CHAPTER 6-JAVA LIBRARY, PACKAGES, USE OF IMPORT

6.1 Objective

a) Introduction to Packages and Imports
b) Understanding Java Libraries
c) More on access specifiers

6.2 Content

6.2.1 Packages

A package is a namespace that organizes a set of related classes and interfaces, providing access protection and name space management. Conceptually packages can be thought of as similar to different folders on the computer. Packages are used in Java to prevent naming conflicts, to control access, and to make searching/locating and usage of classes easier.

Java platform provides a very large library of classes and interfaces organized into different packages known as "Application Programming Interface" or API. Some of the packages in Java API Library are:

java.lang - bundles the fundamental classes

java.io - classes for input, output functions are bundled in this package

Because software written in Java can be composed of hundreds of individual classes, programmers can define their own packages to group related classes together. It is a good practice to organize by grouping related classes and interfaces into packages so that a programmer can easily locate and find the classes he needs.

Since the package creates a new namespace there won't be any class name conflicts with other class names in other packages.

Eg: Both java.awt and java.util packages have a class called "List". There is no naming conflict because the two "List" classes are part of two different packages.

Using packages, it is easier to provide access control and it is also easier to locate the related classes.

Eg: Consider a scenario where a bank has many employees. New employees are added, some may resign from bank, so those employees need to be removed from bank. Many customers come to the bank to get services. Customers open account, deposit money to account, withdraw money from account etc.

Here we have different objects and so different classes need to be created. If it is going to be a large application, it will be difficult to manage all the files without using packages. Also if we want to set different visibility, like Customer should not see the details of Employee, packages can be used to organize and set visibility using access modifiers.

6.2.2 Implementation of Packages in Java

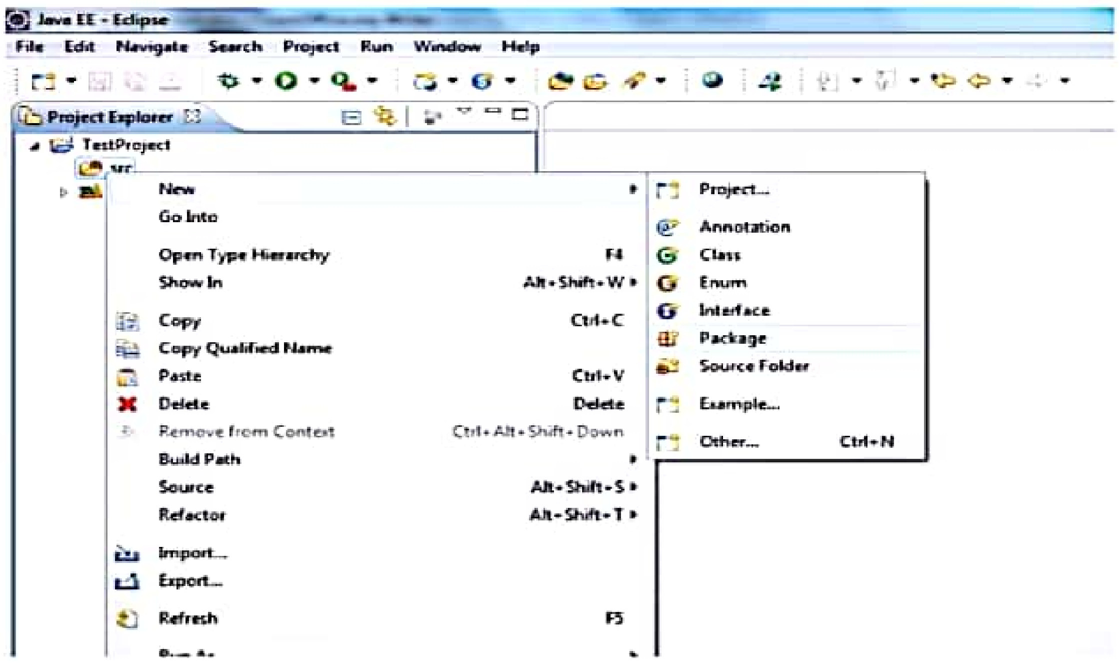
The package in Java is represented using the keyword 'package'. The package statement is written with the keyword 'package' followed by the name of package and ends with a semi colon.

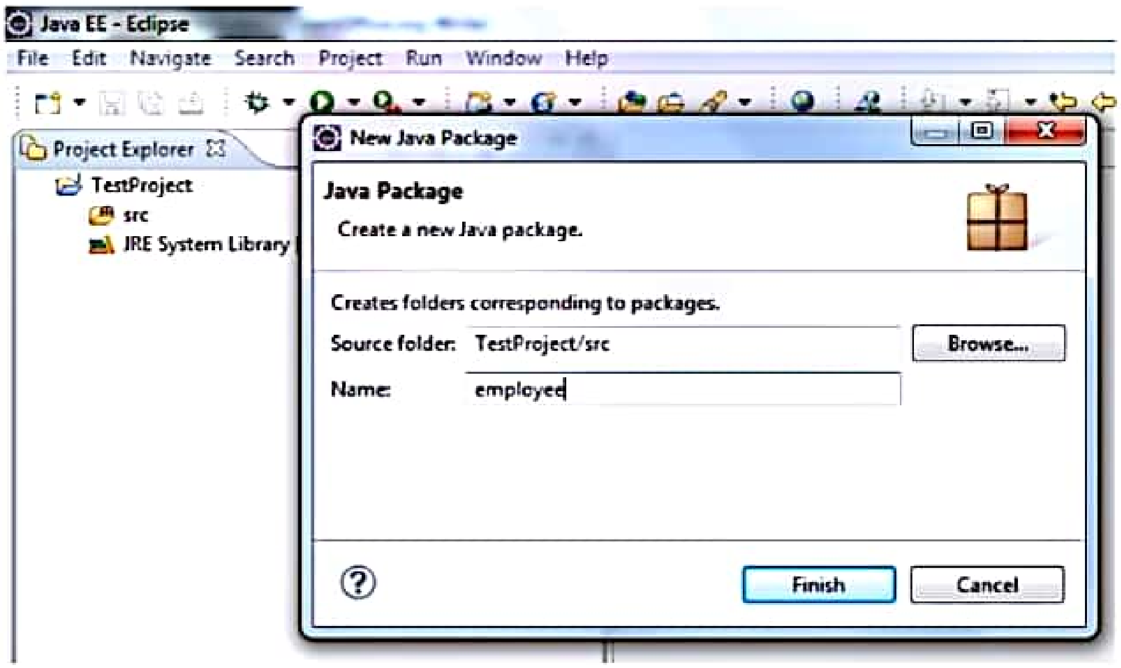
Syntax: package packagename ;
Eg:
```
package employee ;
package com.tcs.employees ;
```

The package statement should be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.

6.2.3 Creating packages in Eclipse IDE

Step1: Create a package in eclipse and provide a name for the package as shown below. Package name can be a one word name or multiple words separated by a dot.
Eg: com.tcs.employees (or) employee





Java Package

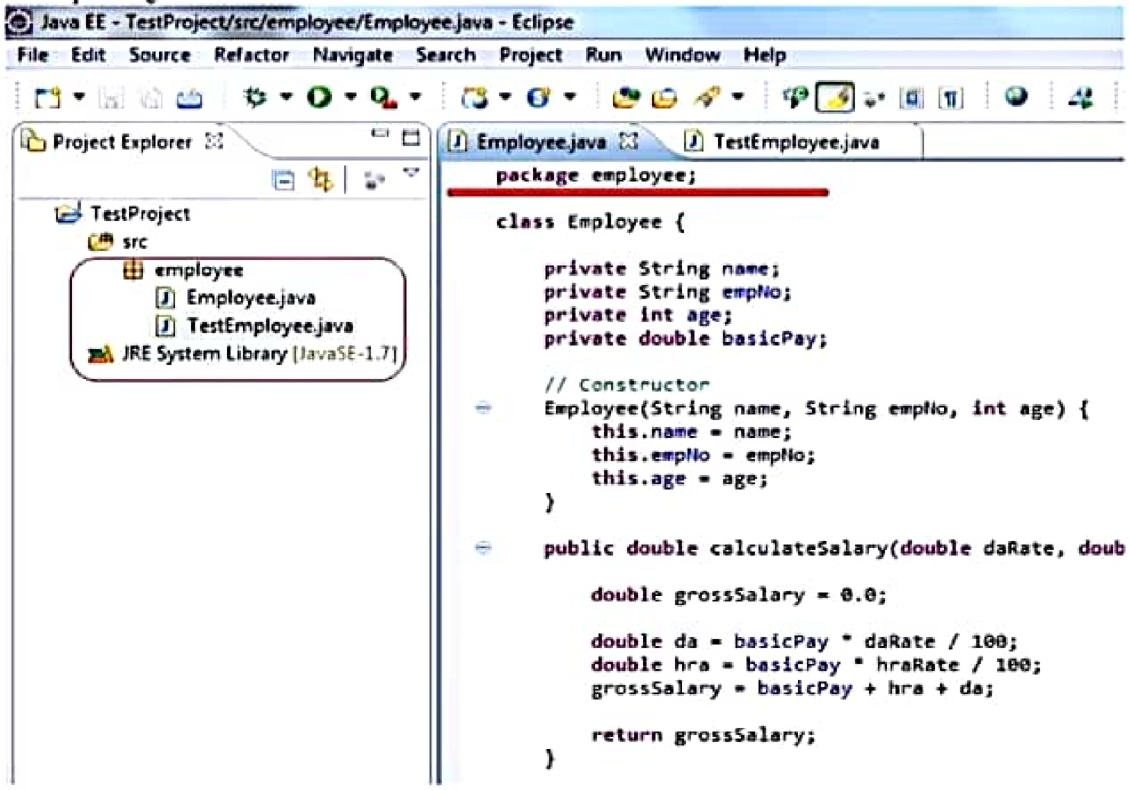
Create a new Java package.

Creates folders corresponding to packages.

Source folder: TestProject/src

Name: employee

Step 2: Create classe(s) inside the employee package. The java class file should have package statement as the first line. In the screenshot below you can see the employee package. The classes Employee.java and TestEmployee.java are created under this package. You can see that the first line in Employee.java is the package statement.



Java EE - TestProject/src/employee/Employee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
TestProject
src
employee
Employee.java
TestEmployee.java
JRE System Library [JavaSE-1.7]
Employee.java
TestEmployee.java

```
package employee;

class Employee {

    private String name;
    private String empNo;
    private int age;
    private double basicPay;

    // Constructor
    Employee(String name, String empNo, int age) {
        this.name = name;
        this.empNo = empNo;
        this.age = age;
    }

    public double calculateSalary(double daRate, doub

        double grossSalary = 0.0;

        double da = basicPay * daRate / 100;
        double hra = basicPay * hraRate / 100;
        grossSalary = basicPay + hra + da;

        return grossSalary;
    }
```

6.2.4 Import Statement

A package is to group related classes or other type of files together. The classes and other types under a package are called as package members. So as we saw above, we have grouped all files into different packages.

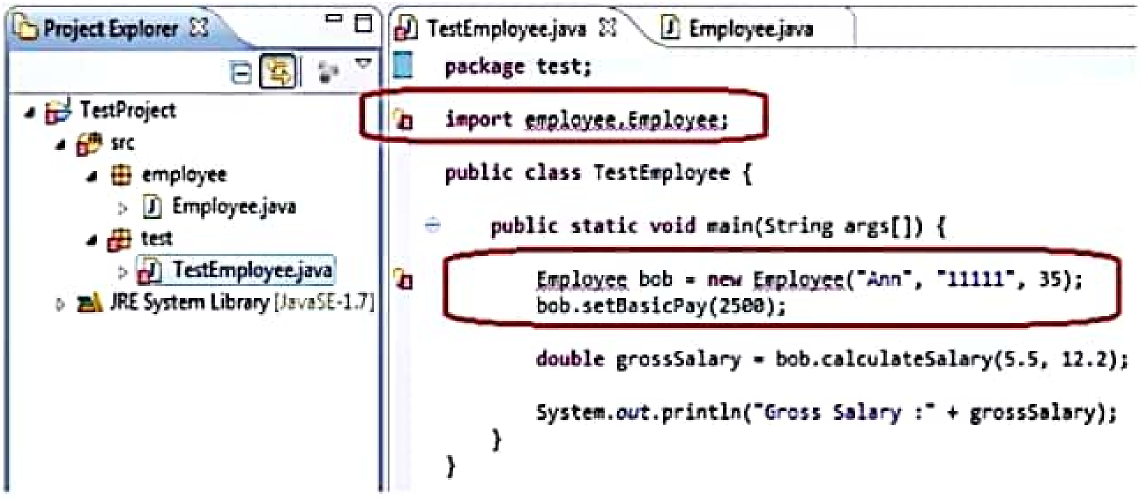
If a class wants to access/use another class/member from the same package, the class need not be imported. Classes/members within the same package can access each other without importing them.

But in the scenario where the classes belonging to different packages need to access each other, the class belonging to the other package must be imported into the class that wants to use it.

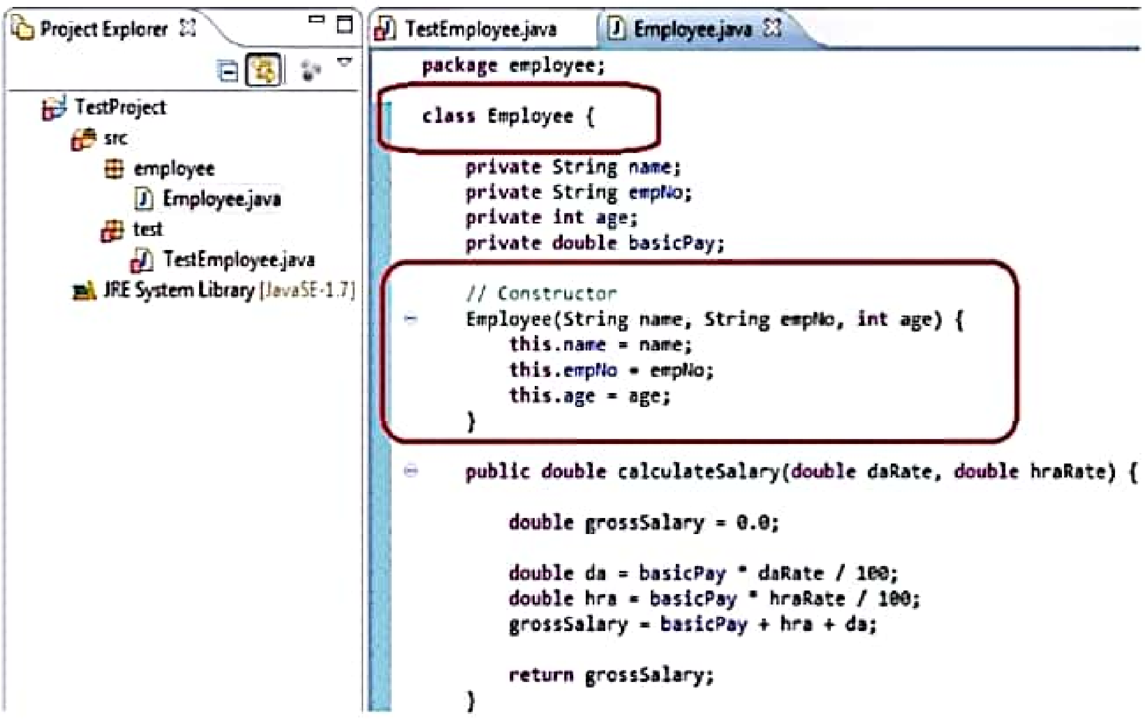
The classes are imported using import statements. Import keyword is used to import the classes from different packages. Import keyword is followed by the fully qualified name of the class which includes the package name under which the class is present.

Eg: import java.util.Date ;

In the example below the Employee class is imported into the TestEmployee class.
TCS Internal



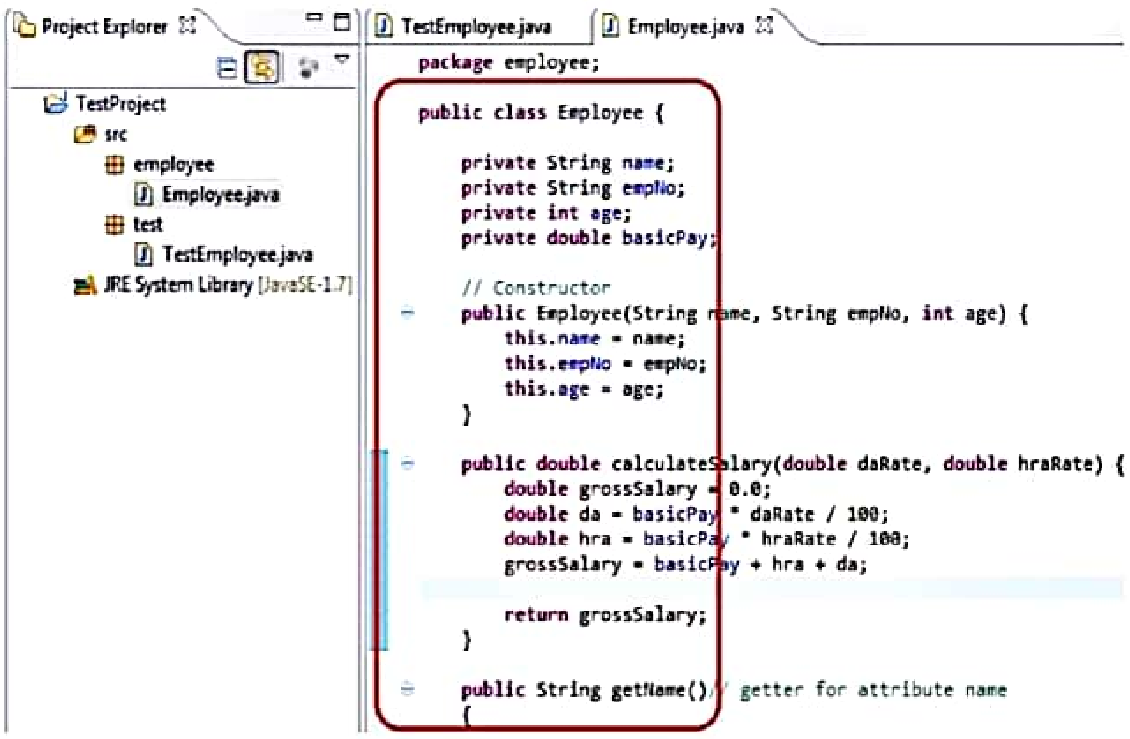
```
package test;

import employee.Employee;

public class TestEmployee {

    public static void main(String args[]) {

        Employee bob = new Employee("Ann", "11111", 35);
        bob.setBasicPay(2500);

        double grossSalary = bob.calculateSalary(5.5, 12.2);

        System.out.println("Gross Salary :" + grossSalary);
    }
}
```



```
package employee;

class Employee {

    private String name;
    private String empNo;
    private int age;
    private double basicPay;

    // Constructor
    Employee(String name, String empNo, int age) {
        this.name = name;
        this.empNo = empNo;
        this.age = age;
    }

    public double calculateSalary(double daRate, double hraRate) {

        double grossSalary = 0.0;

        double da = basicPay * daRate / 100;
        double hra = basicPay * hraRate / 100;
        grossSalary = basicPay + hra + da;

        return grossSalary;
    }
```

You can see in the above example, the TestEmployee class is under the test package. You can also see the import statement to import Employee in TestEmployee class. This is there because TestEmployee is using Employee class, which is part of a different package employee. However even after the Employee class is imported into TestEmployee there are still errors in Testemployee class. This is because the Employee class is not visible to classes outside its package. Earlier when the TestEmployee class was inside employee package, it knew the Employee class. But when it was moved into a different package, and even after Employee class was imported the Employee class and its constructor are not visible. This is because only public members of a package/class are visible to classes from other packages. So specify public access specifier to the class, constructor and method which need to be accessed from outside.

In the below screen, you can see the changes made to access specifiers of Employee class and its constructor and methods. They are all declared with public access modifier.



```java
package employee;

public class Employee {

    private String name;
    private String empNo;
    private int age;
    private double basicPay;

    // Constructor
    public Employee(String name, String empNo, int age) {
        this.name = name;
        this.empNo = empNo;
        this.age = age;
    }

    public double calculateSalary(double daRate, double hraRate) {
        double grossSalary = 0.0;
        double da = basicPay * daRate / 100;
        double hra = basicPay * hraRate / 100;
        grossSalary = basicPay + hra + da;

        return grossSalary;
    }

    public String getName()// getter for attribute name
    {
```

To use a public package member from outside its package any one of the following can be used

a) Refer to the class/member by its fully qualified name
b) Import specific class/member in a package
c) Import all members/classes in a package

<u>Refer to the class/member by its fully qualified name</u>
employee.Employee = new employee.Employee() ;

<u>Import all members/classes in a package:</u>
By using * wild card, we can import all the classes inside employee package.

    import employee.* ;

<u>Import specific members/classes in a package:</u>
We can also specify only the needed classes in the import statement.

    import employee.Employee ;

If a class in a package has same name as another class in another package and both packages are imported then you must refer to each class by its qualified name to resolve any name ambiguities.

6.2.5 More on Access Modifiers

We have discussed about 2 access specifiers in the previous courses. Which are they?

Yes -> public and private.

In Java, there are 4 access specifiers, they are:

Private Access Modifier – private
Methods, Variables and Constructors that are declared private can only be accessed within the declared class itself. Private access modifier is the most restrictive access level.

Variables that are declared private can be accessed outside the class if public getter methods are present in the class. Using the private modifier is the main way that an object encapsulates itself and hide data from the outside world.

Public Access Modifier – public
A class, method or constructor declared public can be accessed from any other class. Therefore attributes or methods declared inside a public class can be accessed from any other Java class which is inside the same package or inside another package. If the public class which we try to access is in a different package, it needs to be imported first.

Default Access Modifier - No keyword
For default access modifier, we do not explicitly declare an access modifier for a class, field, method, constructors etc.

A variable or method declared without any access control modifier is available to any other class in the same package. Such members can be considered as package private.
We have already seen such a scenario when we discussed about import statements.

Eg: The constructor method in the following Java class is in default access.

```
    class Employee {

        private String name;
        private String empNo;
        private int age;
        private double basicPay;

        // Constructor
        Employee(String name, String empNo, int age) {
            this.name = name;
            this.empNo = empNo;
            this.age = age;
        }
    }
```

Protected Access Modifier – protected
Protected is related to inheritance (a parent child relation). Protected members in a class can be

accessed only by its child classes from same/different package as well as by any other classes in the same package.

The child class can be anywhere, in other package or within the same package of the parent class.

The protected access modifier can be applied only to methods or fields, not to class. We can see more on it when we go through inheritance.

6.2.6 Java Libraries

Java differs from most other languages in that the number of classes and interfaces in its standard libraries is very large. Many common tasks have already been implemented by these libraries.

They are generally of high quality and are widely used. Implementing something which already exists in the libraries is probably wasted effort. Significant changes and additions to the standard libraries occur in each major release of Java, and it pays to keep current.

The most widely used packages are java.lang and java.util. There are other packages used for working with data such as java.sql, javax.sql, java.io etc.
For graphical applications, see the Swing classes (javax.swing, and so on).

You should go through JDK documentation just to get an idea of what is available. Later, when a specific need arises, you will often know which packages might be helpful.

6.3 Test Your Knowledge

1. What modifier can be used to prevent any method or attribute to be not visible to external classes?
   a. Private
   b. Public
   c. Protected
   d. Default

Ans: a

2. What modifier can be used to make any method or attribute to be visible to external classes?
   a. Private
   b. Public
   c. Protected
   d. Default

Ans: b

3. What modifier can be used to restrict any method or attribute's visibility to all the classes in the same package and to only its child classes in other packages?
   a. Private
   b. Public
   c. Protected
   d. Default

Ans: c

4. What modifier can be used to restrict a method or attribute to be invisible outside its package?
   a. Private
   b. Public
   c. Protected
   d. Default (No modifier)

Ans: d

5. If package name containing more than one word is used, the words must be separated by
   a. comma
   b. dot
   c. hyphen
   d. underscore

Ans: b

6. A namespace that organizes classes and interfaces by functionality is called a _______
Ans: Package

7. All the classes in a package can be simultaneously imported by using ___ wildcard
Ans: *