PACKAGE

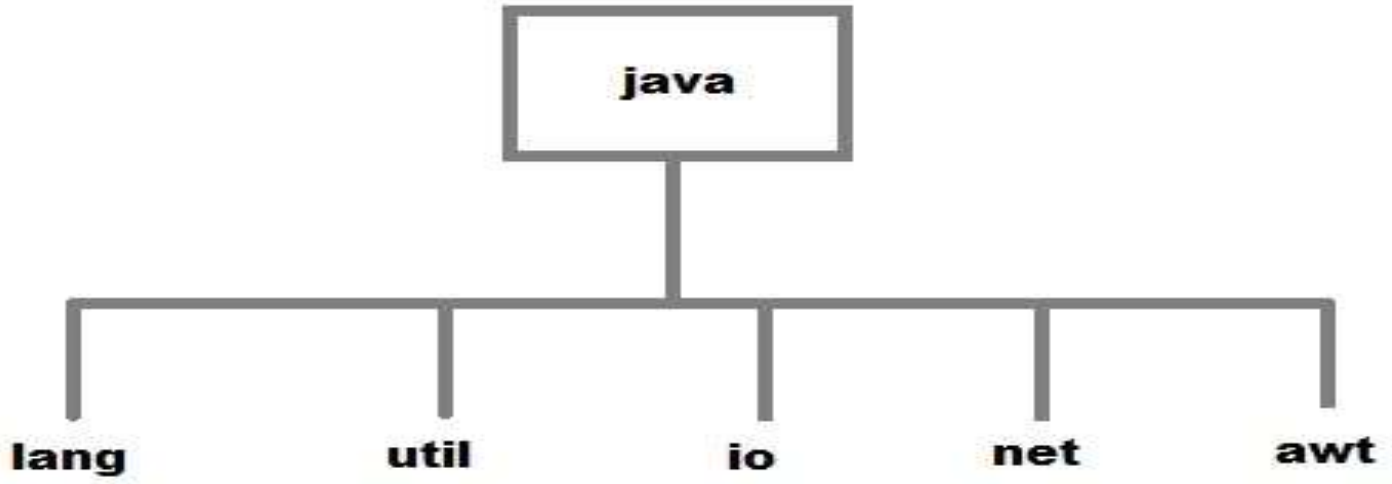
Prepared by

Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

- Package are used in Java, in-order to avoid name conflicts and to control access of class, interface and enumeration etc.
- A package can be defined as a group of similar types of classes, interface, enumeration and sub-package.
- Using package it becomes easier to locate the related classes.

Category of Package

- Built-in Package:-Existing Java package for example java.lang, java.util etc.
- User-defined-package:- Java package created by user to categorized classes and interface



Creating a package

- Creating a package in java is quite easy. Simply include a package command followed by name of the package as the first statement in java source file.

```
package mypack;
public class employee
{
…statement;
 }
```

- The above statement create a package called **mypack**.
- Java uses file system directory to store package. For example the .class for any classes you to define to be part of **mypack** package must be stored in a directory called mypack.

Example of package creation

```
package mypack
class Book {
String bookname;
String author;
Book(String b, String c) {
this.bookname = b;
this.author = c;
 }
public void show() {
System.out.println(bookname
+" "+ author);
 }
}
```

```
class test {
public static void main(String[] args) {
Book bk = new
Book("java","Herbert");
bk.show();
}
 }
```

To run this program :

- create a directory under your current working development directory(i.e. JDK directory), name it as **mypack**.
- compile the source file
- Put the class file into the directory you have created.
- Execute the program from development directory.

Uses of java package

Package is a way to organize files in java, it is used when a project consists of multiple modules. It also helps resolve naming conflicts. Package's access level also allows you to protect data from being used by the non-authorized classes.

import keyword

- **import** keyword is used to import built-in and user-defined packages into your java source file. So that your class can refer to a class that is in another package by directly using its name.
- There are 3 different ways to refer to class that is present in different package
- **Using fully qualified name** (But this is not a good practice.)
  Example :

```
class MyDate extends java.util.Date
{
  //statement;
}
```

import keyword

import the only class you want to use.

Example :

```
import java.util.Date; class MyDate extends Date
{
   //statement.;
}
```

import keyword

import all the classes from the particular package

Example :

```
import java.util.*;
class MyDate extends Date
{
  //statement;
}
```

Static import

- **static import** is a feature that expands the capabilities of **import** keyword. It is used to import **static** member of a class. We all know that static member are referred in association with its class name outside the class.
- Using **static import**, it is possible to refer to the static member directly without its class name. There are two general form of static import statement.

Static import

- The first form of **static import** statement, import only a single static member of a class
- **Syntax**

import static ***package.class-name.static-member-name;***

- **Example**

import static java.lang.Math.sqrt; //importing static method **sqrt** of **Math** class

Example using static import

```
import static java.lang.Math.*;
public class Test
{
public static void main(String[] args) {
System.out.println(sqrt(144));
}
}
```

Output: 12

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.

A COMPLETE EXAMPLE OF PACKAGE

```
//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

Syntax:

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 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

- **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.

Using 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();
}
}
```

O/P: Hello

Using packagename.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();
}
}
```

O/P: Hello

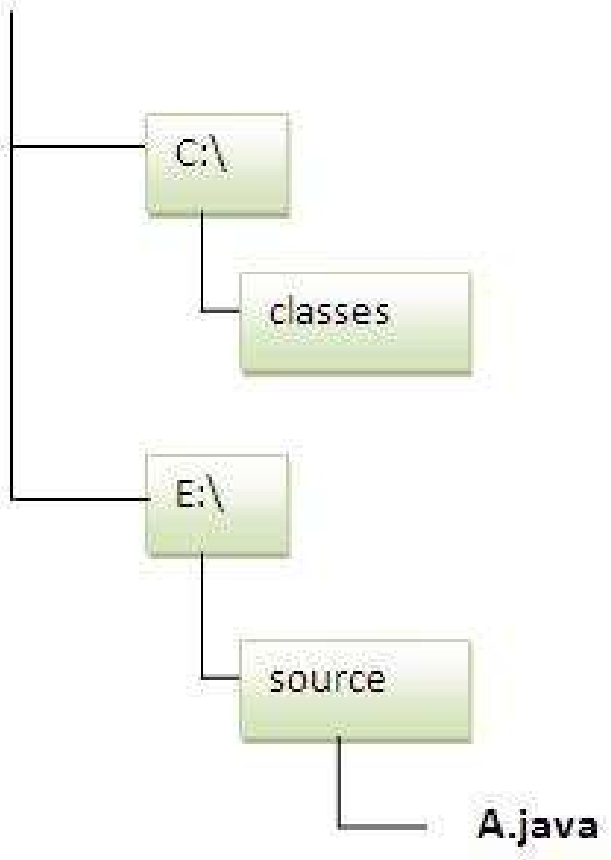
Using fully qualified name

```
//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 A();
obj.msg();
}
}
```

O/P: Hello

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



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 resides.
- **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**

Thank you