Unit V (6 Contact hours)

Packages: Creating a Package, setting CLASSPATH, Access control protection, importing packages.

What is Package in Java?

Package is a collection of related classes. Java uses package to group related classes, interfaces and sub-packages in any Java project.

We can assume package as a folder or a directory that is used to store similar files.

In Java, packages are used to avoid name conflicts and to control access of class, interface and enumeration etc. Using package it becomes easier to locate the related classes and it also provides a good structure for projects with hundreds of classes and other files.

Lets understand it by a simple example, Suppose, we have some math related classes and interfaces then to collect them into a simple place, we have to create a package.

Each package in Java has its unique name and organizes its classes and interfaces into a separate namespace, or name group.

Although interfaces and classes with the same name cannot appear in the same package, they can appear in different packages. This is possible by assigning a separate namespace to each java package.

A **java package** is a group of similar types of classes, interfaces and sub-packages.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.

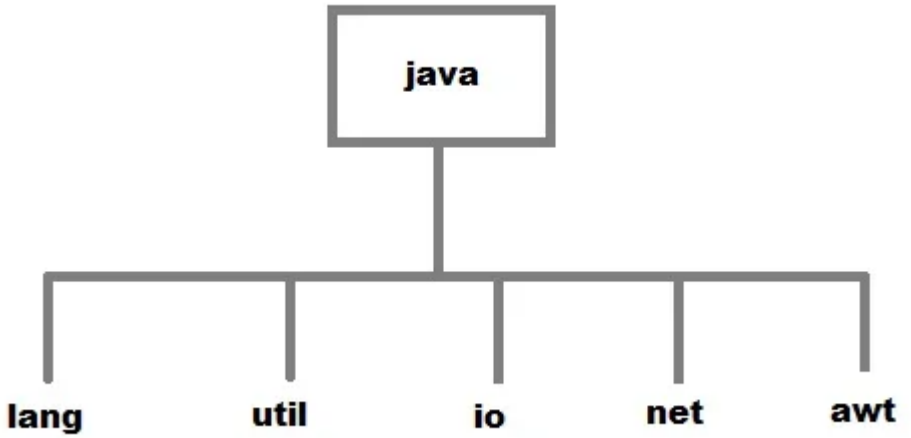
Syntax:-

```
package nameOfPackage;
```

Types Of Java Package

Package can be built-in and user-defined, Java provides rich set of built-in packages in form of API that stores related classes and sub-packages.

- **Built-in Package:** math, util, lang, i/o etc are the example of built-in packages.
- **User-defined-package:** Java package created by user to categorize their project's classes and interface are known as user-defined packages.



How to Create 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

{

    String empId;

    String name;

}
```

The above statement will create a package woth name **mypack** in the project directory.

Java uses file system directories to store packages. For example the .java file for any class you define to be part of **mypack** package must be stored in a **directory** called **mypack**.

Additional points about package:

- Package statement must be first statement in the program even before the import statement.

- A package is always defined as a separate folder having the same name as the package name.
- Store all the classes in that package folder.
- All classes of the package which we wish to access outside the package must be declared public.
- All classes within the package must have the package statement as its first line.
- All classes of the package must be compiled before use.

Example of Java packages

Now lets understand package creation by an example, here we created a **leanjava** package that stores the FirstProgram class file.

```
//save as FirstProgram.java

package learnjava;

public class FirstProgram{

    public static void main(String args[]) {

        System.out.println("Welcome to package
example");

    }

}
```

How to compile Java programs inside packages?

This is just like compiling a normal java program. If you are not using any IDE, you need to follow the steps given below to successfully compile your packages:

```
javac -d . FirstProgram.java
```

The -d switch specifies the destination where to put the generated class file. You can use any directory name like **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 run the compiled class that we compiled using above command, we need to specify package name too. Use the below command to run the class file.

```
java learnjava.FirstProgram
```

After running the program, we will get "Welcome to package example" message to the console. You can tally that with print statement used in the program.

or

How to Create a package?(prgramme-demo.java)
Creating a package is a simple task as follows

- Choose the name of the package
- Include the package command as the first line of code in your Java Source File.
- The Source file contains the classes, interfaces, etc you want to include in the package
- Compile to create the Java packages

Step 1) Consider the following package program in Java:

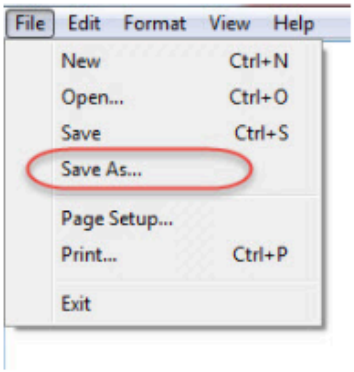
```
package p1;

class c1(){
public void m1(){
System.out.println("m1 of c1");
}
public static void main(string args[]){
```

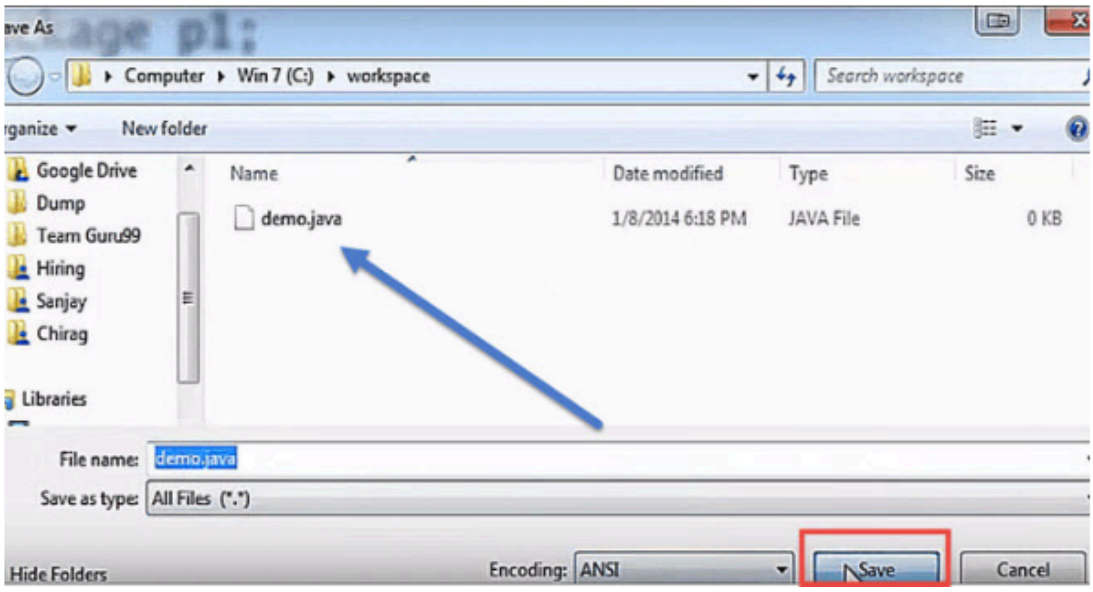
```
c1 obj = new c1();
obj.m1();
}
}
```

Here,

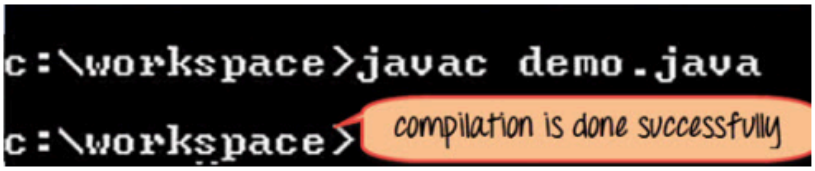
1. To put a class into a package, at the first line of code define package p1
2. Create a class c1
3. Defining a method m1 which prints a line.
4. Defining the main method
5. Creating an object of class c1
6. Calling method m1

Step 2) In next step, save this file as demo.java

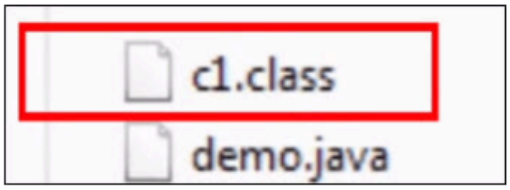




Step 3) In this step, we compile the file.



The compilation is completed. A class file c1 is created. However, no package is created? Next step has the solution

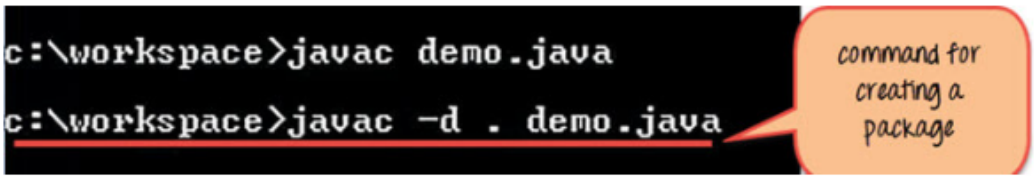


Step 4) Now we have to create a package, use the command

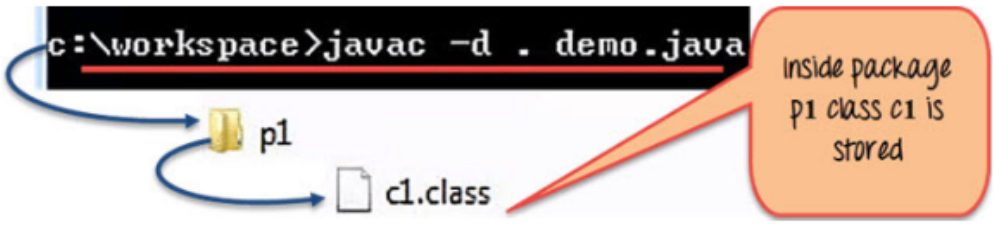
```
javac -d . demo.java
```

This command forces the compiler to create a package.

The **"."** operator represents the current working directory.



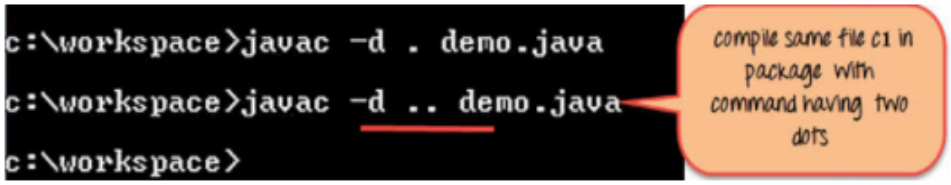
Step 5) When you execute the code, it creates a package p1. When you open the java package p1 inside you see the c1.class file.



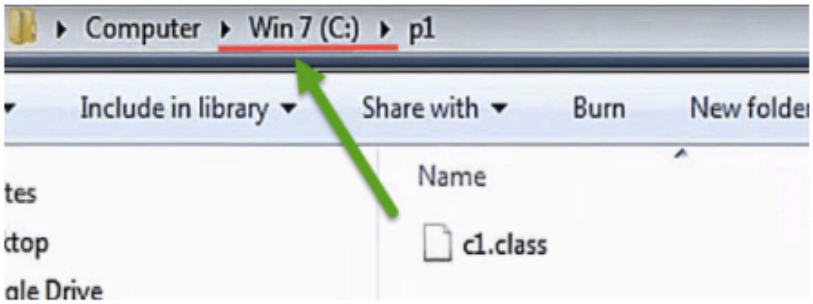
Step 6) Compile the same file using the following code

```
javac -d .. demo.java
```

Here ".." indicates the parent directory. In our case file will be saved in parent directory which is C Drive

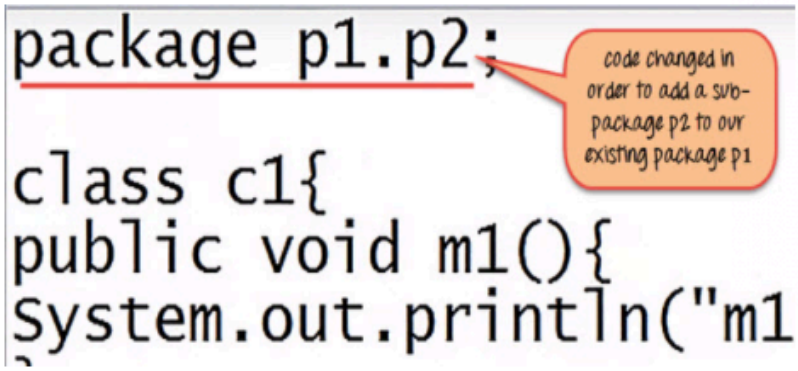


File saved in parent directory when above code is executed.

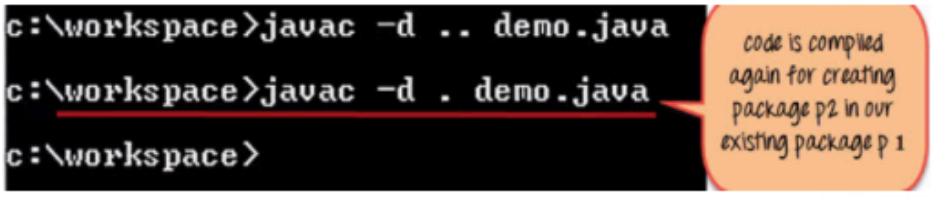


Step 7) Now let's say you want to create a sub package p2 within our existing java package p1. Then we will modify our code as

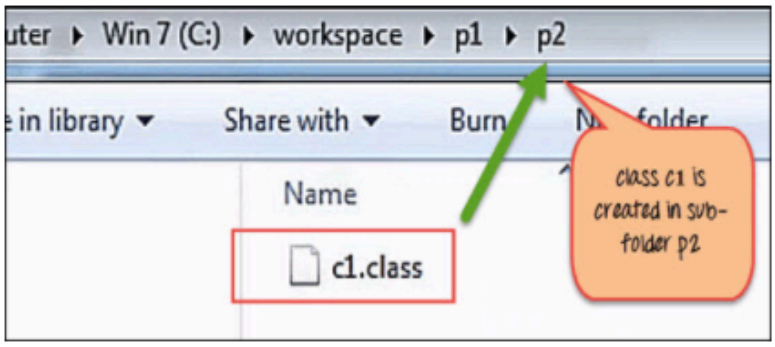
```
package p1.p2;

class c1{
public void m1() {
System.out.println("m1 of c1");
}
}
```



Step 8) Compile the file

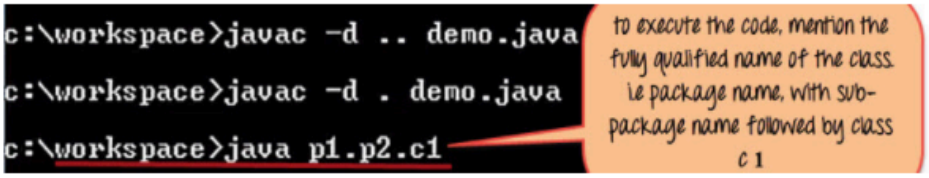


As seen in below screenshot, it creates a sub-package p2 having class c1 inside the package.

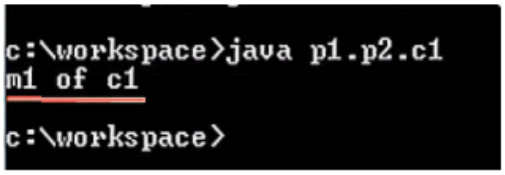


Step 9) To execute the code mention the fully qualified name of the class i.e. the package name followed by the sub-package name followed by the class name -

```
java p1.p2.c1
```



This is how the package is executed and gives the output as "m1 of c1" from the code file.



3.Import all classes of the package

If we use **packagename.* statement**, then all the classes and interfaces of this package will be accessible but the classes and interface inside the sub-packages will not be available for use.

The import keyword is used to make the classes of another package accessible to the current package.

Example :

In this example, we created a class First in **learnjava** package that access it in another class Second by using import keyword.

Example: To import package

Step 1) Copy the code into an editor.

```
package p3;
import p1.*; //imports classes only in package p1 and NOT  in the sub-package p2
class c3{
  public   void m3(){
     System.out.println("Method m3 of Class c3");
  }
  public static void main(String args[]){
    c1 obj1 = new c1();
    obj1.m1();
  }
}
```

Step 2) Save the file as Demo2.java. Compile the file using the command **javac –d . Demo2.java**

Step 3) Execute the code using the command **java p3.c3**

Packages - points to note:

- To avoid naming conflicts packages are given names of the domain name of the company in reverse Ex: com.guru99. com.microsoft, com.infosys etc.
- When a package name is not specified, a class is in the default package (the current working directory) and the package itself is given no name. Hence you were able to execute assignments earlier.
- While creating a package, care should be taken that the statement for creating package must be written before any other import statements

```
// not allowed
import package p1.*;
package p3;

//correct syntax
package p3;
import package p1.*;
```

- the *java.lang package* is imported by default for any class that you create in Java.
- The Java API is very extensive, contains classes which can perform almost all your programming tasks right from Data Structure Manipulation to Networking. More often

than not, you will be using API files in your code. You can see the API documentation here.

Packages - points to note:

- To avoid naming conflicts packages are given names of the domain name of the company in reverse Ex: com.guru99. com.microsoft, com.infosys etc.
- When a package name is not specified, a class is in the default package (the current working directory) and the package itself is given no name. Hence you were able to execute assignments earlier.
- While creating a package, care should be taken that the statement for creating package must be written before any other import statements

```
// not allowed
import package p1.*;
package p3;

//correct syntax
package p3;
import package p1.*;

References:

https://www.guru99.com/java-packages.html
```

Access protection in java packages

In java, the access modifiers define the accessibility of the class and its members. For example, private members are accessible within the same class members only. Java has four access modifiers, and they are default, private, protected, and public.

In java, the package is a container of classes, sub-classes, interfaces, and sub-packages. The class acts as a container of data and methods. So, the access modifier decides the accessibility of class members across the different packages.

In java, the accessibility of the members of a class or interface depends on its access specifiers. The following table provides information about the visibility of both data members and methods.

Access control for members of class and interface in java

Accessibility / Access Specifier	Same Class	Same Package		Other Package	
		Child class	Non-child class	Child class	Non-child class
Public	Yes	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	Yes	No
Default	Yes	Yes	Yes	No	No
Private	Yes	No	No	No	No

Notes:

The **public** members can be accessed everywhere.

The **private** members can be accessed only inside the same class.

The **protected** members are accessible to every child class (same package or other packages).

The **default** members are accessible within the same package but not outside the package.

```
class ParentClass{
        int a = 10;
        public int b = 20;
        protected int c = 30;
        private int d = 40;

        void showData() {
                System.out.println("Inside ParentClass");
                System.out.println("a = " + a);
                System.out.println("b = " + b);
                System.out.println("c = " + c);
                System.out.println("d = " + d);
        }
}

class ChildClass extends ParentClass{
```

```
        void accessData() {
                System.out.println("Inside ChildClass");
                System.out.println("a = " + a);
                System.out.println("b = " + b);
                System.out.println("c = " + c);
                //System.out.println("d = " + d);          // private member can't be accessed
        }

}
public class AccessModifiersExample {

        public static void main(String[] args) {

                ChildClass obj = new ChildClass();
                obj.showData();
                obj.accessData();

        }

}
```

```
D:\java\packages>javac -d . AccessModifiersExample.java

D:\java\packages>java AccessModifiersExample
Inside ParentClass
a = 10
b = 20
c = 30
d = 40
Inside ChildClass
a = 10
b = 20
c = 30
```

Importing Packages in java

In java, the ***import*** keyword used to import built-in and user-defined packages. When a package has imported, we can refer to all the classes of that package using their name directly.

The import statement must be after the package statement, and before any other statement.

Using an import statement, we may import a specific class or all the classes from a package.

- Using one import statement, we may import only one package or a class.
- Using an import statement, we can not import a class directly, but it must be a part of a package.
- A program may contain any number of import statements.

To import java package into a class, we need to use java **import** keyword which is used to access package and its classes into the java program.

Use import to access 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 any class that is present in a different package:

1. without import the package
2. import package with specified class
3. import package with all classes

Let's understand each one with the help of example.

1.Accessing package without import keyword

If you use fully qualified name to import any class into your program, then only that particular class of the package will be accessible in your program, other classes in the same package will not be accessible. For this approach, there is no need to use the import statement. But you will have to use the fully qualified name every time you are accessing the class or the interface. This is generally used when two packages have classes with same names. For example: java.util and java.sql packages contain Date class.

Example

S

```
//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();

    }

}
```

OUPUT:

Hello

2. Import the Specific Class

Package can have many classes but sometimes we want to access only specific class in our program in that case, Java allows us to specify class name along with package name. If we use import packagename.classname statement then only the class with name classname in the package will be available for use.

Using an importing statement, we can import a specific class. The following syntax is employed to import a specific class.

Syntax:

```
import packageName.ClassName;
```

Let's look at an import statement to import a built-in package and Scanner class.

```
package myPackage;

import java.util.Scanner;

public class ImportingExample {

        public static void main(String[] args) {

                Scanner read = new Scanner(System.in);

                int i = read.nextInt();

                System.out.println("You have entered a number " + i);
        }
}
```

In the above code, the class **ImportingExample** belongs to **myPackage** package, and it also importing a class called **Scanner** from **java.util** package.

3.Importing all the classes

Using an importing statement, we can import all the classes of a package. To import all the classes of the package, we use * symbol. The following syntax is employed to import all the classes of a package.

Syntax:

```
import packageName.*;
```

Let's look at an import statement to import a built-in package.

```
package myPackage;

import java.util.*;

public class ImportingExample {

        public static void main(String[] args) {

                Scanner read = new Scanner(System.in);

                int i = read.nextInt();

                System.out.println("You have entered a number " + i);

                Random rand = new Random();

                int num = rand.nextInt(100);

                System.out.println("Randomly generated number " + num);
        }
}
```

In the above code, the class **ImportingExample** belongs to **myPackage** package, and it also importing all the classes like Scanner, Random, Stack, Vector, ArrayList, HashSet, etc. from the **java.util** package.

- The import statement imports only classes of the package, but not sub-packages and its classes.
- We may also import sub-packages by using a symbol '.' (dot) to separate parent package and sub-package.

Consider the following import statement.

```
import java.util.*;
```

The above import statement **util** is a sub-package of **java** package. It imports all the classes of **util** package only, but not classes of **java** package.

```
References:
http://www.btechsmartclass.com/java/java-importing-packages.html
```

https://www.youtube.com/watch?v=nL5nym76j8Q