OopJava15:M14

From JNTU-MSIT 2015

OOPS with Java

→ Week1

→ Week2

→ Week3

→ Week4

Contents

Module14: Java Packages

Learning Objective

Java Package Example of package Accessing package

Introduction : Package class

A java package is a group of similar types of classes, interfaces and sub-packages.

Definition: A package is a grouping of related types providing access protection and name space management. Note that types refers to classes, interfaces, enumerations, and annotation types. Enumerations and annotation types are special kinds of classes and interfaces, respectively, so types are often referred to in this lesson simply as classes and interfaces.

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.

Here, we will have the detailed learning of creating and using user-defined packages.
Advantage of Java Package

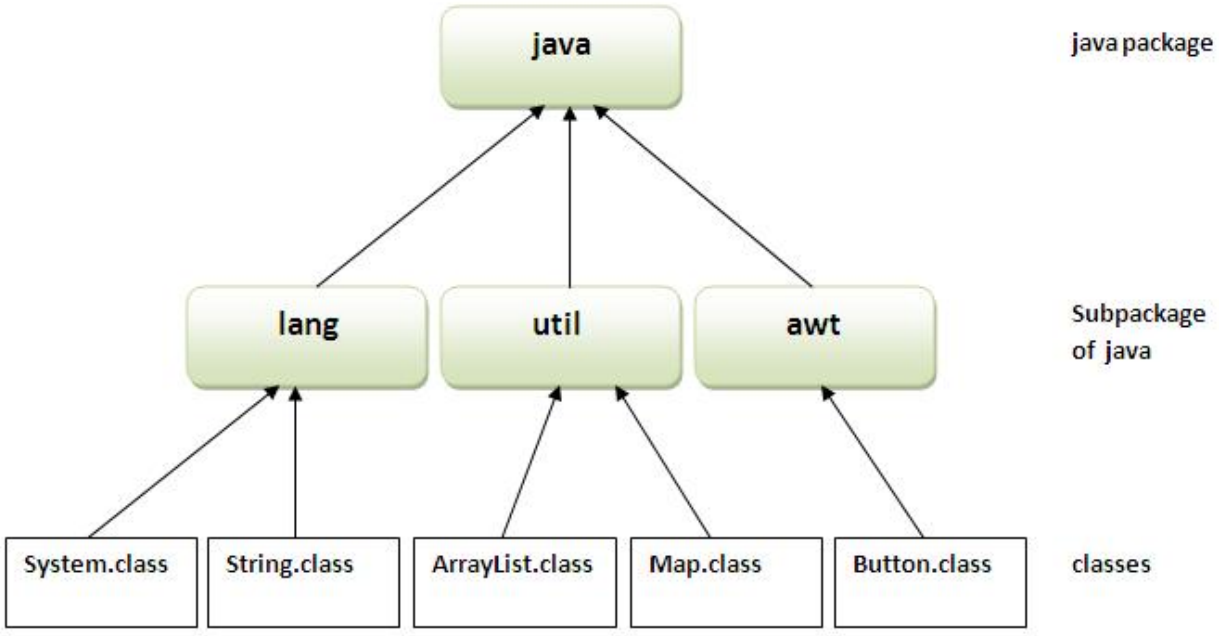
1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

[Video : Packages]

[Video : Packages Example]



Pic: package in java 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:

javac -d directory javafilename
For example

javac -d . Simple.java
The -d switch specifies the destination where to put the generated class file. You can use any directory name li
```

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 . repr
```

How to access package from another package?

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

```
import package.*;
import package.classname;
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.*
```

```
//save by A.java

package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}

//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

```
//save by A.java

package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//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

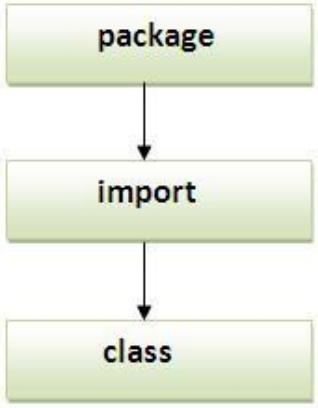
```
//save by A.java

package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//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
```

```
Note: If you import a package, subpackages will not be imported.
```

If you import a package, all the classes and interface of that package will be imported excluding the classes and interfaces of the subpackages. Hence, you need to import the subpackage as well.

```
Note: Sequence of the program must be package then import then class.
```

sequence of package



Subpackage in java

Package inside the package is called the subpackage. It should be created to categorize the package further.

Let's take an example, Sun Microsystem has definded a package named java that contains many classes like System, String, Reader, Writer, Socket etc. These classes represent a particular group e.g. Reader and Writer classes are for Input/Output operation, Socket and ServerSocket classes are for networking etc and so on. So, Sun has subcategorized the java package into subpackages such as lang, net, io etc. and put the Input/Output related classes in io package, Server and ServerSocket classes in net packages and so on.

The standard of defining package is domain.company.package e.g. com.javatpoint.bean or org.sssit.dao.

Example of Subpackage

```
package com.javatpoint.core;
class Simple{
  public static void main(String args[]){
   System.out.println("Hello subpackage");
  }
}
```

```
To Compile: javac -d . Simple.java
To Run: java com.javatpoint.core.Simple
Output:Hello subpackage
```

How to send the class file to another directory or drive?

There is a scenario, I want to put the class file of A.java source file in classes folder of c: drive. For example: how to put class file in another package

//save as Simple.java

```
package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Welcome to package");
   }
}
```

To Compile:

```
e:\sources> javac -d c:\classes Simple.java
```

To Run:

```
To run this program from e:\source directory, you need to set classpath of the directory where the class file res
e:\sources> set classpath=c:\classes;.;
e:\sources> java mypack.Simple
```

Another way to run this program by -classpath switch of java:

```
The -classpath switch can be used with javac and java tool.
```

To run this program from e:\source directory, you can use -classpath switch of java that tells where to look for class file. For example:

```
e:\sources> java -classpath c:\classes mypack.Simple
```

Output:Welcome to package

Ways to load the class files or jar files There are two ways to load the class files temporary and permanent. Temporary

```
By setting the classpath in the command prompt
By -classpath switch
```

Permanent

```
By setting the classpath in the environment variables
By creating the jar file, that contains all the class files, and copying the jar file in the jre/lib/ext folder.
```

```
Rule: There can be only one public class in a java source file and it must be saved by the public class name.
```

```
//save as C.java otherwise Compilte Time Error
class A{}
class B{}
public class C{}
```

How to put two public classes in a package?

If you want to put two public classes in a package, have two java source files containing one public class, but keep the package name same. For example:

```
//save as A.java

package javatpoint;
public class A{}
//save as B.java

package javatpoint;
public class B{}
```

MCQS on Packages in Java

```
1. Which of these keywords is used to define packages in Java?
a) pkg
b) Pkg
c) package
d) Package

2. Which of these is a mechanism for naming and visibility control of a class and its content?
a) Object
b) Packages
c) Interfaces
d) None of the Mentioned.

3. Which of this access specifies can be used for a class so that its members can be accessed by a different clas
a) Public
b) Protected
c) No Modifier
d) All of the mentioned

4. Which of these access specifiers can be used for a class so that it’s members can be accessed by a different c
a) Public
b) Protected
c) Private
d) No Modifier

5. Which of the following is correct way of importing an entire package ‘pkg’?
a) import pkg.
b) Import pkg.
c) import pkg.*
d) Import pkg.*

6. Which of the following is incorrect statement about packages?
a) Package defines a namespace in which classes are stored.
b) A package can contain other package within it.
c) Java uses file system directories to store packages.
d) A package can be renamed without renaming the directory in which the classes are stored.

7. Which of the following package stores all the standard java classes?
a) lang
b) java
```

```
c) util
d) java.packages

8. What is the output of this program?

    package pkg;
    class display {
        int x;
        void show() {
            if (x > 1)
                System.out.print(x + " ");
        }
    }
    class packages {
        public static void main(String args[]) {
            display[] arr=new display[3];
            for(int i=0;i<3;i++)
                arr[i]=new display();
            arr[0].x = 0;
            arr[1].x = 1;
            arr[2].x = 2;
            for (int i = 0; i < 3; ++i)
                arr[i].show();
         }
    }
Note : packages.class file is in directory pkg;
a) 0
b) 1
c) 2
d) 0 1 2


9. What is the output of this program?

    package pkg;
    class output {
        public static void main(String args[])
        {
            StringBuffer s1 = new StringBuffer("Hello");
            s1.setCharAt(1, x);
            System.out.println(s1);
        }
    }
a) xello
b) xxxxx
c) Hxllo
d) Hexlo


10. What is the output of this program?

    package pkg;
    class output {
        public static void main(String args[])
        {
            StringBuffer s1 = new StringBuffer("Hello World");
            s1.insert(6 , "Good ");
            System.out.println(s1);
        }
   }
Note : Output.class file is not in directory pkg.
a) HelloGoodWorld
b) HellGoodoWorld
c) Compilation error
d) Runtime error
```

Discussion points in Inheritance and

Interface in Java

```
1.what is inheritance?
inheritance is one of the oops concepts in java.inheritance is concept of  getting properties of one class object
Inheritance represents the IS-A relationship,also known as parent-child relationship.

2.what are the types of inheritance?
1.Multiple inheritance( java doesn't support multiple inheritance).
2.Multilevel inheritance.

3.How Inheritance can be implemented in java?
Inheritance can be implemented in JAVA using below two keywords:
1.extends
2.implements
extends is used for developing inheritance between two classes and two interfaces.
implements keyword is used to developed inheritance between interface and class.



4.Why we need to use Inheritance?
1.For Code Re usability.
2.For Method Overriding.

5.what is syntax of inheritance?
public class subclass extends superclass{
//all methods and variables declare here
}

6.what is multilevel inheritance?
Getting the properties from one class object to another class object level wise with different priorities.
Check here for more points on Inheritance

7.what is Multiple inheritance?why Java Doesn't Support multiple Inheritance.
The concept of Getting the properties from multiple class objects to sub class object with same priorities is kno
In multiple inheritance there is every chance of multiple properties of multiple objects with  the same name ava
Because of multiple inheritance there is chance of the root object getting created more than once.
Always the root object i.e object of object class hast to be created only once.
Because of above mentioned reasons multiple inheritance would not be supported by java.
Thus in java a class can not extend more than one class simultaneously. At most a class can extend only one class
Check here for more points on Why java does not supports multiple inheritance

8.How do you implement multiple inheritance in java?
Using interfaces java can support multiple inheritance concept in java. in java can not extend more than one clas
Program:
interface A{

}
interface B{
}
class C extends interface A,B{
}

9.Can a class extend itself?
No,A class can't extend itself.


10.What happens if super class and sub class having same field name?
Super class field will be hidden in the sub class. You can access hidden super class field in sub class using sup
```

Java Abstract Class and Interface

Interview Questions

What is the difference between Abstract class and Interface
Or
When should you use an abstract class, when an interface, when both?
Or
What is similarities/difference between an Abstract class and Interface?
Or
What is the difference between interface and an abstract class?
1. Abstract class is a class which contain one or more abstract methods, which has to be implemented by sub class

2. Abstract class definition begins with the keyword "abstract" keyword followed by Class definition. An Interfac

3. Abstract classes are useful in a situation when some general methods should be implemented and specialization

4. All variables in an Interface are by default - public static final while an abstract class can have instance v

5. An interface is also used in situations when a class needs to extend an other class apart from the abstract cl

6. An Interface can only have public members whereas an abstract class can contain private as well as protected m

7. A class implementing an interface must implement all of the methods defined in the interface, while a class ex

8. The problem with an interface is, if you want to add a new feature (method) in its contract, then you MUST imp

9. Interfaces are slow as it requires extra indirection to to find corresponding method in in the actual class. A

10.Interfaces are often used to describe the peripheral abilities of a class, and not its central identity, E.g.
implement the Recyclable interface, which could apply to many otherwise totally unrelated objects.

Note: There is no difference between a fully abstract class (all methods declared as abstract and all fields are

Note: If the various objects are all of-a-kind, and share a common state and behavior, then tend towards a commor
share is a set of method signatures, then tend towards an interface.

Similarities:
Neither Abstract classes nor Interface can be instantiated.

What does it mean that a method or class is abstract?

An abstract class cannot be instantiated. Only its subclasses can be instantiated. A class that has one or more a

    public abstract class AbstractClass

Abstract classes may contain abstract methods. A method declared abstract is not actually implemented in the clas
or itself be declared abstract. Only the method’s prototype is provided in the class definition. Also, a final me
. It has no body. For example,

public abstract float getInfo()

What must a class do to implement an interface?

The class must provide all of the methods in the interface and identify the interface in its implements clause.

What is an abstract method?

An abstract method is a method whose implementation is deferred to a subclass.

What is interface? How to support multiple inhertance in Java?

Or

What is a cloneable interface and how many methods does it contain?

An Interface are implicitly abstract and public. Interfaces with empty bodies are called marker interfaces having

```
Interfaces provide support for multiple inheritance in Java. A class that implements the interfaces is bound to i
Example of Interface:
public interface sampleInterface {
public void functionOne();

public long CONSTANT_ONE = 1000;
}

What is an abstract class?
Or
Can you make an instance of an abstract class?

Abstract classes can contain abstract and concrete methods. Abstract classes cannot be instantiated directly i.e

Example of Abstract class:

abstract class AbstractClassExample{

protected String name;
public String getname() {
return name;
}
public abstract void function();
}

Example: Vehicle is an abstract class and Bus Truck, car etc are specific implementations

No! You cannot make an instance of an abstract class. An abstract class has to be sub-classed.
If you have an abstract class and you want to use a method which has been implemented, you may
need to subclass that abstract class, instantiate your subclass and then call that method.

What is meant by "Abstract Interface"?

Firstly, an interface is abstract. That means you cannot have any implementation in an interface.
All the methods declared in an interface are abstract methods or signatures of the methods.

How to define an Interface?

In Java Interface defines the methods but does not implement them. Interface can include constants.
A class that implements the interfaces is bound to implement all the methods defined in Interface.
Example of Interface:

public interface SampleInterface {
public void functionOne();

public long CONSTANT_ONE = 1000;
}

Can Abstract Class have constructors? Can interfaces have constructors?

Abstract class's can have a constructor, but you cannot access it through the object, since you cannot instantiat

Example
public abstract class AbstractExample {
public AbstractExample(){
System.out.println("In AbstractExample()");
}
}

public class Test extends AbstractExample{
public static void main(String args[]){
Test obj=new Test();
}
}

If interface & abstract class have same methods and those methods contain no implementation, which one would you

Obviously one should ideally go for an interface, as we can only extend one class. Implementing an interface for
```

Retrieved from "http://wiki2015.msitprogram.net/index.php?title=OopJava15:M14&oldid=910"