3.5. Packages

- A package is a namespace that organizes a set of related classes, interfaces, enums and annotations.
- Package indicates the directory or folder in your file system where you can place your project files.
- Package can be defined as namespace to identify the classes uniquely.
- Package can be used to specify the access scope of your class or class members.

3.5.1. Package Declaration

Syntax:

```
package <packageName>;
```

Ex:

```
package com.myjlc.p1;
package com.myjlc.p2;
etc
```

Hai.java

```
class Hai{
public static void main(String args[]){
System.out.println("Hai Guys !!!");
}
}
```

Hello.java

```
package com.myjlc;

class Hello{
public static void main(String args[]){
System.out.println(" Hello Guys !!!");
}
}
```

Steps to compile without package

Syntax:

```
javac <sourceFileName>.java
```

Ex:

```
javac  Hai.java
```

Steps to compile with package

Syntax:

```
javac –d <location> <sourceFileName>.java
```

Ex:

```
javac –d . Hello.java
```

Steps to run without package

Syntax:

```
java <className>
```

Ex:

```
java  Hai
```

Steps to run with package

Syntax:

```
java <packageName>.<className>
```

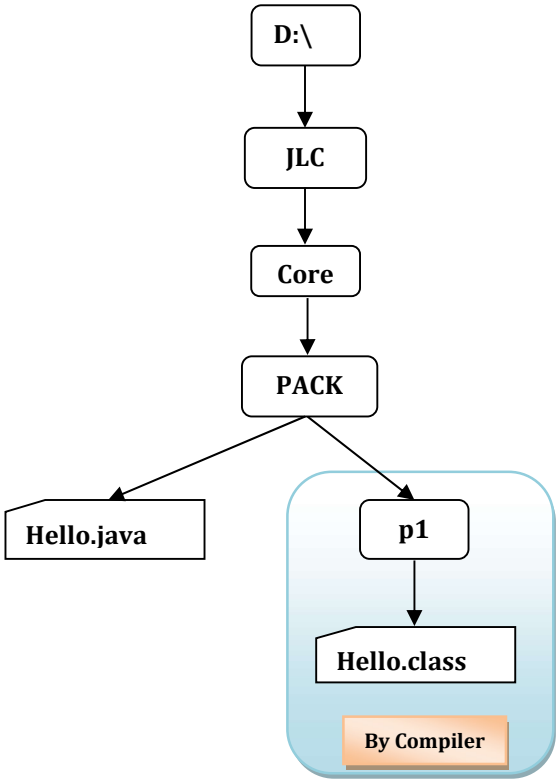
Ex:

```
java  com.myjlc.Hello
```

- In the above examples:
  - Hai is written without any package declaration statement .
  - Hello is written with a package declaration statement called "package com.myjlc".
- When you are writing classes without any package declaration statement then those classes will be placed in the default package.
- Default package is the package without any name i.e. when you compile source files with default package then compiled class files will be placed in current folder only.

- When you are writing classes with a package declaration statement then those classes will be placed in the specified package i.e. when you compile source files with package declaration statement then compiled class files will be placed in specified package.
- When a class is specified inside a package then that class must be referred always with the package name which is also called as fully qualified class name.

Ex:

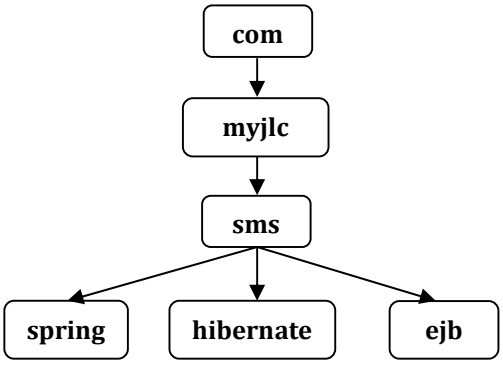


3.5.2. Organizing Package Names

- Package name is a combination of multiple words seprated with dot (.) symbol.
- Each word in the package name represents one folder or directory in the file system.

Ex:

```
package  com.myjlc.sms.spring;
package  com.myjlc.sms.hibernate;
package  com.myjlc.sms.ejb;
```



Lab419.java

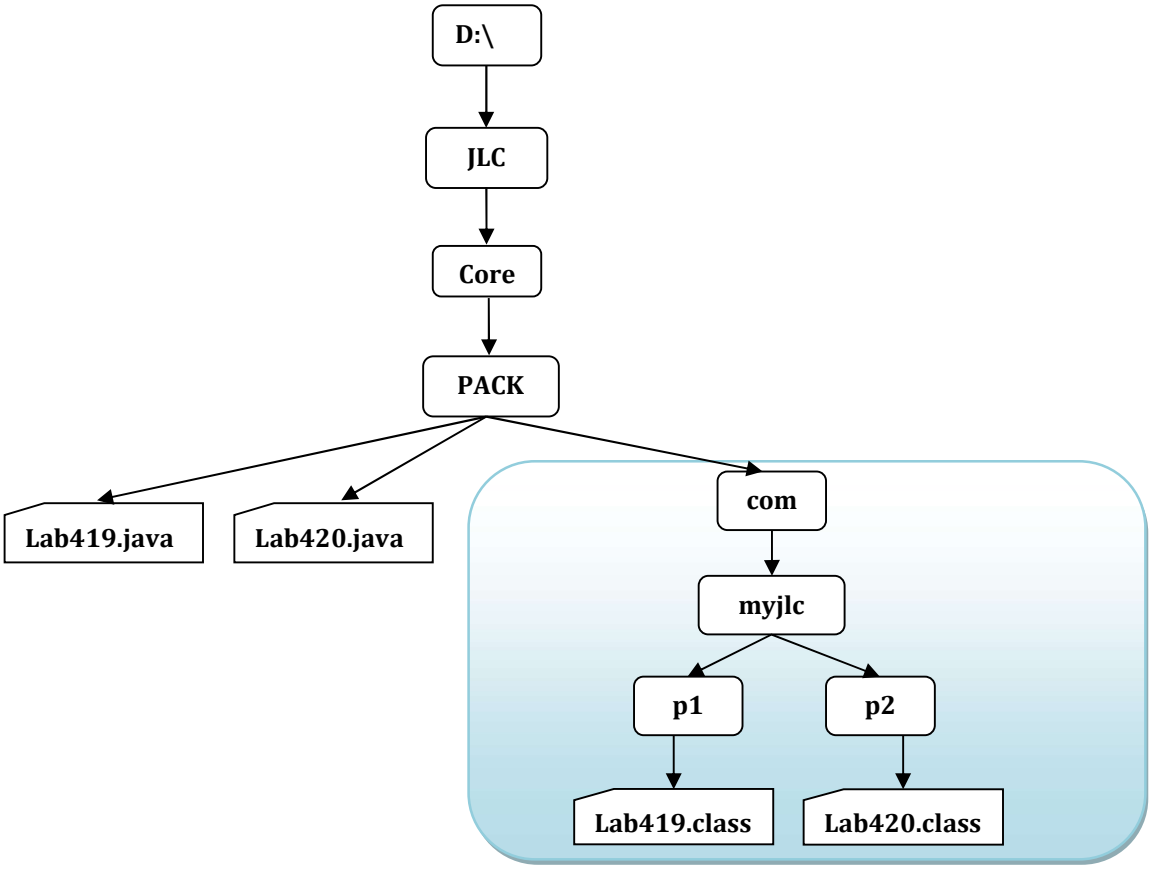
```
package com.myjlc.p1;

class Lab419{
public static void main(String args[]){
System.out.println(" Hello Guys !!! from p1
package");
}
}
```

Lab420.java

```
package com.myjlc.p2;

class Lab420{
public static void main(String args[]){
System.out.println(" Hello Guys !!! from p2
package");
}
}
```



3.5.3. Exploring Compilation

Syntax:

```
javac  –d  <Location>  <SourceFileName>.java
```

Ex:

```
javac  –d  .  Lab.java
```

- **Here dot(.) is used as Location, so package will be created in the current working directory and compiled classes will be placed in that package.**
- **If you want to create the package in specified directory instead of current directory then you have to specify the location.**
- **Assume that current working directory is D:\JLC\Core\pack**

Hello.java

```
package com.myjlc;

class Hello{
public static void main(String str[]){
System.out.println("Hello Guys!");
}
}
```

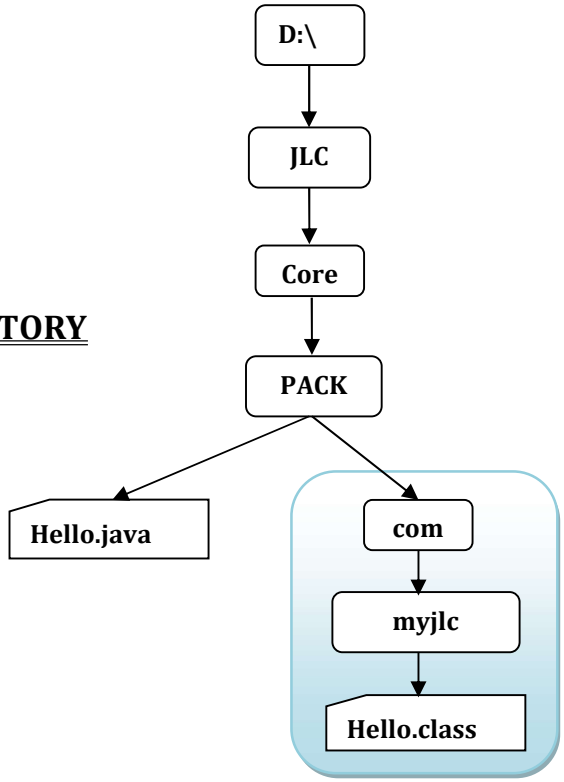
Case 1:Creating package in CURRENT DIRECTORY

Relative Location

```
javac -d . Hello.java
```

Absolute Location

```
javac -d D:\JLC\Core\pack Hello.java
```



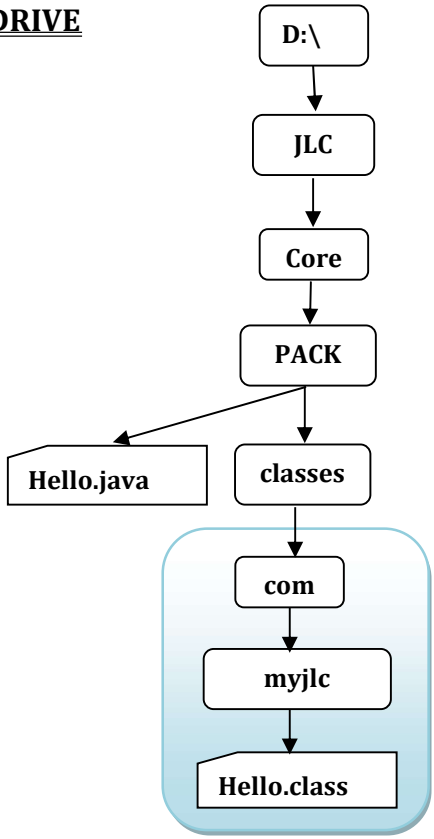
Case 2: Creating package in OTHER DIRECTORY IN SAME DRIVE

Relative Location

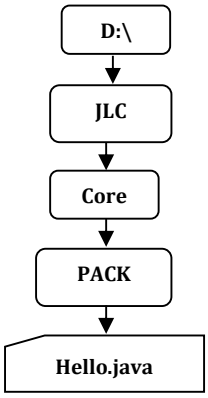
```
javac -d classes Hello.java
```

Absolute Location

```
javac -d D:\JLC\Core\pack\classes Hello.java
```

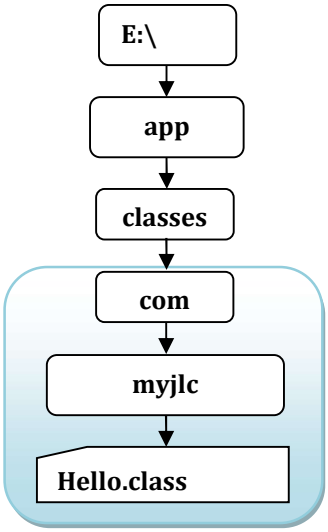


Case 3: Creating package in OTHER DIRECTORY IN OTHER DRIVES



Absolute Location

```
javac -d E:\app\classes  Hello.java
```



3.5.4. Exploring CLASSPATH

- **CLASSPATH represents the path where class files or packages or JAR files available.**
- **By default you can access the classes from the same location where it is available.**
  - **For Example, Consider the above CASE 3.**

    java com.jlcindia.Hello -> Gives an error
- **When you want to access classes from other location of file system then you need to specify the location of class file by setting the CLASSPATH.**
  - **For Example, Consider the above CASE 3.**

    Option 1: Temporarly for Current Execution

    java –cp E:\app\classes com.jlcindia.Hello -> Runs Successfully

    Option 2: Temporarly for Current Command Prompt

    set CLASSPATH=%CLASSPATH%;E:\app\classes;

    java com.jlcindia.Hello -> Runs Successfully

    Option 3: Permanently in System Environment Variables

    Steps:

    - **Open My Computer**
    - **Right Click on Blank Area and Select Properties**
    - **From Left Side Panel, Click on Advanced system settings**
    - **Click on Environment Variables Button**
    - **Click on New Button in the User Variable**
    - **Provide the required info**

      Variable name CLASSPATH

      Variable value %CLASSPATH%;E:\app\classes;
    - **Click on OK button in all the windows.**

3.5.5. JAR (Java Archive)

- **Package is the collection of classes whereas JAR is the collection of packages.**
- **JAR file format is platform-independent.**
- **You can compress and bundle multiple files associated with a Java application into a single file called as JAR file.**
- **It is based on the ZIP algorithm.**

Syntax:

jar -cf <NameOfJARFile> <Pack1> <Pack2> ...
jar -cvf <NameOfJARFile> <Pack1> <Pack2> ...

Ex:

Jar cvf myjlc.jar com

- **When the JAR file is not available in current location, packages will not be accessed by compiler or JVM.**
- **You need to set the CLASSPATH for the JAR file to access the packages from JAR file.**
  - **SET CLASSPATH=%CLASSPATH%;D:\jlc\myjar\jlc.jar;**
- **Now the classes available in JAR file will be accessed by compiler or JVM.**

3.5.6. Labs on Packages

Lab421.java

```
class Hello{

}

package com.myjlc;

class Lab421{
public static void main(String args[]){
System.out.println(" Hello Guys !!! ");
}
}
```

Lab422.java

```
package com.myjlc.p1;
package com.myjlc.p2;

class Hello{

}

class Lab422{
public static void main(String args[]){
System.out.println(" Hello Guys !!! ");
}
}
```

Lab423.java

```
package com.myjlc;

public class Hello{ }

public class Lab423{
public static void main(String args[]){
System.out.println(" Hello Guys !!! ");
}
}
```

Lab424.java

```
package com.myjlc;

class Hello{ }
class Hai { }
public class Lab424{
public static void main(String args[]){
System.out.println(" Hello Guys !!! ");
}
}
```

Lab425.java

```
package com.myjlc;

public class Hello{
public static void main(String args[]){
System.out.println(" Hello Guys !!! ");
}
}
```

Lab425.java

```
package com.myjlc;

public class Hello{
public static void main(String args[]){
System.out.println(" Hello Guys !!! ");
}
}
```

3.5.7. Importing Packages

- **If you want to access one class from another class with in the same package then you can access directly without using any package name.**

Hello.java

```
package com.myjlc.p1;
class Hello{
void m2(){
System.out.println("Hello - m2() ");
}
}
```

Hai.java

```
package com.myjlc.p1;
class Hai{
void m1(){
System.out.println("Hai - m1() ");
}
}
```

Lab426.java

```
package com.myjlc.p1;

public class Lab426 {
public static void main(String args[]){
Hai hai = new Hai();
hai.m1();

Hello hello=new Hello();
hello.m2();
}
}
```

- **If you want to access a class from different package then:**
  - That class must be public.
  - That package must be available in the classpath.
  - That package information must be specified.

- **You can specify the package information in two ways:**
  - Using Import Statements
  - Using Fully Qualified Class Name (<packageName>.<className>)

Using Fully Qualified Class Name

- When you want to access any class from any package, just refer the class with the package name.

  Ex

  com.jlcindia.p1.Hai
  com.jlcindia.p1.Hello

Using Import Statements

- Import statement allows you to access the classes from other packages.

  Syntax:

  1) Single type import statement

  import <packageName>.<class/interfaceName>;

  Ex:

  a) import com.jlcindia.p1.Hello;
  b) import com.jlcindia.p1.Hai;

  In above case a, only the Hello class of p1 package will be accessed.
  In above case b, only the Hai class of p1 package will be accessed.

  2) Multi type import statement

  import <packageName>.*;

  Ex:

  a) import com.jlcindia.p1.*;

  In above case a, all the classes of p1 package will be accessed.

Hai.java

```
package com.myjlc.p1;

public class Hai{
public void m1(){
System.out.println("Hai - m1() ");
}
}
```

Hello.java

```
package com.myjlc.p1;

public class Hello{
public void m2(){
System.out.println("Hello - m2() ");
}
}
```

Lab427.java

```
package com.myjlc.p2;

import com.myjlc.p1.*;
//import com.myjlc.p1.Hai;
//import com.myjlc.p1.Hello;
public class Lab427 {
public static void main(String args[]){
Hai hai = new Hai();
hai.m1();

Hello hello=new Hello();
hello.m2();
}
}
```

Lab428.java

```
package com.myjlc.p2;

public class Lab428 {
public static void main(String args[]){

com.myjlc.p1.Hai hai =
        new com.myjlc.p1.Hai();
hai.m1();

com.myjlc.p1.Hello hello=new
        com.myjlc.p1.Hello();
hello.m2();
}
}
```

Lab429.java

```
package com.myjlc;

import java.util.ArrayList;

public class Lab429{
public static void main(String args[]){

ArrayList mylist = new ArrayList();
java.io.File myfle = new java.io.File("hello");
String str = new String("Ok Guys");
}
}
```

Lab430.java

```
package com.myjlc;

import java.util.*;
import java.sql.*;

public class Lab430{
public static void main(String args[]){

java.util.Date mydate = new java.util.Date();
System.out.println(mydate);
}
}
```

3.5.8. Static Imports

- It is the new feature of Java 5.
- You can access the classes of other packages by using normal import statements where as you can acess public static members of the class by using static import statements.

Normal Imports

Syntax:

- import <PackageName>.<ClassName>;
- import <PackageName>.*;

Ex:

```
import com.jlcindia.p1.Hello;
import com.jlcindia.p1.*;
```

In above case 1, only the Hello class of p1 package will be accessed.
In above case 2, all the classes of p1 package will be accessed.

Static Imports

Syntax:

- import static <PackageName>.<ClassName>.<StaticMember>;
- import static <PackageName>.<ClassName>.*;

Ex:

```
import static com.jlcindia.p1.Hello.A;
import static com.jlcindia.p1.Hello.m1;
import static com.jlcindia.p1.Hello.*;
```

In above case 1, only the public static variable A of class Hello will be accessed.
In above case 2, only the public static method m1() of class Hello will be accessed.
In above case 3, all the public static members of class Hello will be accessed.

Hai.java

```
package com.myjlc.p1;

public class Hai{

public int A=10;
public static int B=20;

public void m1(){
System.out.println("Hai - m1() ");
}

public static void m2(){
System.out.println("Hai - m2() ");
}

}
```

Hello.java

```
package com.myjlc.p2;

public class Hello{

public int C=30;
public static int D=40;

public void m3(){
System.out.println("Hello - m3() ");
}

public static void m4(){
System.out.println("Hello - m4() ");
}

}
```

Lab431.java

```
package com.myjlc.p3;

import static com.myjlc.p1.Hai.*;
import static com.myjlc.p2.Hello.*;

public class Lab431 {
public static void main(String args[]){

Hai hai = new Hai();
Hello hello=new Hello();

}
}
```

Lab432.java

```
package com.myjlc.p3;

import static com.myjlc.p1.Hai.*;
import static com.myjlc.p2.Hello.*;

public class Lab432{
public static void main(String args[]){

System.out.println(A);
System.out.println(C);
m1();
m3();

}
}
```

Lab433.java

```
package com.myjlc.p3;

import static com.myjlc.p1.Hai.*;
import static com.myjlc.p2.Hello.*;

public class Lab433{
public static void main(String args[]){
System.out.println(B);
System.out.println(D);
m2();
m4();
}
}
```

Lab434.java

```
package com.myjlc.p3;

import static com.myjlc.p2.Hello.m4;

public class Lab434{

public static void m4(){
System.out.println("Lab435 - m4() ");
}

public static void main(String args[]){
m4();
}
}
```

Lab435.java

```
package com.myjlc.p3;

import static java.lang.Math.*;
import static java.lang.System.*;

public class Lab435{

static int A=123;

public static void main(String args[]){

out.println(sqrt(25));
out.println(min(10,20));
out.println(PI);
out.println(A);

}
}
```

Lab435.java

```
package com.myjlc.p3;

import static java.lang.Math.*;
import static java.lang.System.*;

public class Lab435{

static int A=123;

public static void main(String args[]){

out.println(sqrt(25));
out.println(min(10,20));
out.println(PI);
out.println(A);

}
}
```

SUMMARY

PACKAGE STATEMENTS

1. Packages can be used :
    a. to group the classes, interfaces, enums and annotations.
    b. to prevent naming conflicts.
    c. to specify access scope.
2. Only one package declaration statement can be defined in the source file.
3. Package declration statement must be the first statement in the source file.
4. One source file can contain multiple classes and all the classes available in the source file will be placed in the same package
5. One source file can contain:
    a. only one public class.
    b. many non public classes.
6. When a class is public then source file name must be same as public class name.
7. One source file can have only package declaration statement and one public class but one package can have multiple public classes.
8. If you want to place multiple public classes in the same pacakge then you have to place the public classes in the seperate source file.

Hello.java

```
package com.jlcindia.p1;
public class Hello{
public static void main(String str[]){
System.out.println("Hello Guys!");
}
}
```

Lab.java

```
package com.jlcindia.p1;
public class Lab{
public static void main(String str[]){
System.out.println("Hello Guys!");
}
}
```

IMPORT STATEMENTS

9. One source file can have zero or more import statements.
10. Import statement should be available after the package declration statement and before the class definitions.

11. When you import a package then classes available in that package will not be loaded. Classes will be loaded only when you use the members of the class.
12. When you are importing multiple packages which are having same class then don't use multi type import statements because that may cause ambiguity.
13. You can't import same class two times using SINGLE TYPE IMPORT.
14. Package name should not start with following words:
    a. java
    b. con
    c. com1 to com9
15. When you are importing multiple packages that contains common classes then reference to that common class will be ambigious in your program. To resolve that ambiguity you can refer that class with fully qualified class name.

STATIC IMPORTS

16. Only public static memebrs of a class will be accessed by using static imports.
17. Non static memebrs of a class will not be accessed by using static imports.
18. Normal imports allow you to access classes whereas static imports allows you to access static members. So when you use static imports then classes will not be accessed.
19. When you refer any static member which is available in same class as well as imported by static imports then priority will be given to same class member.
20. When you are using multiple static imports that contain common staic members then reference to those common static members will be ambigious in your program. To resolve that ambiguity you can refer that static member with class name.
21. When you refer any class which is defined in same package as well as in imported package then priority will be given to same package class. You can use fully qualified class name to access specified class available in imported package.

Assignment #11

Q1) What is a package?

Q2) How to declare a package?

Q3) What is the location of package declaration statement in a java program?

Q4) How many package declaration statements allowed in a java program?

Q5) What are the use of Packages?

Q6) What are the invalid words that cannot be used to start a package name?

Q7) What are the valid files that can be clubed into a package?

Q8) How many public and non public classes can be placed in one source file?

Q9) What is the name of java source file when it contains public class?

Q10) Is it mandatory to place main() method in public class?

Q11) How to place multiple public classes in one package?

Q12) How to place multiple non public classes in one package?

Q13) What is default package?

Q14) How to refer the classes which is available in default package?

Q15) What is fully qualified class name?

Q16) Can I refer class with fully qualified class name which is available in default package?

Q17) What is the syntax to compile a package program?

Q18) What is the syntax to execute a package program?

Q19) Can I compile multiple java programs at a time?

Q20) How to create a package in CURRENT DIRECTORY?

Q21) How to create a package in OTHER DIRECTORY IN SAME DRIVE?

Q22) How to create a package in OTHER DIRECTORY IN OTHER DRIVE?

Q23) What is a CLASSPATH?

Q24) What are the ways available to set the CLASSPATH?

Q25) Can I access non public classes from different package?

Q26) What is the use of import statements in your java program?

Q27) What are the type of import statements available?

Q28) What is the location of import statement in a java program?

Q29) How many import statements can be used in a java program?

Q30) What is ambiguity problem in normal imports and how to resolve it?

Q31) What is static imports and which version of Java supports this?

Q32) What is the difference between static imports and normal imports?

Q33) Can I access non public static members using static imports?

Q34) Can I access non public instance members using static imports?

Q35) What is ambiguity problem in static imports and how to resolve it?

Q36) What is JAR?

Q37) What is the syntax to create normal JAR file?

Q38) What is the syntax to extract JAR file?

Q39) What is use of verbose option in creating or extracting JAR files?

Practice Test #11

Q.No	Question	Options	Answer

1	class Hello{
	int a=99;
	}
	
	package com.jlc;
	class Test1{
	public static void main(String args[]){
	Hello h=new Hello();
	System.out.println(h.a);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) No Output	

2	package com.jlc;
	class Hello{
	int a=99;
	}
	
	class Test2{
	public static void main(String args[]){
	Hello h=new Hello();
	System.out.println(h.a);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) No Output	

3	class Test3{
	public static void main(String args[]){
	Object obj="SRI";
	System.out.println(obj);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) JLC
		D) No Output	

4	class Test4{
	public static void main(String args[]){
	java.lang.Object obj="SRI";
	System.out.println(obj);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) JLC
		D) No Output	

5	**Hello.java**
	package com.jlc;
	class Hello{
	int a=99;
	}
	
	Test5.java
	package org.sd;
	class Test5{
	public static void main(String args[]){
	Hello h=new Hello();
	System.out.println(h.a);
	} }
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) No Output	

6	**Hello.java**
	package com.jlc;
	class Hello{
	int a=99;
	}
	
	Test6.java
	package org.sd;
	class Test6{
	public static void main(String args[]){
	com.jlc.Hello h=new com.jlc.Hello();
	System.out.println(h.a);
	} }
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) No Output	

7	**Hello.java**
	package com.jlc;
	public class Hello{
	int a=99;
	}
	
	Test7.java
	package org.sd;
	import com.jlc.Hello;
	class Test7{
	public static void main(String args[]){
	Hello h=new Hello();
	System.out.println(h.a);
	} }
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) No Output	

8	**Hello.java**
	package com.jlc;
	public class Hello{
	public int a=99;
	}
	
	Test8.java
	package org.sd;
	import com.jlc.Hello;
	class Test8{
	public static void main(String args[]){
	Hello h=new Hello();
	System.out.println(h.a);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) No Output	

9	**Hello.java**
	package com.jlc;
	public class Hello{
	public int a=99;
	}
	
	Test9.java
	package org.sd;
	class Test9{
	public static void main(String args[]){
	System.out.println("Hello Guys");
	}
	}
	
	import com.jlc
	class Test9{
	public static void main(String args[]){
	Hello h=new Hello();
	System.out.println(h.a);
	}
	}
		When you execute Test1 class
		
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) Hello Guys
		E) No Output
		
		When you execute Test2 class
		
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) Hello Guys
		E) No Output	

10	**Hello.java**
	package com.jlc;
	public class Hello{
	public int a=99;
	}
	Hai.java
	package com.jlc;
	public class Hai{
	public int b=88;
	}
	Test10.java
	package org.sd;
	import com.*;
	class Test10{
	public static void main(String args[]){
	Hello h1=new Hello();
	System.out.println(h1.a);
	Hai h2=new Hai();
	System.out.println(h2.b);
	} }
		A) Compilation Error
		B) Runtime Error
		C) 99
		88
		D) 99
		E) 88
		F) No Output	

11	**Hello.java**
	package com.jlc;
	public class Hello{
	public static int A=99;
	}
	Hai.java
	package com.jlc;
	public class Hai{
	public static int B=88;
	}
	Test11.java
	package org.sd;
	import static com.jlc.Hello.A;
	import static com.jlc.Hai.B;
	class Test11{
	public static void main(String args[]){
	System.out.println(A);
	System.out.println(B);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 99
		88
		D) 99
		E) 88
		F) No Output	

12	**Hello.java**
	package com.jlc;
	public class Hello{
	public static int A=99;
	}
	
	Test12.java
	package org.sd;
	import static com.jlc.Hello;
	class Test12{
	public static void main(String args[]){
	Hello h=new Hello();
	System.out.println(h.A);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) No Output	

13	**Hello.java**
	package com.jlc;
	public class Hello{
	public static int A=99;
	}
	
	Hai.java
	package com.jlc;
	public class Hai{
	public static int A=88;
	}
	
	Test13.java
	package org.sd;
	import static com.jlc.Hello.*;
	import static com.jlc.Hai.*;
	class Test13{
	public static void main(String args[]){
	System.out.println(A);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) No Output	

14	**Hello.java**
	package com.jlc;
	public class Hello{
	public static int A=99;
	}
	Hai.java
	package com.jlc;
	public class Hai{
	public static int A=88;
	}
	Test14.java
	package org.sd;
	import static com.jlc.Hello.*;
	import static com.jlc.Hai.*;
	class Test14{
	public static void main(String args[]){
	System.out.println(Hello.A);
	System.out.println(Hai.A);
	} }
		A) Compilation Error
		B) Runtime Error
		C) 99
		88
		D) 99
		E) 88
		F) No Output	

15	**Hello.java**
	package com.jlc;
	public class Hello{
	public static int A=99;
	}
	Hai.java
	package com.jlc;
	public class Hai{
	public static int A=88;
	}
	
	Test15.java
	package org.sd;
	import com.jlc.Hello;
	import com.jlc;
	class Test15{
	public static void main(String args[]){
	System.out.println(Hello.A);
	System.out.println(Hai.A);
	} }
		A) Compilation Error
		B) Runtime Error
		C) 99
		88
		D) 99
		E) 88
		F) No Output	

3.6. Access Modifiers

- Access modifiers can be used to specify the scope or visibility of the class or members of the class.
- There are three access modifiers:
  - private
  - protected
  - public
- **There are four scopes available:**

Scope or Visibility	Modifier
Private Scope	private
Default Scope	No Modifier
Protected Scope	protected
Public Scope	public

Private Scope

- When you are using private modifier with the members of the class then the scope of those members will be private scope.
- Private scope is also called as class scope i.e private members must be accessed from the same class where it is declared.
- Private members cannot be accessed from outside the class, not even from sub class i.e private members will not be inherited to sub classes.
- Private modifier can be used only for members of the class not for class itself.

Default Scope

- When you are not using any modifier with the members of the class then the scope of those members will be default scope.
- Default scope is also called as packaged scope i.e. default members can be accessed from the same class, sub class and non sub class which are available in same package.
- Default members cannot be accessed from outside the package i.e default members will not be inherited to sub class available in outside the package.
- Default scope is available for both classes and their members.

Protected Scope

- When you are using protected modifier with the members of the class then the scope of those members will be protected scope.
- Protected members can be accessed from the same class, sub class and non sub class which are available in same package.
- Protected members can be accessed from the sub classes which are available in different package.
- Protected members cannot be accessed from the non sub classes which are available in different package.
- Protected modifier can be used only for members of the class not for class itself.

Public Scope

- When you are using public modifier with the members of the class then the scope of those members will be public scope.
- Public members can be accessed from anywhere.
- Public modifier can be used for both classes and their members.

A.java

```
package com.myjlc.p1;

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

public void showA(){
System.out.println("showA() ");
System.out.println(a);
System.out.println(b);
System.out.println(c);
System.out.println(d);
}

}
```

B.java

```
package com.myjlc.p1;

public class B extends A{
public void showB(){
System.out.println("showB() ");
//System.out.println(a);
System.out.println(b);
System.out.println(c);
System.out.println(d);
}

}
```

C.java

```
package com.myjlc.p1;

public class C{
public void showC(){
System.out.println("showC() ");
A ao = new A();
//System.out.println(ao.a);
System.out.println(ao.b);
System.out.println(ao.c);
System.out.println(ao.d);
}
}
```

D.java

```
package com.myjlc.p2;

import com.myjlc.p1.*;

public class D extends A{
public void showD(){
System.out.println("showD() ");
//System.out.println(a);
//System.out.println(b);
System.out.println(c);
System.out.println(d);
}
}
```

E.java

```
package com.myjlc.p2;

import com.myjlc.p1.*;

public class E{

public void showE(){
System.out.println("showE() ");
A ao = new A();
//System.out.println(ao.a);
//System.out.println(ao.b);
//System.out.println(ao.c);
System.out.println(ao.d);
}

}
```

Lab436.java

```
package com.myjlc.p3;

import com.myjlc.p1.*;
import com.myjlc.p2.*;

public class Lab436{
public static void main(String args[]){

A ao=new A();
ao.showA();

B bo=new B();
bo.showB();

C co=new C();
co.showC();

D do1=new D();
do1.showD();

E eo=new E();
eo.showE();
}
}
```

Hai.java

```
package com.myjlc.p4;

public class Hai{
protected void show(){
System.out.println("Hai - show() ");
}
}
```

Hello.java

```
package com.myjlc.p5;

import com.myjlc.p1.Hai;

public class Hello extends Hai{
public void m1(){
show();
}
}
```

Lab437.java

```
package com.myjlc.p6;

import com.myjlc.p4.Hai;
import com.myjlc.p5.Hello;

class Test extends Hai{

}

public class Lab437{
public static void main(String args[]){

Test test = new Test();
test.show();

Hello hello = new Hello();
hello.m1();
}
}
```

Lab437.java

```
package com.myjlc.p6;

import com.myjlc.p4.Hai;
import com.myjlc.p5.Hello;

class Test extends Hai{

}

public class Lab437{
public static void main(String args[]){

Test test = new Test();
test.show();

Hello hello = new Hello();
hello.m1();
}
}
```

SUMMARY

1) Protected members can be accessed directly or with super or this keyword in sub class of different package.
2) Protected members cannot be accessed outside the package:
    a. With object of super class.
    b. With one sub class object in other sub class.
3) Private variables cannot be accessed from outside the class.
4) You can use constructors or setter methods to modify the value of private variables.
5) You can use getter methods to access the value of private variables from outside the class.

Access rights for the different elements					
		private	default (no modifier)	protected	public
SAME PACKAGE	Same class	YES	YES	YES	YES
	Subclass	NO	YES	YES	YES
	Non sub-class	NO	YES	YES	YES
OTHER PACKAGE	Sub class using super keyword	NO	NO	YES	YES
	Sub class using current sub-class object	NO	NO	YES	YES
	Sub class using super class object	NO	NO	NO	YES
	subclass using other sub-class object	NO	NO	NO	YES
	Non sub-class another package	NO	NO	NO	YES

Assignment #12

Q1) How many access modifiers available in Java?

Q2) How many access scopes available in Java?

Q3) What is private scope?

Q4) What is default scope?

Q5) What is package-private scope?

Q6) What is protected scope?

Q7) What is public scope?

Q8) Can I access private members directly in subclass of same package?

Q9) Can I access private members with object in sub class of same package?

Q10) Can I use default keyword to specify default scope?

Q11) Can I access default members in sub class of different package?

Q12) Can I access default members directly in non sub class of same package?

Q13) Can I access protected members directly in sub class of different package?

Q14) Can I access protected members using super in sub class of different package?

Q15) Can I access protected members using this in sub class of different package?

Q16) Can I access protected members using super class object in sub class of different package?

Q17) Can I access protected members directly in non sub class of same package?

Q18) Can I access public members directly in non sub class of different package?

Q19) Can I access public members directly in non sub class of different package?

Q20) How to modify private variables from outside the class?

Q21) How to access private variables from outside the class?

Practice Test #12

Q.No	Question	Options	Answer

1	class Hello{
	private int x;
	}
	
	class Test1{
	public static void main(String[] args){
	Hello h=new Hello();
	System.out.println(h.x);
	}
	}
		E) Compilation Error
		F) Runtime Error
		G) 0
		H) No Output	

2	class Hello{
	private int x;
	}
	
	class Hai extends Hello{
	private int x;
	public static void main(String[] args){
	Hai h=new Hai();
	System.out.println(h.x);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 0
		D) No Output	

3	class Hello{
	private int x;
	Hello(int x){
	this.x=x;
	}
	int getX(){
	return this.x;
	}
	}
	class Test2{
	public static void main(String[] args){
	Hello h=new Hello(99);
	System.out.println(h.getX());
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 0
		D) 99
		E) No Output	

4	**Hello.java**
	package com.jlc.p1;
	class Hello{
	public static void main(String[] args){
	System.out.println("Hai -> Main");
	}
	}
	
	Hai.java
	
	package com.jlc.p2;
	import com.jlc.p1.Hello;
	class Hai extends Hello{ }
		When we execute as
		Java com.jlc.p2.Hai
		
		A) Compilation Error
		B) Runtime Error
		C) Hai -> Main
		D) No Output
		E) None of above	

5	**Hello.java**
	package com.jlc.p1;
	public class Hello{
	public static void main(String[] args){
	System.out.println("Hai -> Main");
	}
	}
	
	Hai.java
	package com.jlc.p2;
	import com.jlc.p1.Hello;
	class Hai extends Hello{ }
		When we execute as
		Java com.jlc.p2.Hai
		
		A) Compilation Error
		B) Runtime Error
		C) Hai -> Main
		D) No Output
		E) None of above	

6	**Hello.java**
	package com.jlc.p1;
	public class Hello{
	void show(){
	System.out.println("SHOW");
	} }
	
	Hai.java
	package com.jlc.p2;
	import com.jlc.p1.Hello;
	class Hai extends Hello{
	int show(){
	return 99;
	}
		A) Compilation Error
		B) Runtime Error
		C) SHOW
		D) 99
		E) SHOW
		99
		F) No Output
		G) None of above	

	public static void main(String[] args){
	Hello h=new Hello();
	h.show();
	Hai ref=new Hai();
	System.out.println(ref.show());
	}
	}		

7	Hello.java
	package com.jlc.p1;
	public class Hello{
	void show(){
	System.out.println("SHOW");
	} }
	Hai.java
	package com.jlc.p2;
	import com.jlc.p1.Hello;
	class Hai extends Hello{
	int show(){
	return 99;
	}
	public static void main(String[] args){
	Hai ref=new Hai();
	System.out.println(ref.show());
	} }
		A) Compilation Error
		B) Runtime Error
		C) SHOW
		D) 99
		E) SHOW
		99
		F) No Output
		G) None of above	

8	A.java
	package com.jlc.p1;
	public class A{
	protected void show(){
	System.out.println("SHOW");
	}
	}
	Test.java
	package com.jlc.p2;
	import com.jlc.p1.A;
	class B extends A{}
	class C extends A{
	public static void main(String[] args){
	B ref=new B();
	ref.show();
	} }
		A) Compilation Error
		B) Runtime Error
		C) SHOW
		D) No Output
		E) None of above	

9	**A.java**
	package com.jlc.p1;
	public class A{
	protected void show(){
	System.out.println("SHOW");
	}
	}
	
	Test.java
	package com.jlc.p2;
	import com.jlc.p1.A;
	class B extends A{
	protected void show(){}
	}
	
	class C extends A{
	public static void main(String[] args){
	B ref=new B();
	ref.show();
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) SHOW
		D) No Output
		E) None of above	

10	**A.java**
	package com.jlc.p1;
	public class A{
	void show(){
	System.out.println("SHOW 1");
	}
	}
	Test4.java
	import com.jlc.p1.A;
	class B extends A{
	static void show(){
	System.out.println("SHOW 2");
	}
	}
	class Test4 {
	public static void main(String[] args){
	B.show();
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) SHOW 1
		D) SHOW 2
		E) SHOW 1
		SHOW 2
		F) None of above	

11	class Test5{
	public static void main(String args[]){
	B bobj=new B();
	bobj.x=99;
	System.out.println(bobj.x);
	} }
	class A{
	int x=10;
	}
	class B extends A{
	String x="JLC";
	}
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) 10
		E) None of above	

12	class Test6{
	public static void main(String args[]){
	B bobj=new B();
	bobj.x="HELLO";
	System.out.println(bobj.x);
	} }
	class A{
	int x=10;
	}
	class B extends A{
	String x="JLC";
	}
		A) Compilation Error
		B) Runtime Error
		C) HELLO
		D) JLC
		E) 10
		F) None of above	

13	class Test7{
	public static void main(String args[]){
	B bobj=new B();
	((A)bobj).x=99;
	System.out.println(bobj.x);
	} }
	class A{
	int x=10;
	}
	class B extends A{
	String x="JLC";
	}
		A) Compilation Error
		B) Runtime Error
		C) 99
		D) JLC
		E) 10	