Object Oriented Programming with Java

(BCS – 403)

NOTES

[Unit – I: Part – 4]

Packages

Prepared By:

Amol Sharma,
Assistant Professor (CSE),
MIT, Meerut.

Object Oriented Programming with Java

Package

"A package is a collection of related Java elements such as classes, interfaces etc."

Uses of Packages in Java

1. Namespace Management

Packages are used to *divide the class name space in to compartments*. A class name must not be repeated in a compartment but it may be repeated/ reused in other compartments.
Naming conflict of classes may be resolved by prefixing the class name with a package name.

Example:
pack1.Rectangle and pack2.Rectangle are two distinct classes. Although they share the same class name Rectangle, but they belong to two different packages: pack1 and pack2.

Object Oriented Programming with Java

Package

Uses of Packages in Java

2. Access Control

We can define classes inside a package that are not accessible by code outside that package. We can also define class members that are exposed only to other members of the same package.

3. Reuse and Distribution

Packages group related reusable classes and interfaces together. This collection may be distributed as *Java Archive (JAR)* files for easy sharing and reuse.

Object Oriented Programming with Java

Package

Package Name & Directory Structure

In Java, package names are mapped to the directory structure used to store the class files.

Example:

If a class Rectangle belongs to the package p1.pack1.project1, it is stored as: *p1\pack1\project1\Rectangle.class* in the current working directory.

Packages with Common Prefixes

If we have package names like com.school and com.school.exam.
Although their names seem related, they are two different packages. They just share part of the directory path.
The classes in the package com.school are stored in com\school\ and the classes in the package com.school.exam are stored in com\school\exam\.

Object Oriented Programming with Java

Package

Naming Convention for Packages

In Java, packages are named using a naming convention to avoid conflicts between different organizations and projects.

- Use Your Organization's Internet Domain Name in Reverse

  Example, if your company's domain is example.com, you should start your package name with com.example.

  Example:

  com.example.myapp

- Use all Lowercase Letters, package names are written in all small letters.
- Choose meaningful names after the reversed domain, use descriptive names of the project or module.

  Example:

  com.example.shopping.cart

Object Oriented Programming with Java

Package

Defining a Package

A package in Java is defined *using the package keyword as the first statement* in the source file. All the classes and interfaces in that file become part of the specified package.

Example:

```
package mypack;
```

Note:

- If no package is defined, the class belongs to the default package (unnamed).
- Multiple files can belong to the same package by using the same package statement.
- Directory name must match the package name exactly (Java is case-sensitive). For *package a.b.c;* , compiled .class files should be stored in the folders on the path *a\b\c*.

Object Oriented Programming with Java

Package

Compiling Java Classes into Package Directories

Two Ways to Organize .class Files in Packages:

1. *Manual Method*

   Compile the Java file normally

   javac MyClass.java

   Create the required directory structure matching the package name.

   For *package mypack;* create mypack/ directory.

   Move the .class file manually to mypack directory.

2. *Using -d Option (Recommended)*

   Compile and automatically create the package directory structure.

   javac -d classes MyClass.java

   If MyClass.java contains *package mypack;*

   The file is stored as:

   classes/mypack/MyClass.class

Object Oriented Programming with Java

Package

CLASSPATH Setting for Packages

To run Java programs that use user-defined packages, the Java runtime needs to know where to find those packages.

Three Ways to find the packages:

1. Default: Current Working Directory
   By default, Java looks in the current working directory.
   If your package is in a subdirectory of the current directory, Java will find it automatically.

2. Using the CLASSPATH Environment Variable
   You can define the path to your package directory using the system's CLASSPATH environment variable.
   Important: The CLASSPATH should contain the path up to the package folder, not the package name itself.

Object Oriented Programming with Java

Package

CLASSPATH Setting for Packages

Three Ways to find the packages:

2. Using the CLASSPATH Environment Variable
   Example:
   If the package is in D:\docs\JavaPrograms\Java\classes\mypack then set
   CLASSPATH = D:\docs\JavaPrograms\Java\classes

3. Using '-classpath' or '–cp' Option
   While compiling and running, you can specify the class path
   using the -classpath or -cp option.
   Example:
   java -classpath D:\docs\JavaPrograms\Java\classes mypack.MyClass

Object Oriented Programming with Java

Package

Access Control

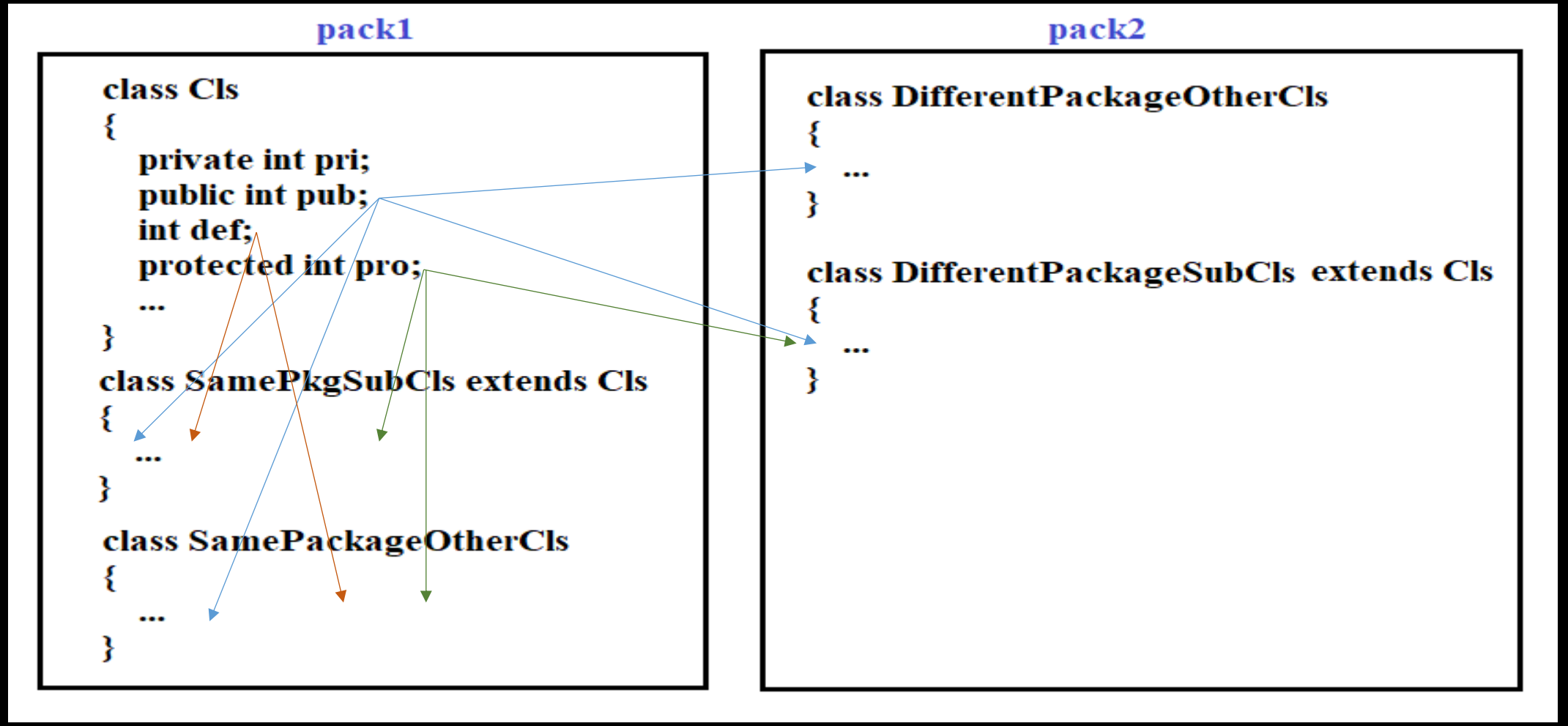
- *public* members of a class can be accessed from different classes and different packages.
- *private* members cannot be seen outside of its class.
- When a member has *no access specifier*, it is visible to all the other classes in the same package. This is the *default access*.
- *protected* members of a class can be accessed by any other class of the package as well as by the subclasses in other packages.

	Default	Private	Protected	Public
Same Class	Yes	Yes	Yes	Yes
Same Package Subclass	Yes	No	Yes	Yes
Same Package Non-Subclass	Yes	No	Yes	Yes
Different Package Subclass	No	No	Yes	Yes
Different Package Non-Subclass	No	No	No	Yes

Object Oriented Programming with Java

Package

Access Control



Object Oriented Programming with Java

Package

Access Control

A class or an interface has only two possible access levels: default and public.

- When a class/ interface is declared as public, it is accessible outside its package.
  When a class is public, it must be the only public class declared in the file, and the file must have the same name as the class.
- If a class/ interface has default access, then it can only be accessed by other code within its same package.

Object Oriented Programming with Java

Package

Importing Packages

In Java, classes and interfaces from packages must be imported before they can be used in a program (unless they belong to the same package as the current class). This is done using the import statement.

- import statements must appear after the package statement (if any) and before any class definitions.
- Classes in the same package can be accessed without import.
- Import can be used to access built-in classes from built-in packages like java.util, java.io, etc.
- The java.lang package is automatically imported in every Java program and contains fundamental classes such as String, System etc.

Example:

```
import java.util.Scanner;  // Imports only Scanner class
import java.util.*; //Imports all classes in java.util)

import mypackage.MyClass;  // User-defined package and class
import mypackage.*;      // Imports all classes from mypackage
```

Object Oriented Programming with Java

Package

Static Import

Static import allows members (fields and methods) defined as static in other classes to be used without class qualification.
It is useful when we frequently use static members like constants or utility methods in the code.

Example:

```
import static java.lang.Math.PI;

// Imports all static members like sqrt, pow, PI, etc.
import static java.lang.Math.*;

public class Test {
    public static void main(String[] args) {
        System.out.println(PI);   // No need to write Math.PI

        // No need to write Math.sqrt(16)
        System.out.println(sqrt(16));
    }
}
```

Object Oriented Programming with Java

Package

Making JAR Files for Library Packages

A JAR (Java Archive) file bundles together multiple .class files, images, and other resources into a single compressed file for distribution and reuse.

Creating a JAR File

jar cf MyLib.jar mypack

c → Create a new JAR
f → Specify JAR file name
mypack → Folder containing .class files (package)

Using the JAR in Another Program

Compile source file (.java file) using classes available in the JAR file specified in the classpath.

```
javac -cp utility.jar Main.java
```

Run the compiled class, setting the classpath to include the current directory (.) and the JAR file.

```
java -cp .;utility.jar Main
```

Object Oriented Programming with Java

Comments

Comments are used to make code more understandable. They are ignored by the compiler and do not affect the execution of the program.

Types of Comments:

Single-line Comment (//):

It is used for brief explanations or notes spanning only a single line.

Example:

```
// This is a single-line comment
int x = 10; // declaring a variable
```

Multi-line Comment (/ ... */)*

It is used for longer descriptions spanning multiple lines or to temporarily disable code.

Example:

```
/* This is a
 multi-line comment */
```

Object Oriented Programming with Java

Command Line Arguments

These are the arguments (values) passed to the main() method during program execution from the command line.

- All command line arguments are stored as String objects in the args[] array.
- args.length gives the number of arguments.
- Arguments are indexed from 0.

Example:

```
public class Example {
    public static void main(String[] args) {
        System.out.println("First Argument: " + args[0]);
        System.out.println("Second Argument: " + args[1]);
    }
}
```

Executing program with Command Line Arguments

> java Example Hello Java