Working with Packages

- Package is collection of types , Where type are referred as classes, Abstract class ,interface and sub packages
- In simple words package is the collection of .class files
  - Advantages of packages :
    - Reusability of types
    - Providing Access restriction on the types using Access Modifiers [private | default | protected | public]
- Packages are classified into 2 types
  - Pre-defined packages
    - Packages which are provided by JAVA Compiler [JDK software]
      - java.lang | java.io | java.util …..
  - User defined Packages
    - The Packages which are defined by us for our application requirements
    - Also known as customized packages

- Ways to Create Package
  - 2 Ways
  - Creating a package by taking an already defined classes
    - Create a new folder in the desired location with any name [eg : e:\java_online11\ mypack]
    - Select all required .class Files and place them into the folder we created . Then that folder is acts as a package.

Creating a package by using package keyword

```
<package> <packagename>;
[import <predefined|userdefined package>];
[modifiers] <class> <ClassName>
{
        fields
                Methods
}
[modifiers] <interface> <Interfacename>
{
        public final static fields
                public abstract methods...
}
.
.
.
```

Example:

```
//p.java

package mypack11;

class Super

{ }

abstract class Duper

{ }

interface Bumper
```

```
{ }

//javac -d  Path For Package  prgname.java

//javac -d  e:\java_online11   p.java

//javac –d c:\intel\logs p.java

//javac –d  .  p.java    [here . represents CWD]
```

- In order to use the package then we have to use "import"

```
Syn: import  <packagename>.TypeName;
Eg:   import  java.util.Scanner;
      import java.util.Date;
      import java.util.*;
```

```
//A.java
package mypack12;
public class A
{
      public void methodA()
      { System.out.println("Mtd-A"); }
}
//javac -d . A.java

//p2.java
import mypack12.A;
class B
{
      public static void main(String args[ ])
      {
            A a=new A();
```

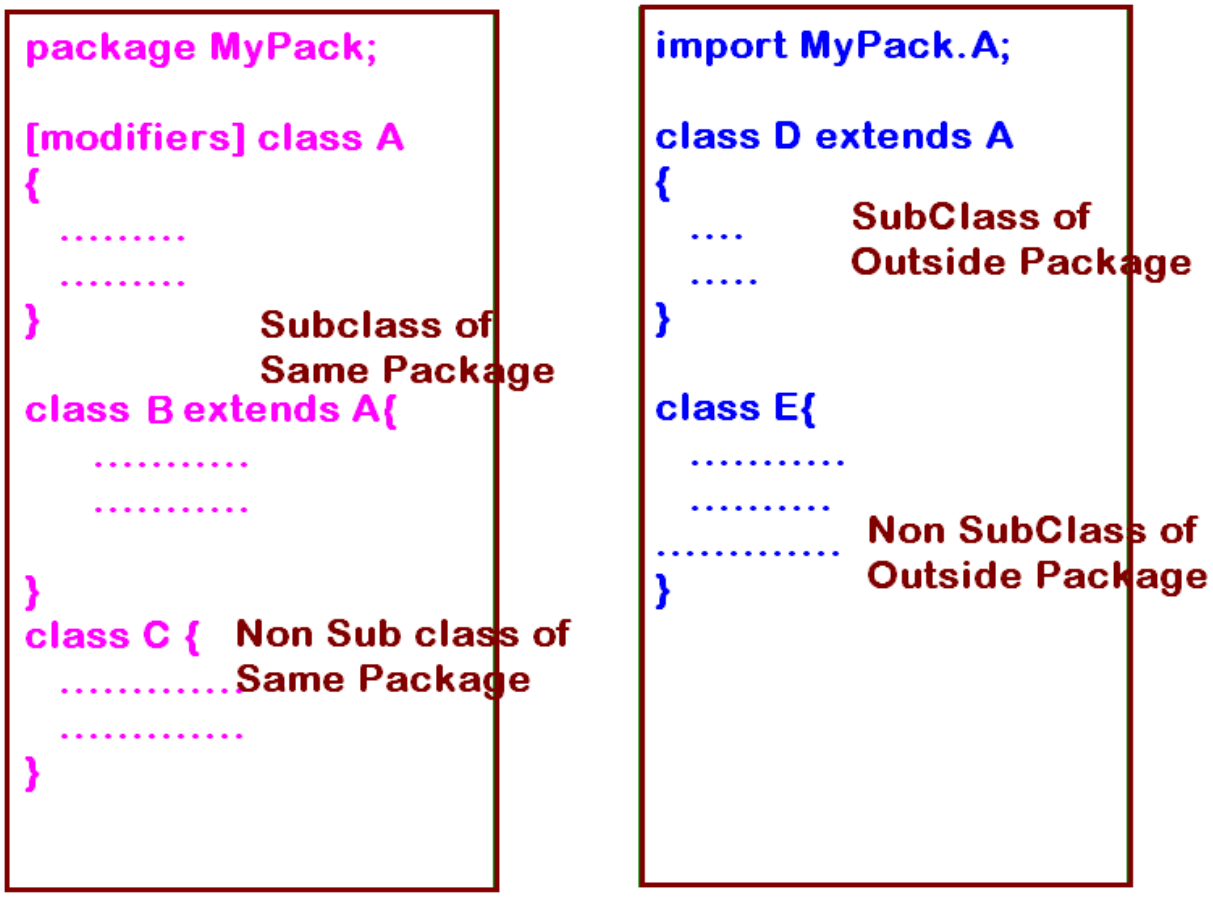
```
            a.methodA();
        }
}
```

Note:

- **If you want use any type [class | abstract class | interface] and its members from the package then all members and types must be declared as public**
- **If you define an type with "public" modifier then file name should exactly same as "TypeName"**

Table :

	Private	Default	Protected	Public
With in sub class of same package	NO	YES	YES	YES
With in non sub class of Same package	NO	YES	YES	YES
With in subclass of outside Package	NO	NO	YES	YES
With in non subclass outside package	NO	NO	NO	YES



Note:

- **For Top Level Classes we should not write private | protected | default modifiers**

Ex

```
//C.java
package mypack12;

public class C
 {
   protected void methodC( )
   { S.o.pln("Mtd-C"); }
}

//javac -d . C.java
```

```
//p3.java

import mypack12.C;
class D extends C
{
   p s v main(String args[ ])
   {
      C c=new C( );
         c.methodC( );
   }
}

//javac p3.java
```

Example:

```
//F.java
package mypack12;

class E {   //default
   void methodE( )    //default
    { S.o.pln("Mtd-E"); }
}

public class F {
   public void methodF( )
    { S.o.pln("Mtd-F"); }
}

//javac -d . F.java
```

```
import mypack12.F;

class H {
   p s v main(String args[ ]) {
     F f=new F();
        f.methodF();
   }
}
```

Example:

```
//F.java
package mypack12;

class E {   //default
   void methodE( )    //default
    { S.o.pln("Mtd-E"); }
}

public class F {
  E e=new E( );
  public void methodF()
  {   E.methodE( ); }
}

//javac -d . F.java
```

```
import mypack12.F;

class H {
   p s v main(String args[ ]) {
     F f=new F();
       f.methodF();
   }
}
```

Nested Packages : Process of defining a package inside another package

```
//IA.java
package sssit.online;
public interface IA
{       public abstract void method1(); }


//javac -d . IA.java
```

```
//Demo.java
import sssit.online.IA;

class Sub implements IA
{
	public void method1()
	{ System.out.println("OR m1 of IA"); }
}

class Testing
{
	public static void main(String args[ ])
	{
		IA ia=new Sub();
		   ia.method1();
	}
}
```