UNIT-IV

Inheritance, Packages & Interfaces

Abstract

- Introduction of Package
- Create package
- Import keyword
- Accessing rules for package

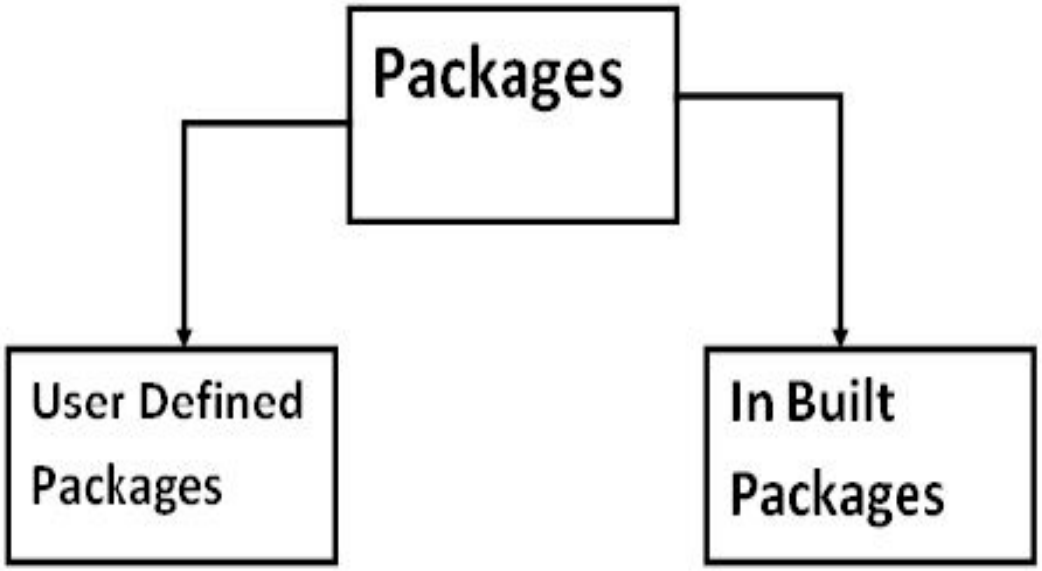
Package

- Packages are used in Java in order to **prevent naming conflicts**, to **control access**, to make **searching/locating and usage of classes**, **interfaces, enumerations and annotations easier, etc.**

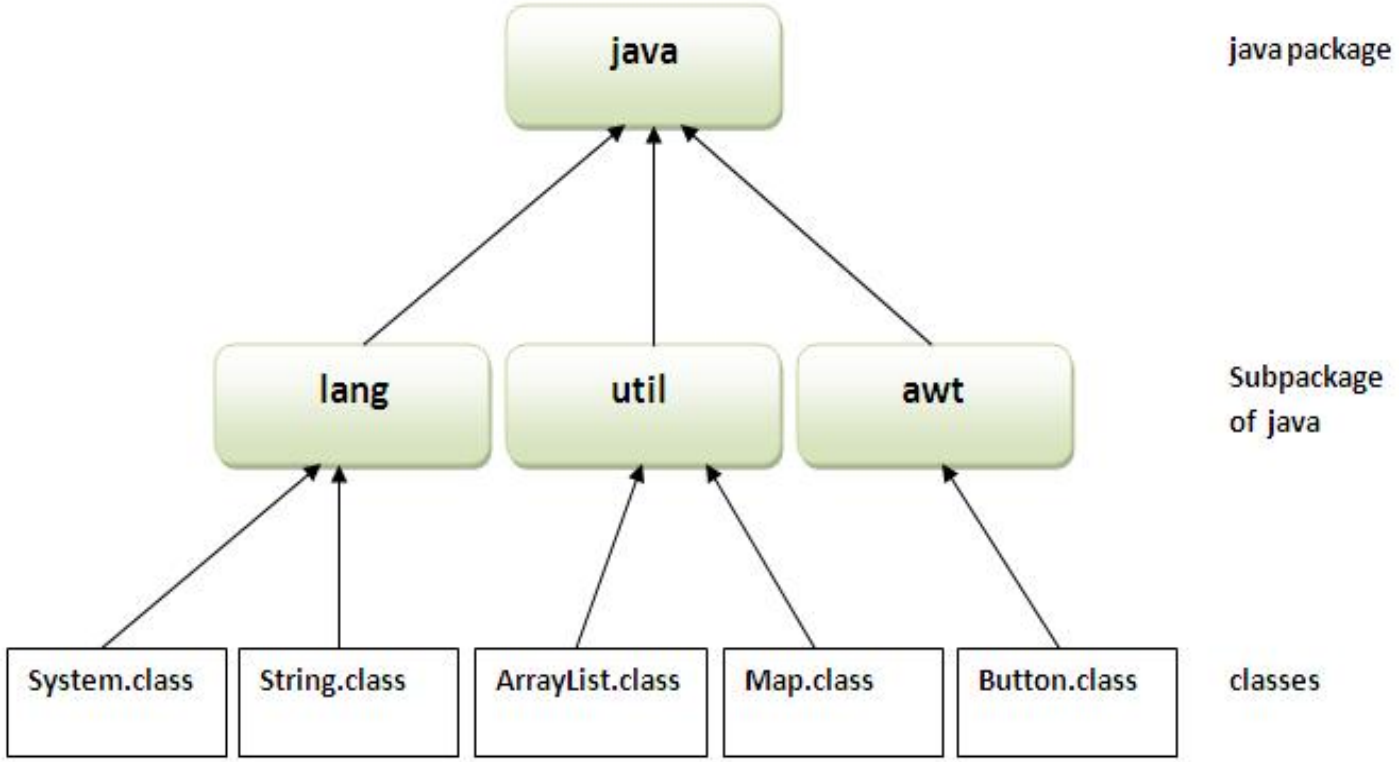
- A **Package** can be defined as a **grouping of related types (classes, interfaces, enumerations and annotations) providing access protection** and **namespace management.**

Package

- Basically, there are **2 types** of packages in JAVA.



In Built Package



Package

- **Advantage :**
  - Java package is used to **categorize** the classes and interfaces so that they can be easily maintained.
  - Java package **provides access protection.**
  - Java package **removes naming collision.**

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

Package

- **Creating package Syntax:**

  package package_name;

- **Example:**

  package employee;

Package

```
package first;

public class Simple // simple.java

{

     public static void main(String args[])

     {

    System.out.println(“Package is created");

   }

}
```

Package

- **Compile java package:**

  javac -d Destination_folder file_name.java

- **After compile package,**
  - The folder with the given package name is created in the specified destination,
  - the compiled class files will be placed in that folder.

Package

- **Compile :** javac -d . Simple.java
- **Run :** java first.Simple
- **Output:** Package is created

- **Note:**
  - The -d is a switch that tells the compiler where to put the class file. It represents destination.
  - The . represents the current folder.

‘import’ keyword

- If a class wants to use another class in the same package, then 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 packagename.*;**
  - **import packagename.classname;**
  - **fully qualified name.**

‘import’ keyword

- **Using packagename.* :**
  - If you use **packagename.*** then all the **classes and interfaces** of this package will be accessible but **not subpackages**.
  - **Syntax:** import packagename.*;

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

‘import’ keyword

- **Using fully qualified name :**
  - If you use **fully qualified name** then only **declared class** of this package will be accessible. So, no need to import the package.
  - 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.**

    Ex: java.util and java.sql packages contain **Date** class.

Access rules for packages

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes