BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on Package in Java.

Java Package:-

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

A Package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

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.

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

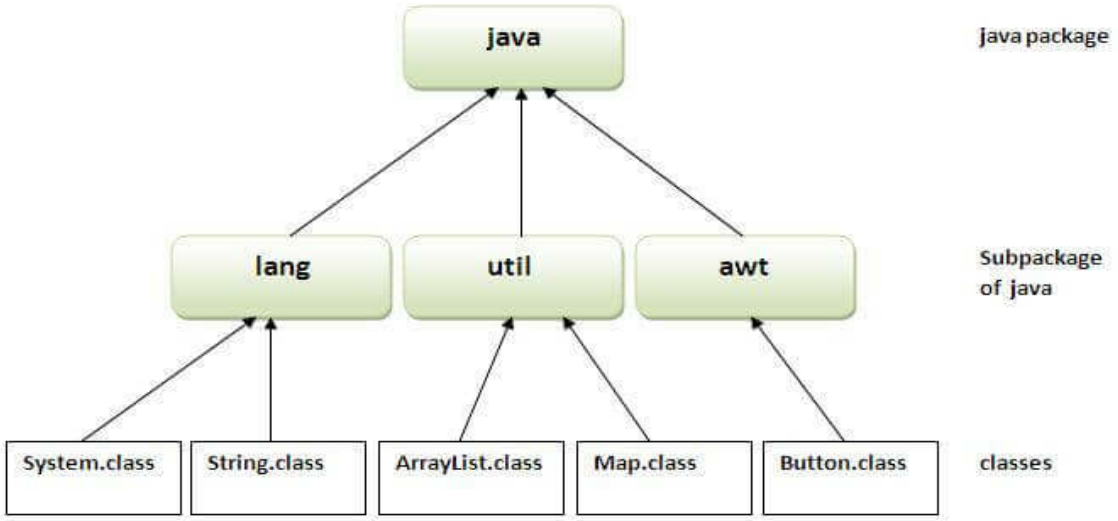
Here, we will have the detailed learning of creating and using user-defined packages.

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.



Example of create package in java:-

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

Compile java package:-

Syntax:-

javac -d directory javafilename

Here, The -d switch specifies the destination where to put the generated class file i.e. it represents destination. 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). The . represents the current folder.

Example:-

javac -d . Simple.java

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

How to access package from another package?

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

1. import package.*; (Using packagename.*)
2. import package.classname;
3. fully qualified name.

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

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

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

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.

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

Hiding Classes:-

When we import a package within a program, only the classes declared as public in that package will be made accessible within this program. In other words, the classes not declared as public in that package will not be accessible within this program.

When we import a package using astric(*), all public classes are imported. However, we may preffer to “not import” certain classes. i.e, we may like to hide these classes from accessing from outside of the package such classes should be declared”not public”.