**ASSIGNMENT NO.-G-27**

**Title:**

Write a Java program which will demonstrate a concept of Interfaces and packages: In this assignment design and use of customized interfaces and packages for a specific application are expected.

**Objectives:**

1. To understand concept of Interface in java language.

2. To understand concept of Package in java language.

**Software Required:** java, 64 bit Fedora, eclipse IDE

**Theory:**

**1. Interface:**

An **interface in java** is a blueprint of a class. It has static constants and abstract methods only.

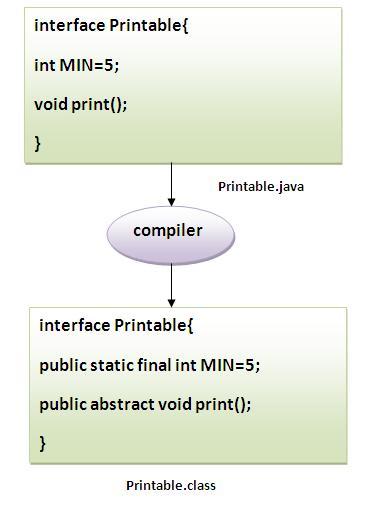
The interface in java is **a mechanism to achieve fully abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java. It cannot be instantiated just like abstract class.

**Why use Java interface?**

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

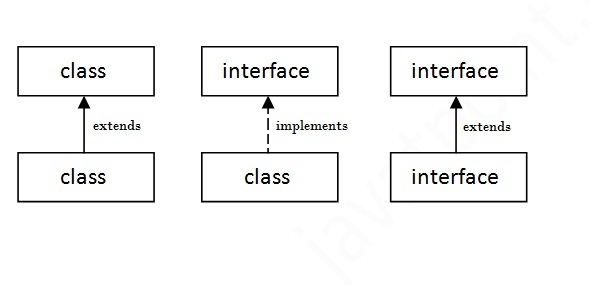
* It is used to achieve fully abstraction.
* By interface, we can support the functionality of multiple inheritance.
* It can be used to achieve loose coupling.

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



#### Understanding 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.



**Simple example of Java interface:**

In this example, Printable interface have only one method, its implementation is provided in the A class.

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();

 }

}

OUTPUT: Hello

## Multiple inheritance in Java by interface:

If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance.

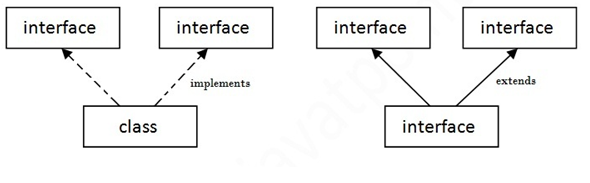


Fig.: Multiple Inheritance in java

**2.Package:**

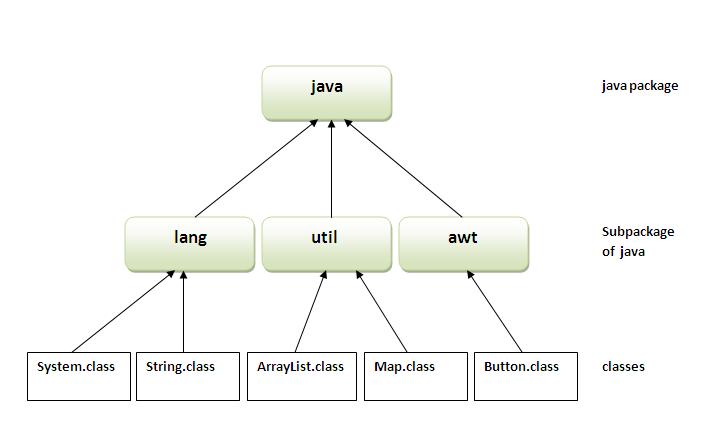
A **java package** is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two form, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

## Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.



**Simple example of java package:**

The package keyword is used to create a package in java.

//save as Simple.java

package mypack;

public class Simple

{

 public static void main(String args[])

{

    System.out.println("Welcome to package");

   }

}

**How to compile java package**

If you are not using any IDE, you need to follow the syntax given below:

1. javac -d directory javafilename

For example

1. javac -d . Simple.java

The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home (in case of Linux), d:/abc (in case of windows) etc. If you want to keep the package within the same directory, you can use . (dot).

**How to run java package program**

You need to use fully qualified name e.g. mypack.Simple etc to run the class.

**To Compile:** javac -d . Simple.java

To Run: java mypack.Simple

Output: Welcome to package

The -d is a switch that tells the compiler where to put the class file i.e. it represents destination. The . represents the current folder.

## How to access package from another package?

There are three ways to access the package from outside the package.

1. import package.\*;
2. import package.classname;
3. fully qualified name.

#### 1) Using packagename.\*

If you use package.\* then all the classes and interfaces of this package will be accessible but not subpackages.

The import keyword is used to make the classes and interface of another package accessible to the current package.

## Example of package that import the packagename.\*

**Program 1.**

//save by A.java

package pack;

public class A{

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

}

**Program 2.**

//save by B.java

package mypack;

import pack.\*;

class B{

  public static void main(String args[]){

   A obj = new A();

   obj.msg();

  }

}

**Output:**Hello

#### 2) Using packagename.classname

If you import package.classname then only declared class of this package will be accessible.

## Example of package by import package.classname

## Program 1.

//save by A.java

package pack;

public class A{

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

}

## Program 2.

//save by B.java

package mypack;

import pack.A;

class B{

  public static void main(String args[]){

   A obj = new A();

   obj.msg();

  }

}

Output:Hello

#### 3) Using fully qualified name

If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.

It is generally used when two packages have same class name e.g. java.util and java.sql packages contain Date class

## Example of package by import fully qualified name

**Program 1.**

//save by A.java

package pack;

public class A{

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

}

**Program 2.**

//save by B.java

package mypack;

class B{

  public static void main(String args[]){

   pack.A obj = new pack.A();//using fully qualified name

   obj.msg();

  }

}

**Output:**Hello

**Software Required:** java, 64 bit Fedora, eclipse IDE

**Conclusion:** This program gives us the knowledge about interfaces and packages and use of abstraction.

**Questions asked in university exam.**

1. **What is a Java package and how is it used?**
2. Name few classes of package java.io?
3. Which package stores all the standard java classes?
4. What is an interface in Java?
5. Can we create non static variables in an interface?
6. Can we declare interface members as private or protected?