Core Java

Lesson 08 : Packages and Imports

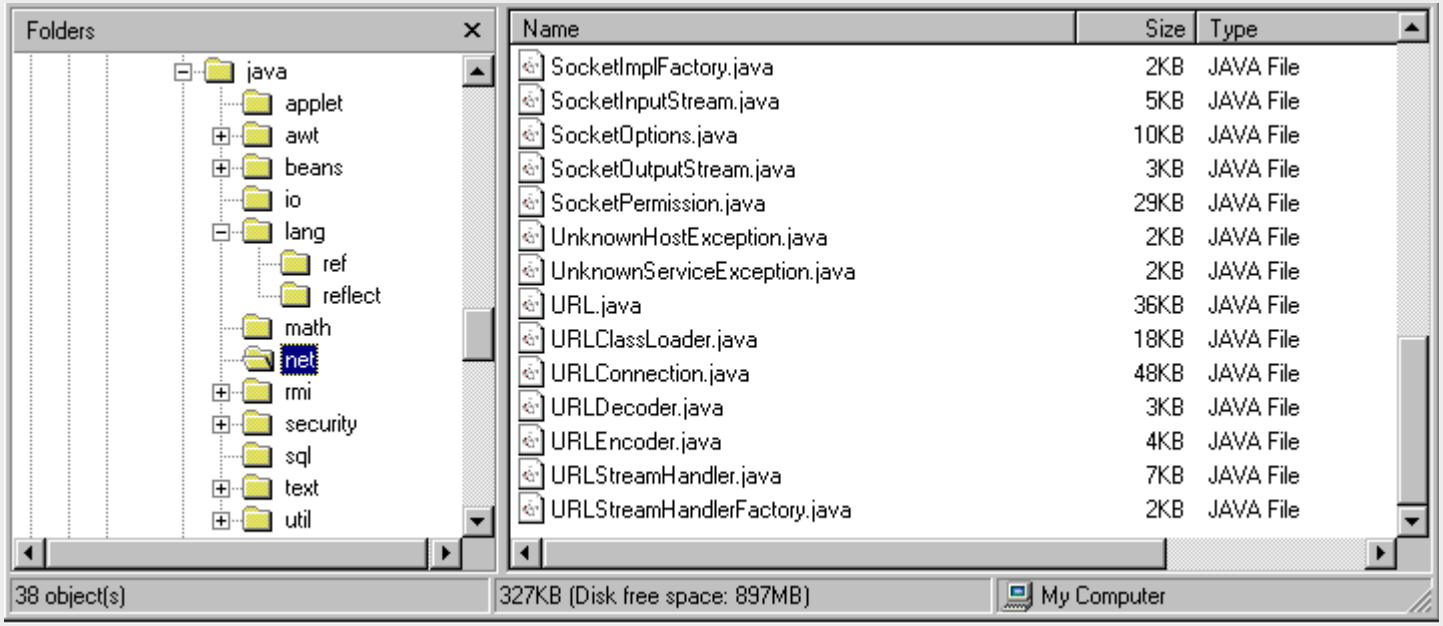
Capgemini

Lesson Objectives

- After completing this lesson, participants will be able to:
  - Create Packages
  - Naming packages
  - Package Access
  - Packages and class path
  - Importing packages
  - Static imports
  - Some Java Packages

Packages

- In Java, by the use of packages, you can group a number of related classes and/or interfaces together into a single unit.



Benefits of Packages

- These are the benefits of organising classes into packages:
  - It prevents name-space collision.
  - It indicates that the classes and interfaces in the package are related.
  - You know where to find the classes you want if they're in a specific package.
  - It is convenient for organizing your work and separating your work from code libraries provided by others.

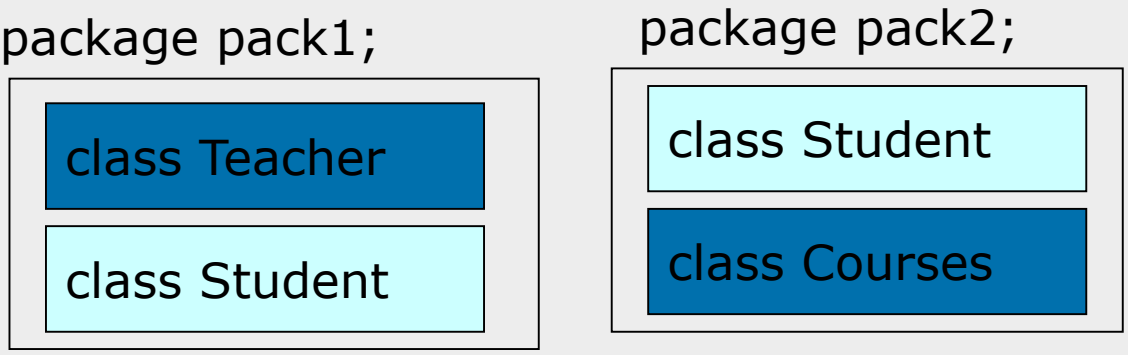
Creating Packages

```
package com.cg.trg.demo;
public class Balance {
    String name;
    public Balance(String n) {
        name = n;
     }
    public void show() {
      ……
       if( bal < 0)
         System.out.println(name + ": $" + bal);
}
```

Package should be the first statement

Naming Packages

- Namespace collision can be avoided by accessing classes with the same name in multiple packages by their fully qualified name.




```
import pack1.*;
import pack2.*;
pack1.Student stud1;
pack2.Student stud2;
Teacher teacher1;
Courses course1;
```

Package Access

- Default
- Private
- Public
- Protected

Location/Access Modifier	Private	Default	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Packages and class path

- **Case 1**
  - We shall write a class called Circle in package com.capgemini.
  - Suppose that we save the source as d:\myJavaProject\src\com\capgemini\Circle.java,
  - and the compiled class as d:\myJavaProject\classes\com\capgemini\Circle.class.
  - Let's write the source as follows:
- // d:\myJavaProject\src\com\capgemini\Circle.java

```
package com.capgemini;
 public class Circle { }
```

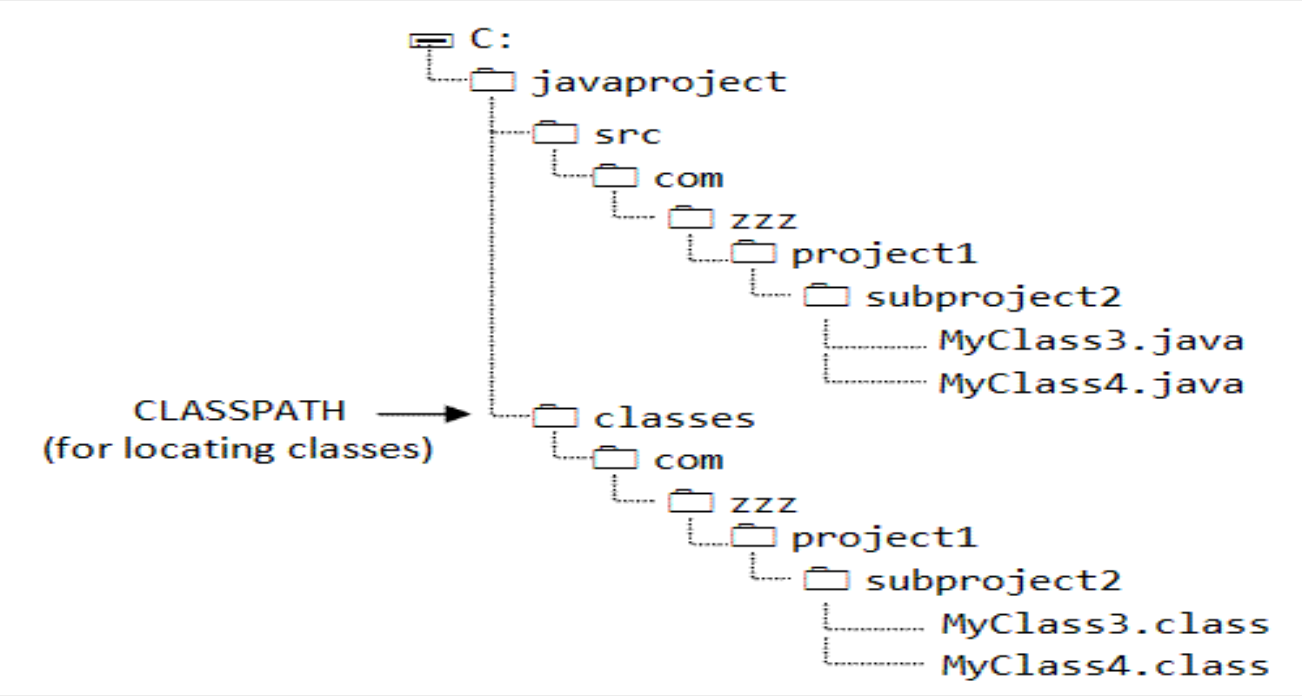
Packages and class path

- To compile the source using JDK, we need to use the -d option to specify the package base directory of the compiled class d:\myJavaProject\classes as follows (-d defaulted to the current directory):

```
> javac -d d:\myJavaProject\classes
d:\myJavaProject\src\com\capgemini\Circle.java
```

Packages and class path

- **Case 2:** Source and class files in separate directories
  - Suppose that you decided to keep the source files and classes in separate directories (for distribution of classes without the sources), and the directory structure of your source files and classes is as follows:



Packages and class path

- To compile the source files and place the classes in the desired directory, we can use the "-d" (for destination) command-line option of the Java compiler, which specifies the location of the compiled classes. We also need to specify the CLASSPATH of the classes, as MyClass3 uses MyClass4, as follows:

```
> cd $SRC_BASE_DIR\zzz\project1\subproject2
> javac –d $CLASS_BASE_DIR -classpath .;$CLASS_BASE_DIR
*.java
// try omitting the classpath and compile just MyClass3 which uses
MyClass4
> javac –d $CLASS_BASE_DIR MyClass3.java
```

Importing packages

- Use fully qualified name.

```
java.util.Date = new java.util.Date();
```

- You can use import to instruct Java where to look for things defined outside your program.

```
import java.util.Scanner;
Scanner sc = new Scanner (System.in);
```

You can use multiple import statements

- You can use * to import all classes in package:

```
import java.util.*;
Scanner sc = new Scanner (System.in);
```

Use * carefully; you may overwrite definitions

Static Import

- Static import enables programmers to import static members.
- Class name and a dot (.) are not required to use an imported static member.

```
import static java.lang.Math.*;
public class StaticImportTest
{
 public static void main( args[] )
 {
System.out.printf( "sqrt( 900.0 ) = %.1f\n", sqrt( 900.0 ) );
 } // end main
}
```

Note: It's not Math.sqrt

Some Java Packages

Package Name	Description
java.lang	Classes that apply to the language itself, which includes the Object class, the String class, and the System class. It also contains the Wrapper classes. "Classes belonging to java.lang package need not be explicitly imported".
java.util	Utility classes, such as Date, as well as collection classes, such as Vector and Hashtable
java.io	Input & output classes for writing to & reading from streams (such as standard input and output) & for handling files
java.net	Classes for networking support, including Socket and URL (a class to represent references to documents on the WWW)
java.applet	Classes to implement Java applets, including the Applet class itself, as well as the AudioClip interface

Demo : Package

- Execute the following programs:
  - Balance.java
  - AccountBalance.java
  - StaticImportDemo.java
  - StaticImportNotUsed.java

Summary

- In this lesson you have learnt:
  - Packages
  - Access Specifiers
  - Importing packages
  - Some Java Packages

Review Questions

- Question 1: Which of the following are the benefits of using Package?
  - **Option1:** prevents name-space collision.
  - **Option2:** To implement security of contained classes.
  - **Option3:** Better code library management.
  - **Option4:** To increase performance of your class.
- Question 2: Which of these access specifiers can be used for a class so that it's members can be accessed by a different class in the different package?
  - **Option1:** Public
  - **Option2:** Protected
  - **Option3:** Private
  - **Option4:** No Modifier