FAQ's on Packages

1. What is a package?

Package is a grouping mechanism in which related class files are grouped and made available to other applications and other parts of the same application. So, the package is a collection of related classes and interfaces.

The package declaration should be the first statement in a Java class.

2. What is a Java package and how is it used?

A Java package is a naming context for classes and interfaces. A package is used to create a separate name space for groups of classes and interfaces. Packages are also used to organize related classes and interfaces into a single API unit and to control accessibility to these classes and interfaces.

3. What is the first keyword used in a Java application development?

The '**Package**' is the first keyword used in a Java Application development.

4. Which package is imported, by default?

By default, the **'Java.lang'** package is imported, even without a package declaration.

5. Can a class declared as private be accessed outside its package?

No. once declared as private, it is not possible to access the class outside its package.

6. What is the naming convention to be followed for declaring a user-defined package in Java application development?

The **Reverse Domain** naming convention is to be followed for declaring a user-defined package in Java Application development. For this the syntax is:

```
com.companyname.projectname.module.submodule;
```

Example: `com.apache.struts.tiles.login;`

7. What are the types of packages in Java?

There are two types of packages in Java

- Standard or Built-in packages and
- User-defined packages.

8. What is standard package in Java?

The standard package is the package that contains all libraries or pre-defined class files. There are two types of standard packages:

- Core packages (which are starts with Java)
- Extension packages (Which are starts with Javax)

Example:

For Core packages	For Extension packages
• Java.lang	Javax.sql
• Java.util	Javax.servlet
• Java.awt	Javax.servlet.http
• Java.applet	Javax.servlet.jsp
• Java.io	Javax.ejb
• Java.net	Javax.swing

9. Does importing a package make sub-packages class files available to the application?

No. Importing a package does not make sub-packages class files available to the application.

Example:

```
     import Java.awt;
     class TestPackage
     {
        Button b;
        ActionEvent ae;
/*it is an error bcoz.. ActionEvent is available in
Java.awt.event.So we have to import Java.awt.event.*; also*/

        Public static void main(String args[])
         {
           System.out.println("hello packages");
         }
      }
```

10. What is a user-defined package in Java?

The packages which are created by the users in the Java application development are known as “user-defined package”.

11. How to create a user-defined package?

The ‘**Package**’ keyword is used for creating a package in Java. Its syntax is:

```
package <fullyqualifiedpackagename>;
```

As per industrial standards, the package name should follow the reverse domain naming convention. Example:

```
package org.companyname.projectname.modulename.submodule;
package  org.talentsprint.hrms.leavemangement;
package: org.talentsprint.pacakgesprograms.userdefinedpackges;
class Employee{
        int empId=30039
        String empName=”smith”;
        float empSal=15000;
        public void getEmpDetails()
                    {
                        System.out.println(“employe id: “
                        +empId);
                        System.out.println(“employe name: “
                        +empId);
                        System.out.println(“employe salary: “
                        +empId);
                    }
                }
```

12. How to compile a source code of Java that is created as package?

The command for compiling source code with special syntax is:

```
Javac –d .  FileName.Java
```

Example: `Javac –d . Employe.Java`

13. What happens at background when "Javac –d. Employe.Java "is executed?

When this command executed the following things happens:

1. org directory is created in the current working directory (or default package).
2. `Employe.class` file is stored in: `org.talentsprint.pacakgesprograms.userdefinedpackges`

14. What happens if we develop another Java class (called Department) and compile the class with the same package name?Does it create new package with the same name?

No, instead it will store the second.class file in the same package directory.
Example:

```
Package org.talentsprint.pacakgesprograms.userdefinedpackges;
        class Department{
           int deptId=1234;
           Public void getDeparmentId()
           {
              return deptId;
           }
         }
```

After compilation it will store the `Department.class` file into the package: `org.talentsprint.pacakgesprograms.userdefinedpackges;`

15. How to use user-defined packages in Java?

There are two fundamental steps to access and use the user-defined packages:

Step-1: Set the classpath to user-defined package ,i.e..in which directory user-defined package is stored up to that directory set the class path.

Example: `set classpath = .;`**E:\talentstsprint\myprograms**

(Assume that our programs are created in the above path)

Step-2: Using import keyword we can import all the specific user-defined and standard packages also.

Example:

```
import org.talentsprint.pacakgesprograms.userdefinedpackges;
class RunProgram{
     Public static void main(String args[]){
```

```
            Employe e1=new Employe();
            System.out.println(e1.showEmpDetails();
            Department d1=new Department();
            System.out.println(d1.getDeptId());
        }
    }
```

16. Why do we use user-defined packages?

We can use user-defined packages to:

- Group related class files into a separate name space.
- Provide same level of security
- Make applications available to other parts of the application and in same application also.

17. Can a source file contain more than one class declaration?

Yes. A single source file can contain any number of class declarations, but only one of the classes can be declared as public.

18. What restrictions are placed on the location of a package statement within a source code file?

A package statement must appear as the first line in a source code file (excluding blank lines and comments).

19. What are the practical benefits, if any, of importing a specific class rather than an entire package (e.g. import Java.net.* versus import Java.net.Socket)?

It makes no difference in the generated class files, since only the classes that are actually used are referenced by the generated class file. The practical benefit of importing single classes can be realized when two (or more) packages with the same class name, like, Java.util.Timer and Javax.swing.Timer. If you import Java.util.* and Javax.swing.* and then try to use "Timer", you will get an error while compiling (the class name is ambiguous between both packages). Let's say what you really wanted was the Javax.swing.Timer class, and the only classes you plan on using in Java.util are Collection and HashMap. In this case, some people will prefer to import Java.util.Collection and import Java.util.HashMap instead of importing Java.util.*. This will now allow them to use Timer, Collection, HashMap, and other Javax.swing classes without using fully qualified class names in.

20. Can I import same package/class twice? Will the JVM load the package twice at runtime?

One can import the same package or same class multiple times. Neither compiler nor JVM complains about it. JVM will internally load the class only once no matter how many times you import the same class.

21. Are the imports checked for validity at compile time, like: will the code containing an import such as Java.lang.ABCD be compiled?

Yes the imports are checked for the semantic validity at compile time. The code containing **Java.lang.ABCD** will not compile. It will throw an error saying, cannot resolve symbol:

- symbol : class ABCD
- location: package io
- import Java.io.ABCD;

22. Does importing a package imports the subpackages as well? Example: Does importing com.MyTest.* also import com.MyTest.UnitTests.*?

No you will have to import the subpackages explicitly. Importing com.MyTest.* will import classes in the package MyTest only. It will not import any class in any of its subpackage.

23. Do I need to import Java.lang package any time? If yes, then why?

No. It is by default loaded internally by the JVM.

24. What is a reflection package?

The java.lang reflect package has the ability to analyze itself in runtime.