Chapter 4.2

Packages and Interface

Package

- The package statement defines a name space in which classes are stored.
- If we omit the package statement, the class names are put into the default package, which has no name.
- While the default package is fine for short, sample programs, it is inadequate for real applications.
- General form of package:
  − package packagename;
- Java uses file system directories to store packages. For example, the .class files for any classes you declare to be part of MyPackage must be stored in a directory called MyPackage.
- Packages can be created in hierarchies:
  − package pkg1[.pkg2[.pkg3]];
- Example of package hierarchy:
  − package java.awt.image;
- needs to be stored in java/awt/image, java\awt\image, or java:awt:image on your UNIX, Windows, or Macintosh file system, respectively

Finding packages and classpath

- the Java run-time system uses the current working directory as its starting point.
- Thus, if our package is in the current directory, or a subdirectory of the current directory, it will be found.
- Second, we can specify a directory path or paths by setting the CLASSPATH environmental variable.

//File1.java
Package Lab
class ABC
{
…...
}
class XYZ
{
…….
}
	//File2.java
	Package Lab
	class PQR
	{
	…...
	}
	class EFG
	{
	…….
	}

Access Protection

	Private	No modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Importing Package

- To use classes,enumeration, etc defined within particular package, we have to import package.
- Syntax:
  − Import pkg1[.pkg2][.pkg3]…..[.pkgn].(classname/*);
  − e.g
  ● Import Lab.*; //import al classes of Lab package
  ● Import Lab.ABC; //import only class ABC of lab package
  ● Import Bim.Lab.*;//import all classes of sub package Lab //of Bim package

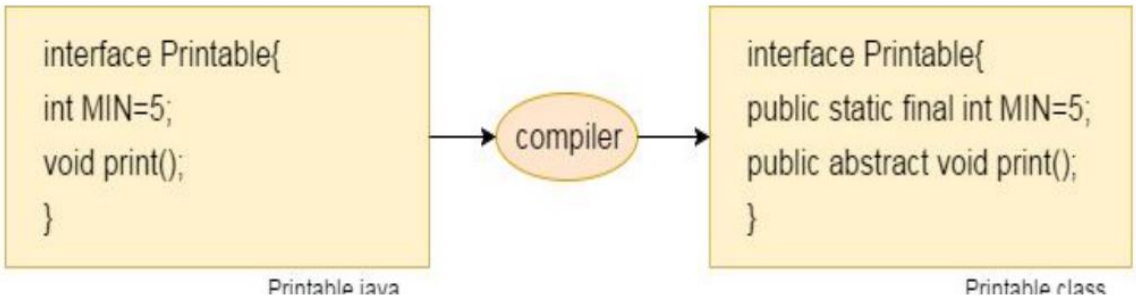
Interface

- An interface in java is a blueprint of a class. It has static constants 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. It is used to achieve abstraction and multiple inheritance in Java.
- It cannot be instantiated just like abstract class.

Defining an interface

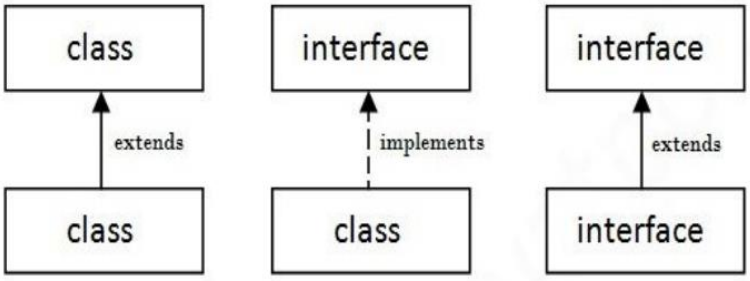
```
access interface name {
return-type method-name1(parameter-list);
return-type method-name2(parameter-list);
type final-varname1 = value;
type final-varname2 = value;
// ...
return-type method-nameN(parameter-list);
type final-varnameN = value;
}
```

Interface fields are public, static and final by default, and methods are public and abstract.



Understanding Relationships between classes and interfaces

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



```
interface printable {
void print();
}
class A6 implements printable {
public void print() {System.out.println("Hello");}
public static void main(String args[]){
A6 obj = new A6();
obj.print();
}
}
```

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

Output : ROI : 9.15

Variables in Interfaces

- We can use interfaces to import shared constants into multiple classes by simply declaring an interface that contains variables which are initialized to the desired values ,e.g:

```
interface testable2
{
        int a=1; //variable in interface
        int b=2; //
        void test2();
        int test1();

}
```

Nested Interface in Java

```
interface Test
{
interface Yes
{ void show();
}
}
class Testing implements Test.Yes
{
public void show()
{
System.out.println("show method of interface");
}
}
```

```
class A
{
public static void main(String[] args)
{
Test.Yes obj;
Testing t = new Testing();
obj=t;
obj.show();
}
}
```