COCSIT

INTRODUCTION TO PACKAGE

Introduction to Packages

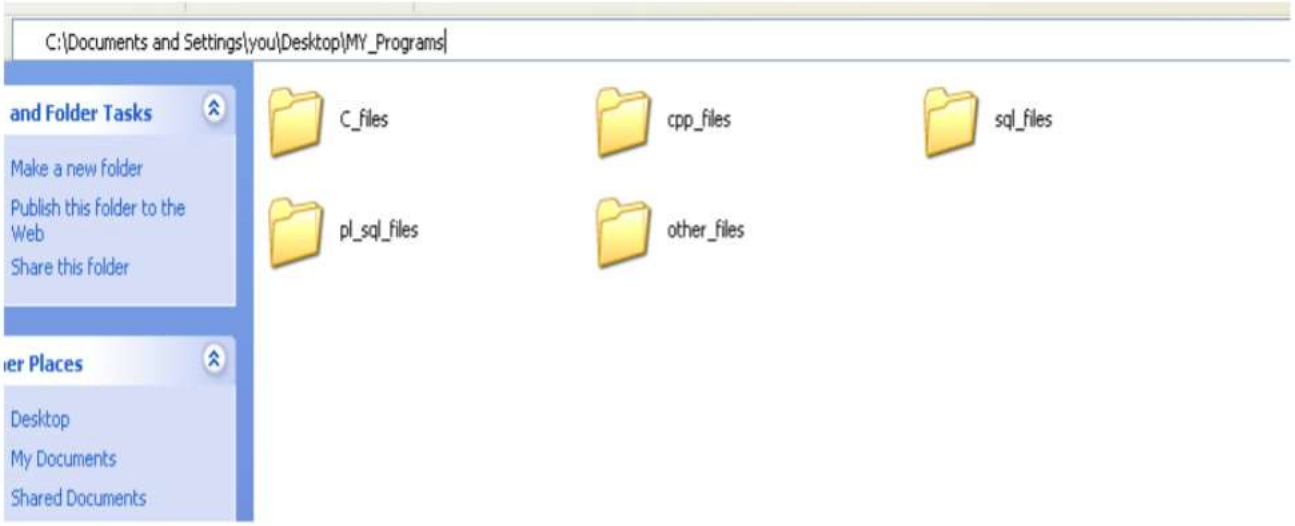
Agenda

1 Introduction to Packages

COCSIT

Introduction to Packages

Package is similar to folders in your Disk



Just relate package concept with directories concept in your file system.
The advantage is we can easily locate the files if they are organized.

Organizing classes into Packages

- Packages are containers for classes and interfaces
- Example:

```
package MyPackage;
class MyClass {// code}
class YourClass{// more code }
```

- Classes and interfaces are grouped together in containers called **packages**
- To avoid namespace collision, we put classes and interfaces into containers called **packages!**
- Whenever you need to access a class, you access it through its package by prefixing the class with the package name

Packages & Access Control

Specifier	Accessibility
private	Accessible in the same class only
No-specifier (default access)	Subclasses and non-subclasses in the same package
protected	Subclasses and non-subclasses in the same package, and subclasses in other packages!
public	Subclasses and non-subclasses in the same package, **as well as** subclasses and non-subclasses in other packages. So, Any class can access from anywhere..

Access Specifiers in a Nutshell

Access Specifier	Private access	Default Access	Protected Access	Public access
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

Access Control

Specifier	Accessibility
private	same class only
No-Keyword (default access)	same package only
protected	same package and subclasses
public	Anywhere in the program

Inbuilt Packages in java

java.lang, java.io, java.util, java.awt, java.applet, java.sql, javax.swing (more packages are there) are some of the in-built packages.

- java.lang – Basic package which is automatically imported in all programs.
  - PrintWriter, String, StringBuilder, StringBuffer,
  - All Wrapper Classes // **(totally 8 – can u list them?)**
  - Throwable
  - Exception
  - Thread
  - Runnable
- java.io –Input / Output related classes are available here.
  - Scanner //(Why we need this ?)
  - File , FileReader , FileWriter
  - BufferedReader
  - InputStreamReader
  - IOException, FileNotFoundException etc

Inbuilt Packages(contd.).

- java.util – Utility classes are available here. We can use these ready-made classes.
  - ArrayList
  - Set
  - HashMap
  - **Date** (to work with Date)
  - Calendar (improved one)
  - Stack (LIFO) , Queue (FIFO) // expand these
  - Vector ,

- java.sql –for JDBC programming
  - Various classes like Connection,
  - DriverManager,
  - ResultSet,
  - SQLException are available here.

Quiz

Which is not a correct inbuilt java package?

A) java.io

B) java.sql

C) java.dbms

D) java.net

Option ?
Find which are valid java packages.

Quiz

Which is a correct inbuilt java package?

A) java.text

B) java.errors

C) java.dbms

C) java.network

All are invalid java packages.

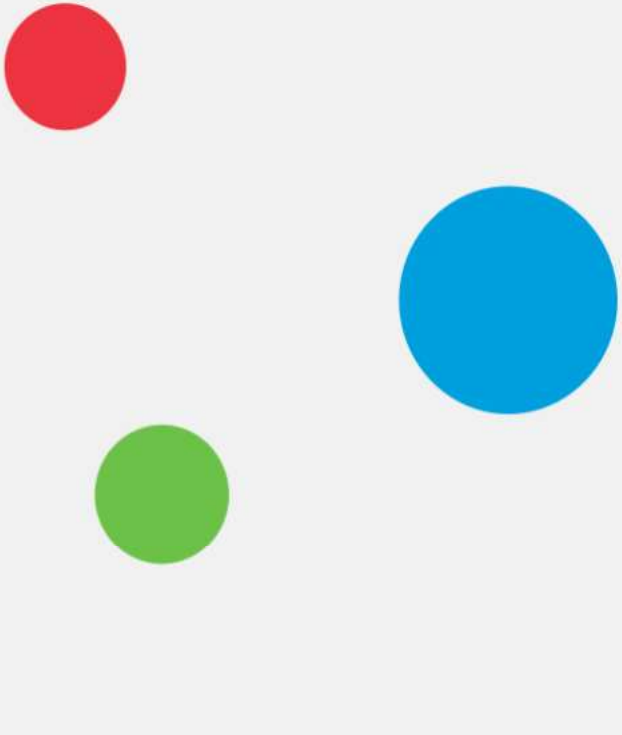
Summary

In this session, you were able to :

- Learn about packages

Import, Static Import & Creating our own Packages

Importing Classes



Packages & import statement

- Java provides import statement.
  - Import means, we are including the classes and interfaces of existing packages in our program.
  - If you need a sub package, then, you need to issue a separate import statement.

- For example,
  - `import java.awt.*;` -- this will be importing awt package
  - `import java.awt.event.*;`
    - this will be importing event package which is a sub package under awt package.

Importing Classes from Packages

- Java has used the package mechanism extensively to organize classes with similar functionality in one package
- If you want to use these classes in your applications, you can do so by including the following statement at the beginning of your program:
  - `import packagename.classname;`
- *If the packages are nested you should specify the hierarchy.*
  - `import package1.package2.classname;`

Importing Classes from Packages (Contd.).

- The class you want to use must be qualified by its package name.
- If you want to use several classes from a package, it would be cumbersome to type so many classes qualified by their packages.
- It can be made easy by giving a star(*) at the end of the import statement. For example:

```
import  package1.*;
```

Static Import

- A static import declaration enables us to refer to imported static members as though they were declared in the current class

- If we use static import, we first have to import this static member in the following way :

```
package p1;
public class Abc {
     public static void xyz() {
     System.out.println("static import demo");
     }
}
```

```
package p2;
import static p1.Abc.xyz;
public class A1 {
     public static void main(String[] args) {
          xyz();
     }
}
```

Output : "static import demo"

Static Import (Contd.).

- If we are invoking multiple static members of the same class, we can also use asterisk(*), which indicates that *all* static members of the specified class should be available for use

```
import static java.lang.Math.*;
public class StaticImportDemo  {
     static float x = 4.556f;
     static double y =4.556D;
     public static void main( String args[] )   {
       float a1 = abs(x);
       int r1  = round(x);
       double s1 = sqrt(y);
       System.out.println("absolute value of "+x+" is" +a1);
       System.out.println("When we round off "+x+"we get" +r1);
       System.out.println("Square Root of "+y+ "is" +s1);
     }
  }
```

Quiz

Which is the correct usage of import statement?

A) import java.*;

B) import java.lang.*;

C) import *;

D) import *.*;

Which Option is correct ?

Quiz

In one java source file, how many package statements can be used?

A) One

B) Two or more

Only Option A is correct;
You can't have two or more package statements in a java source file

Creating our own Packages

We can create our own packages in java

- Package statement helps us to create our own package.
- Package statement should be the first statement in your program.
- We group related classes and interfaces into a package
- We can have sub-packages inside our packages as required

Packages are stored as directories in Hard disk:

- Remember, the case should match exactly
- Look at the program in next page & try it from command line:

Working with Packages – Example 1

```
package automobile;

public class Vehicle {
public void printname() {
        System.out.println("My name is vehicle");
        System.out.println(" I am defined inside automobile
  package");
    }
}
```

What is the package name?
How you will save this file?

Working with Packages – Example 1 (Contd.).

```
package automobile;

public class Bike extends Vehicle {
    public void printname() {
    System.out.println("My name is bike");
    System.out.println(" I am defined inside automobile
  package");
    }
}
```

What is the package name?
How you will save this file?

Working with Packages – Example 1 (Contd.).

```
package automobile;

public class Car extends Vehicle {
    public void printname() {
        System.out.println("My name is car");
        System.out.println(" I am defined inside
  automobile package");
    }
}
```

What is the package name?
How you will save this file?

Working with Packages – Example 1 (Contd.).

```
package au_test;
import automobile.*;
public class tester {
public static void main(String s[ ] ) {
System.out.println(" I am tester class defined inside au_tester
  package");
System.out.println(" I had imported all classes of automobile
  package");
System.out.println(" Creating instances of Vehicle, Car and Bike ");
```

Working with Packages – Example 1 (Contd.).

```
Vehicle v = new Vehicle();
Car c = new Car();
Bike b = new Bike();
System.out.println(" Accessing the functions using objects");
 v.printname();
 c.printname();
 b.printname();
}
}
```

How you will save this file?
Then, In command prompt:
How you will compile?
And How you will run?

What is the output of the program?

Summary

In this session, you were able to learn about:

- Import
- Static Import
- Creating Our own packages

Thank You