Chapter Five

Java Packages

By: Sinodos G

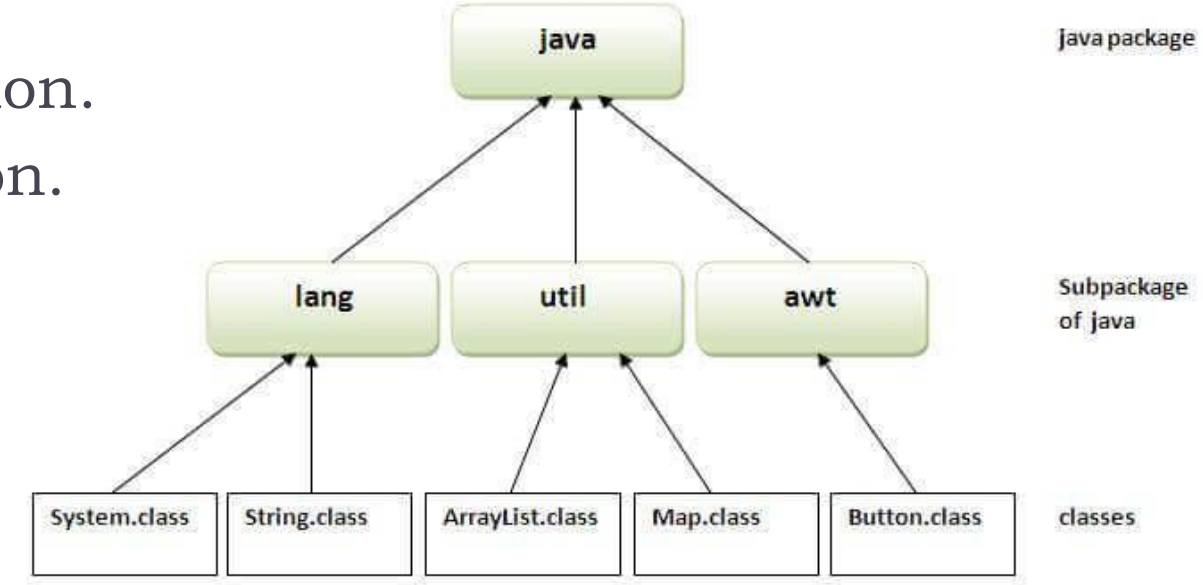
Introduction

- A **java package** is a group of similar types of classes, interfaces and sub-packages.
- Packages are used in Java in order to:-
  - prevent naming conflicts,
  - to control access,
  - to make searching/locating
  - usage of classes, interfaces, enumerations and annotations easier, etc.
- Defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

Cont'd ...

- 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.
  - **java.lang** – bundles the fundamental classes
  - **java.io** – classes for input , output functions are bundled in this package
  - Java.awet
  - Javax.swing

- Programmers can define their own packages to bundle group of classes/interfaces, etc.
- **Advantage of Java Package**
  - used to categorize the classes and interfaces so that they can be easily maintained.
  - provides access protection.
  - removes naming collision.



Creating a Package

- While creating a package, you should choose a name for the package and include a **package** statement along with that name at the top of every source file that contains:-
  - the classes, interfaces, enumerations, and annotation types that you want to include in the package.
- The package statement should be the first line in the source file.
- Only one package statement in each source file, and it applies to all types in the file.
- If a package statement is not used then the class, interfaces, enumerations, and annotation types will be placed in the current default package

- **Simple example of java package**
- The **package keyword** is used to create a package in java.

```
//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

```
/* File name : Animal.java */
package animals;

interface Animal {
   public void eat();
   public void travel();
}
```

Access Package

- If a class wants to use another class in the same package, the package name need not be used.
- Classes in the same package find each other without any special syntax.
- There are three ways to access the package from outside the package.
  - import package.*;
  - import package.classname;
  - 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.

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

- **Using packagename.classname**
- If you import package.classname then only declared class of this package will be accessible.

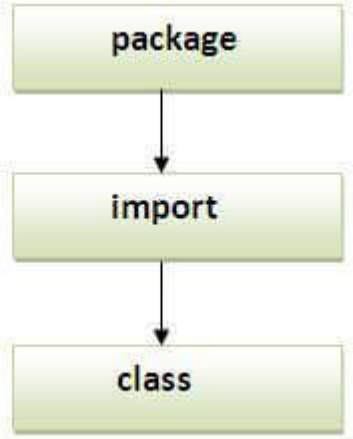
```
//save by A.java

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();
     }
   }
```

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

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



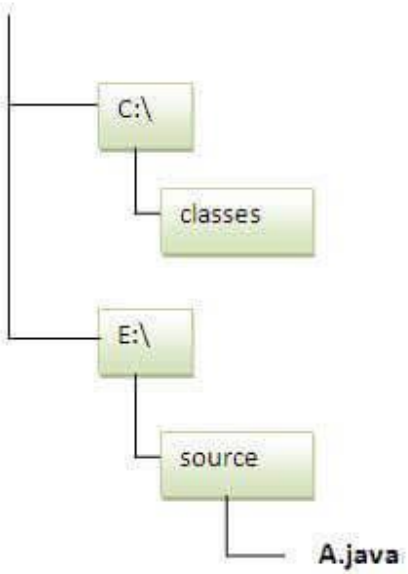
Note: If you import a package, sub-packages will not be imported.

Subpackage

- Package inside the package is called the **subpackage**.
- 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. ***Example:-***
  - Reader and Writer classes are for Input/Output operation,
  - Socket and ServerSocket classes are for networking.
  - java package into subpackages such as lang, net, io etc.
    - 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**
- **Example:-**
  - **com.ddu**

- **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:



- **The Directory Structure of Packages**
- Two major results occur when a class is placed in a package –
  - The name of the package becomes a part of the name of the class
  - The name of the package must match the directory structure where the corresponding bytecode resides.

    - ***....\vehicle\Car.java***

- In general, a company uses its reversed Internet domain name for its package names.
  - Example − A company's Internet domain name is apple.com, then all its package names would start with com.apple. Each component of the package name corresponds to a subdirectory.
  - Example − The company had a com.apple.computers package that contained a Dell.java source file, it would be contained in a series of subdirectories like this −
  - ….\com\apple\computers\Dell.java

```
// File Name: Dell.java
package com.apple.computers;

public class Dell {
}

class Ups {
}
```

Question

End of Chapter 5

Next → Chapter 6