;  public void changeGear(int g)  {  this.gear=g;  }  public void speedUp(int increment)  {  speed=speed+increment;  }  public void applyBrakes(int decrement)  {  speed=speed-decrement;  }  public void display()  {  System.out.println("speed= "+speed+" gear "+gear);  }  }  class Bike implements Vehicle  {  int speed;  int gear;  public void changeGear(int g)  {  this.gear=g;  }  public void speedUp(int increment)  {  speed=speed+increment;  }  public void applyBrakes(int decrement)  {  speed=speed-decrement;  }  public void display()  {  System.out.println("speed= "+speed+" gear "+gear);  }  }  class wheeler  {  public static void main(String args[])  {  Bicycle bicycle=new Bicycle();  bicycle.changeGear(2);  bicycle.speedUp(3);  bicycle.applyBrakes(1);  System.out.println("Bicycle present state : ");  bicycle.display();  Bike bike=new Bike();  bike.changeGear(1);  bike.speedUp(4);  bike.applyBrakes(3);  System.out.println("Bike present state : ");  bike.display();  }  } |

// example for interface inheritance

interface intface1

{

int j=20;

int method1();

}

interface intface2

{

double method2();

}

interface intface3 extends intface1,intface2

{

boolean method3();

}

class sample implements intface3

{

public int method1()

{

return j;

}

public double method2()

{

return 20.5;

}

public boolean method3()

{

return true;

}

}

public class inheritinterface {

public static void main(String args[])

{

sample s=new sample();

System.out.println(s.j);

System.out.println(s.method1());

System.out.println(s.method2());

System.out.println(s.method3());

}

}

interface TVStation

{

String satellitename="satsun";

String cabletv="sun";

double freq=12.3;

void show();

}

class broadcast implements TVStation

{

int bctime;

String date;

broadcast(int b,String d)

{

bctime=b;

date=d;

}

void display()

{

System.out.println(" the broadcast time is :"+bctime);

System.out.println(" the broadcast date is :"+date);

}

public void show()

{

System.out.println(" the satellite name is :"+satellitename);

System.out.println(" the cable tv name is :"+cabletv);

System.out.println(" the frequencies is :"+freq);

}

}

public class interface5 {

public static void main(String args[])

{

broadcast b=new broadcast(7,"1/8/2012");

b.show();

b.display();

}

}

interface TVStation

{

String satellitename="satsun";

String cabletv="sun";

double freq=12.3;

void show();

}

class programme

{

String pname;

String sponsor;

programme(String p, String s)

{

this.pname=p;

this.sponsor=s;

}

void display()

{

System.out.println(" the programme name is :"+pname);

System.out.println(" the sponsor is :"+sponsor);

}

}

class broadcast extends programme implements TVStation

{

int bctime;

String date;

broadcast(int b,String d,String p,String s)

{

super(p,s);

bctime=b;

date=d;

}

void display()

{

super.display();

System.out.println(" the broadcast time is :"+bctime);

System.out.println(" the broadcast date is :"+date);

}

public void show()

{

System.out.println(" the satellite name is :"+satellitename);

System.out.println(" the cable tv name is :"+cabletv);

System.out.println(" the frequencies is :"+freq);

}

}

public class interface5 {

public static void main(String args[])

{

broadcast b=new broadcast(7,"1/8/2012","oru kodi","dabur");

b.show();

b.display();

}

}

//output

the satellite name is :satsun

the cable tv name is :sun

the frequencies is :12.3

the programme name is :oru kodi

the sponsor is :dabur

the broadcast time is :7

the broadcast date is :1/8/2012

interface TVStation

{

String satellitename="satsun";

String cabletv="sun";

double freq=12.3;

void show();

}

interface teleInformation

{

String tvname="samsung";

double price=16400.00;

void showDetails();

}

class programme

{

String pname;

String sponsor;

programme(String p, String s)

{

this.pname=p;

this.sponsor=s;

}

void display()

{

System.out.println(" the programme name is :"+pname);

System.out.println(" the sponsor is :"+sponsor);

}

}

class broadcast extends programme implements TVStation

{

int bctime;

String date;

broadcast(int b,String d,String p,String s)

{

super(p,s);

bctime=b;

date=d;

}

void display()

{

super.display();

System.out.println(" the broadcast time is :"+bctime);

System.out.println(" the broadcast date is :"+date);

}

public void show()

{

System.out.println(" the satellite name is :"+satellitename);

System.out.println(" the cable tv name is :"+cabletv);

System.out.println(" the frequencies is :"+freq);

}

}

class viewer extends broadcast implements teleInformation

{

viewer(int b,String d,String p,String s)

{

super(b,d,p,s);

}

public void showDetails()

{

System.out.println(" the tv name is "+tvname);

System.out.println(" the price of tv is "+price);

}

}

public class interface6 {

public static void main(String args[])

{

broadcast b=new broadcast(7,"1/8/2012","oru kodi","dabur");

b.show();

b.display();

viewer v=new viewer(8,"2/8/2012","aaha","hamam");

v.display();

v.show();

v.showDetails();

}

}

// output

--------------------Configuration: <Default>--------------------

the satellite name is :satsun

the cable tv name is :sun

the frequencies is :12.3

the programme name is :oru kodi

the sponsor is :dabur

the broadcast time is :7

the broadcast date is :1/8/2012

the programme name is :aaha

the sponsor is :hamam

the broadcast time is :8

the broadcast date is :2/8/2012

the satellite name is :satsun

the cable tv name is :sun

the frequencies is :12.3

the tv name is samsung

the price of tv is 16400.0

Process completed.

Output

|  |
| --- |
| E:\javaprgs>javac wheeler.java  E:\javaprgs>java -cp . wheeler  Bicycle present state :  speed= 2 gear 2  Bike present state :  speed= 1 gear 1 |

Packages

A package is a collection of related classes and interfaces. Each source file must have the package statement of the following form as the first statement:

package *package-name*;

Class files can be kept as separate source file. For example if a package **P**consists of three class files. The three files can be kept as separate source file as**a.java , b.java, c.java or the a and b class can be placed in one file and c class can be kept in another file.**

**But in one source file only one class can be declared as public. The main rule of package is that only the class which are declared as public can be accessed in other files in the same package.**

A simple package example

|  |
| --- |
| // package  package mypack;  public class simple  {  public static void main(String args[])  {  System.out.println("welcome to package");  }  } |

The folder called mypack has to be created. The package name also must be same as the folder name.

In the above example mypack folder is created and the first statement is

package mypack;

This says that this belongs to the package mypack.

The above program has to be executed in its parent folder. In this case it is javaprgs

The directory structure is

E:>javaprgs>mypack>

Within this the class file as well as source files are present.

The file has to be executed from its parent folder.

Output:

|  |
| --- |
| E:\javaprgs>java -cp .mypack.simple  welcome to package |

Another example : the name of the file is : Balance.java

This is saved in folder mypack1 and so the package name is also mypack1.

|  |
| --- |
| // a simple package  package mypack1;  public class Balance  {  String name;  double bal;  public Balance(String n,double b)  {  name=n;  bal=b;  }  public void show()  {  if(bal<0)  System.out.println("--->");  System.out.println(name + ": $"+bal);  }  } |

Now create another file called testbalance.java in the same folder mypack1 and include the package name as mypack1.

|  |
| --- |
| package mypack1;  class testbalance  {  public static void main(String args[])  {  Balance test=new Balance("jj",99.88);  test.show();  }  } |

The program can be executed like this.

After the creation of class file the file can be executed with **packagename.classname**

In the above example it is

mypack1.testbalance

mypack1 is the package name and testbalance is the class filename

|  |
| --- |
| E:\javaprgs>javac mypack1\Balance.java  E:\javaprgs>javac -cp . mypack1\testbalance.java  E:\javaprgs>java -cp . mypack1.testbalance  jj: $99.88 |

Another example : Accountbalance.java

|  |
| --- |
| package mypack2;  class balance  {  String name;  double bal;  balance(String n, double b)  {  name=n;  bal=b;  }  void show()  {  if(bal<0)  System.out.println("--->");  System.out.println(name +"--->"+bal);  }  }  class Accountbalance  {  public static void main(String args[])  {  balance current[]=new balance[3];  current[0]=new balance("kj",123.33);  current[1]=new balance("wt",134.90);  current[2]=new balance("ss",-5.9);  for (inti=0;i<3;i++)  current[i].show();  }  } |

output

|  |
| --- |
| E:\javaprgs>java -cp . mypack2.Accountbalance  kj--->123.33  wt--->134.9  --->  ss--->-5.9 |

Another example file name: R1.java package name: r1

|  |
| --- |
| package r1;  class R1  {  protected int i=10;  public void r1()  {  System.out.println("this is from class R1 "+i);  }  } |

File name : R2.java

|  |
| --- |
| package r1;  class R2 extends R1  {  public void r2()  {  System.out.println("this is from class R2 "+i);  }  } |

File name : R3.java

|  |
| --- |
| package r1;  class R3  {  public void r3()  {  R1 obj=new R1();  System.out.println("this is from class R3 "+obj.i);  }  } |

The program which calls all the above class files (R1.class, R2.class, R3.class ) is Accessdemo1.java

|  |
| --- |
| package r1;  class Accessdemo1  {  public static void main(String args[])  {  System.out.println("this is accessing all class in the same package");  R1 obj1=new R1();  obj1.r1();  R2 obj2=new R2();  obj2.r2();  R3 obj3=new R3();  obj3.r3();  }  } |

|  |
| --- |
| r1 |

In the above picture inside the folder javaprgs>r1 is created. Inside r1 all the java files (R1.java, R2.java,R3.java and Accessdemo1.java) are created. The program has to be executed from the parent folder i.ejavaprgs

output

|  |
| --- |
| E:\javaprgs>javac -cp . r1\R1.java  E:\javaprgs>javac -cp . r1\R2.java  E:\javaprgs>javac -cp . r1\R3.java  E:\javaprgs>javac -cp . r1\Accessdemo1.java  E:\javaprgs>java -cp . r1.Accessdemo1  this is accessing all class in the same package  this is from class R1 10  this is from class R2 10  this is from class R3 10 |

Using import statement :

To use the classes declared in different packages , import statement is used.

In the following example three packages namely r1, s1, q1 are created(folders).

In the package r1 – R1 class is created

**Another example:**

The files classA.java and packtest1.java are stored in folder named : p1 (package name also)

The file classC.java is stored in folder : p2 (package name)

Filename :classA

|  |
| --- |
| package p1;  public class classA  {  public void displayA()  {  System.out.println("class A");  }  } |

Filename : packtest1.java

|  |
| --- |
| package p1;  class packtest1  {  public static void main(String args[])  {  classAobj=new classA();  obj.displayA();  } |

Filename : classC.java

|  |
| --- |
| package p2;  import p1.\*;  class classB  {  public void displayB()  {  System.out.println("class B");  }  }  class classC  {  public static void main(String args[])  {  classBobjB=new classB();  classAobjA=new classA();  objA.displayA();  objB.displayB();  }  } |

In this in order to use the classA which is present in p1 package it has to be imported in classC which is present in package p2.

Output

|  |
| --- |
| E:\javaprgs>javac p1\classA.java  E:\javaprgs>javac -cp . p1\packtest1.java  E:\javaprgs>java -cp . p1.packtest1  class A  E:\javaprgs>javac -cp . p2\classC.java  E:\javaprgs>java -cp . p2.classC  class A  class B |

Example: geometry2d, geometry3d – packages

Filename : Circle.java – creates two class files namely Circle.class and shape2d.class

Package name: geometry2d

|  |
| --- |
| package geometry2d;  interface shape2d  {  double computeArea();  }  public class Circle implements shape2d  {  int radius;  public Circle(int r)  {  this.radius=r;  }  public double computeArea()  {  return 3.14\*radius\*radius;  }  } |

Filename : Cube.java – creates two class files namely shape3d.class and Cube.class

Package name: geometry3d

|  |
| --- |
| package geometry3d;  interface shape3d  {  double computeVolume();  }  public class Cube implements shape3d  {  int side;  public Cube(int s)  {  this.side=s;  }  public double computeVolume()  {  return side\*side\*side;  }  } |

The above two packages are stored in package called : packages

i.e

packages-> geometry2d-> Circle.java (Circle.class , shape2d.class)

packages-> geometry3d-> Cube.java (Cube.class , shape3d.class)

in the package > folder create the following file called **importdemo2.java**

|  |
| --- |
| import geometry2d.\*;  import geometry3d.\*;  public class importdemo2  {  public static void main(String args[])  {  Circle obj1=new Circle(5);  System.out.println("Area= "+obj1.computeArea());  Cube obj2=new Cube(5);  System.out.println("Volume= "+obj2.computeVolume());  }  } |

Output

|  |
| --- |
| E:\javaprgs\packages\geometry2d>javac Circle.java  E:\javaprgs\packages\geometry2d>cd ..  E:\javaprgs\packages>cd geometry3d  E:\javaprgs\packages\geometry3d>javac Cube.java  E:\javaprgs\packages\geometry3d>cd..  E:\javaprgs\packages>javac -cp . importdemo2.java  E:\javaprgs\packages>java -cp . importdemo2  Area= 78.5  Volume= 125.0 |

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.

But there are many differences between abstract class and interface that are given below.

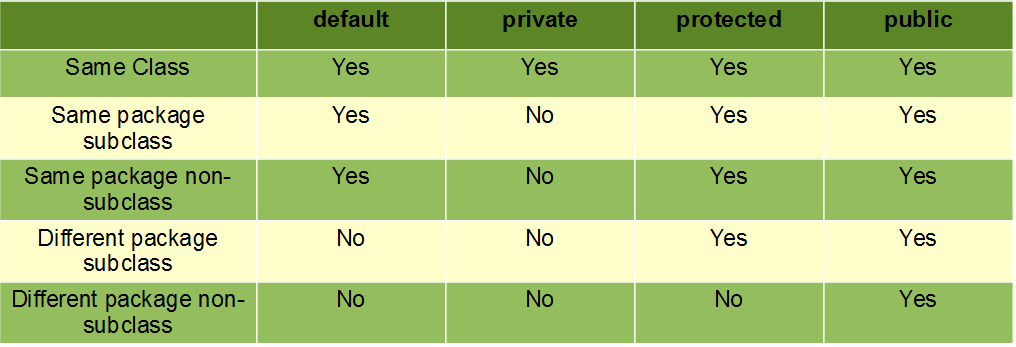
|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) Abstract class **doesn't support multiple inheritance**. | 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. |
| 6) **Example:** public abstract class Shape{ public abstract void draw(); } | **Example:** public interface Drawable{ void draw(); } |

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

Access modifiers in java

There are four types of access modifiers available in java:

1. Default – No keyword required
2. Private
3. Protected
4. Public



**Default**: When no access modifier is specified for a class, method or data member – It is said to be having the **default** access modifier by default.

* The data members, class or methods which are not declared using any access modifiers i.e. having default access modifier are accessible **only within the same package**.

In this example, we will create two packages and the classes in the packages will be having the default access modifiers and we will try to access a class from one package from a class of second package.

Folder name: p1\_default file name: welcome.java

|  |
| --- |
| package p1\_default;  class welcome  {  void display()  {  System.out.println("welcome to package1");  }  } |

Folder name: p2\_default file name: welcome1.java

|  |
| --- |
| package p2\_default;  import p1\_default.\*;  class welcome1  {  public static void main(String args[])  {  welcome obj=new welcome();  obj.display();  System.out.println("this is from another package");  }  } |

While executing the following error will come. As the class welcome.class is default access modifier.

Output

|  |
| --- |
| D:\javaprgs>javac -cp . p2\_default\welcome1.java  p2\_default\welcome1.java:7: error: welcome is not public in p1\_default; cannot b  e accessed from outside package  welcome obj=new welcome();  ^  p2\_default\welcome1.java:7: error: welcome is not public in p1\_default; cannot b  e accessed from outside package  welcome obj=new welcome();  ^  2 errors |