



# CS205 Object Oriented Programming in Java

## Module 3 - More features of Java (Part 1)

**Prepared by**

**Renetha J.B.**

AP

Dept.of CSE,

Lourdes Matha College of Science and Technology

# Topics



- Introduction:
  - ☑ **Packages** and Interfaces:
    - ☑ Defining Package,
    - ☑ CLASSPATH,
    - ☑ AccessProtection
    - ☑ Importing Packages

# Package



- **Packages** *are containers for classes.*
- A package in Java is used to **group** related classes and interfaces.
- They are used to keep the class name space compartmentalized.
  - For example, a package allows us to create a class named **List**, which we can store in our own package and it will not collide with some other class named **List** stored elsewhere.
- Packages are stored in a *hierarchical manner.*
- The package is both a **naming** and a **visibility control mechanism.**

# Packages(contd.)



- We can define classes inside a package
  - that are not accessible by code outside that package.  
(default)

OR

- that can be also accessed by subclasses outside the package. (protected)

OR

- That can be accessed by all classes in all packages(public)

# Defining Package



- To **create a package**, simply include a **package command** as the first statement in a Java source file.
  - All classes declared in that file will belong to the specified package.
- The package statement **defines a name space** in which classes are stored.
- If we are **not** writing package statement, the class names are put into the *default package, which has no name*.

# Defining Package(contd.)



- General form for creating a **package** :  
**package** *packagename*;

*Example:* If we write the following statement at the beginning of our java program then it will create a package named **Oop**.

*package Oop;*

# Defining Package(contd.)



- Java uses file system **directories** to store packages.
- Example: Any classes that we declare to be part of the package **Oop** must store their **.class** files in a directory called **Oop**.
- Any file can include the same package statement.
- The package statement simply specifies to which package the classes defined in a file belongs to.

# Defining Package(contd.)



- We can create a **hierarchy of packages**.
  - Separate each package name from other using period(dot) symbol.
- General form of a multileveled package statement is :

**package** *pkg1.pkg2.pkg3*;

This specifies that package *pkg3* is inside package *pkg2* and *pkg2* package is inside *pkg1*.

- E.g The package declared as

**package java.awt.image**;

- needs to be stored in the path **java\awt\image** in a Windows environment
- We cannot rename a package without renaming the directory in which the classes are stored.



# Finding Packages and CLASSPATH



❖ How does the Java run-time system know where to look for packages that we create?

1. By default, the Java run-time system uses the **current working directory** as its starting point.

→if our package is in a subdirectory of the current directory, it will be found.

2. We can specify a directory path or set paths by setting the CLASSPATH environmental variable.

3. We can use the **-classpath** option with java and javac to specify the path to your classes.

# CLASSPATH (contd.)



- Example  
package MyPack;
- For a program to find MyPack, one of three things must be true.
  - Either the program can be executed from a directory immediately above **MyPack** **or**
  - the **CLASSPATH** must be set to include the path to **MyPack**, **or**
  - the **-classpath option** must specify the path to MyPack when the program is run via **java**
- To execute the program
  - java MyPack.programname

# CLASSPATH9contd.)



- In the case of CLASSPATH and –classpath option , the class path *must not include MyPack, itself*. It must simply specify the *path to MyPack*.
- Suppose the path of MyPack directory is **C:\MyPrograms\Java\MyPack**
  - Then the class path to **MyPack** is **C:\MyPrograms\Java**

# Access Protection



- Addresses four categories of visibility for class members:
  - Subclasses in the same package
  - Non-subclasses in the same package
  - Subclasses in different packages
  - Classes that are neither in the same package nor subclasses

# Access Protection(contd.)



	Private	No Modifier	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

# Access Protection(contd.)



- A **non-nested class** has only two possible access levels:
  - default
  - **public**.
- When a class is declared as **public**, it is accessible by any other code.

```
public class A {////  
}
```

- If a class has **default** access, then it can only be accessed by other code within its same package.

```
class B  
{ }
```

- 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 public class.

# Importing Packages



- All of the standard classes are stored in some named package.
- If we want to use classes in some other packages, they must be *fully qualified with their package name or names*.. It is **difficult** to type in the long dot-separated package path name for every class we want to use.
  - **TO SOLVE THIS PROBLEM**, we can use **import** statement. The **import** statement helps to bring certain classes, or entire packages, into visibility.
- To use a class or a package from the library, we need to use the **import** keyword
- **import** statements is written **after** the package statement(if exists) and **before** all class definitions.

# Importing Packages(contd.)



- General form of the import statement:

**import** *pkg1*[*.pkg2*].(*classname*|\*);

- Here, *pkg1* is the name of a top-level package, and *pkg2* is the name of a subordinate package inside the package *pkg1* separated by a dot (.). Here square bracket denotes that it is optional.

- E.g.

`import pack1;`      *// import the package pack1*

`import java.io.*;`      *// import all the classes from the package java.io*

`import java.util.Date;` *//import the Date class from the package java.util*



# Importing Packages(contd.)



- All of the standard Java classes included with Java are stored in a package called **java**
- The basic language functions are stored in a package inside of the java package called **java.lang**
  - it is implicitly imported by the compiler for all programs.

# Importing Packages(contd.)



- Using an **import** statement:

```
import java.util.*;  
class MyDate extends Date {  
    //statements , methods,variables  
}
```

- Without the import statement looks like this:

```
class MyDate extends java.util.Date  
{  
}
```

Without Using import statement- we have to use class from other package as packagename.classname (fully quantified)



**//Program A.java**

```
package pack1;  
public class A  
{  
    int a=100;  
    public int c=20;  
    protected int d=50;  
        public void msg()  
        {  
            System.out.println("Base class A Hello");  
        }  
}
```

**//Program B.java**

```
package pack2;  
class B{  
    public static void main(String args[])  
    {  
        pack1.A obj = new pack1.A();  
        obj.msg();  
        System.out.println("c="+obj.c);  
        //System.out.println("d="+obj.d);  
        // cannot access protected of  
        //different package i.e. pack1  
        //System.out.println("a="+obj.a);  
        //cannot access private of other class  
    }  
}
```

Using **import package.\*** statement to import all classes in pack1 to program file in pack2



**//Program A.java**

```
package pack1;  
public class A  
{  
int a=100;  
public int c=20;  
protected int d=50;  
public void msg()  
{  
System.out.println("Base class A Hello");  
}  
}
```

**//Program B.java**

```
package pack2;  
import pack1.*;  
class B{  
public static void main(String args[])  
{  
    A obj = new A();  
    obj.msg();  
    System.out.println("c="+obj.c);  
//System.out.println("d="+obj.d);  
// cannot access protected of different package  
pack1  
//System.out.println("a="+obj.a);  
//cannot access private of other class  
}  
}
```

Using **import package.classname** statement to  
import class A in pack1 to program file in pack2

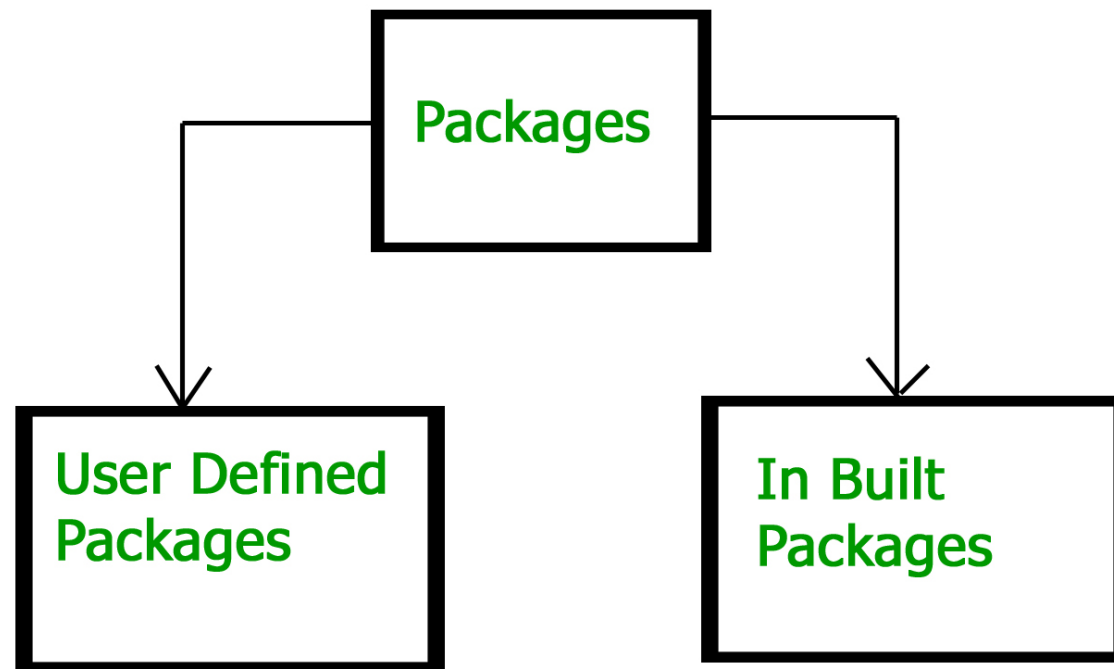


**//Program A.java**

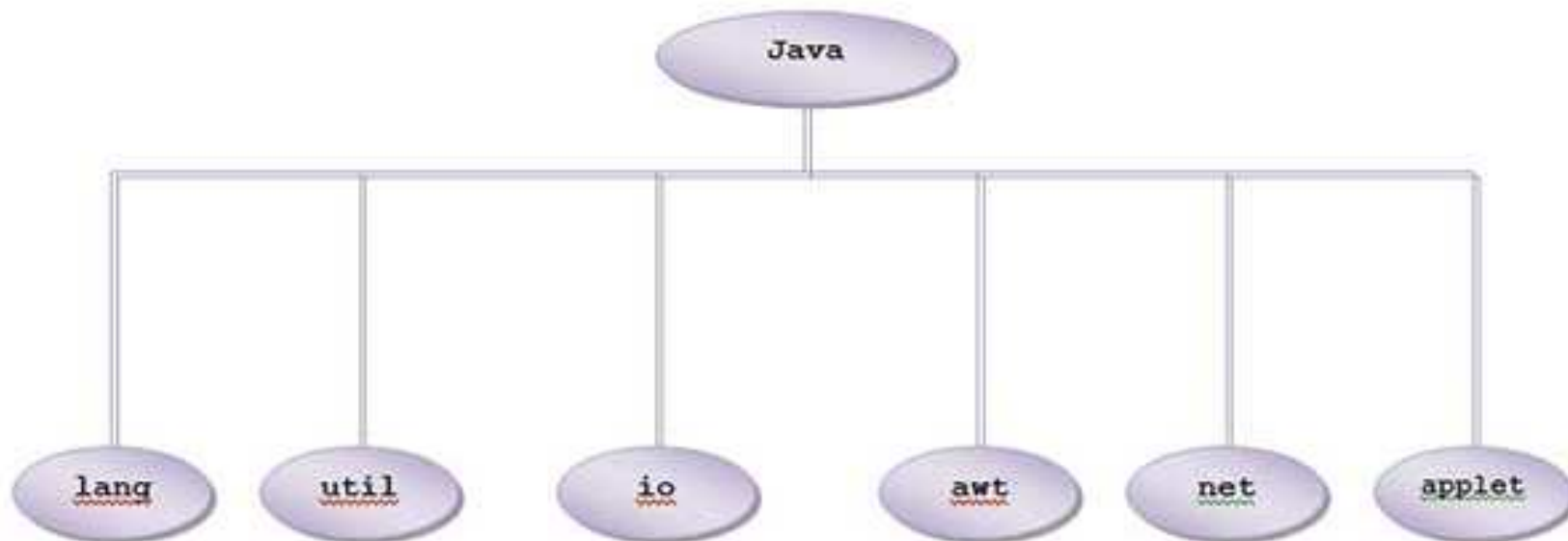
```
package pack1;  
public class A  
{  
int a=100;  
public int c=20;  
protected int d=50;  
public void msg()  
{  
System.out.println("Base class A Hello");  
}  
}
```

**//Program B.java**

```
package pack2;  
import pack1.A;  
class B{  
public static void main(String args[])  
{  
    A obj = new A();  
    obj.msg();  
    System.out.println("c="+obj.c);  
//System.out.println("d="+obj.d);  
// cannot access protected of different package  
pack1  
//System.out.println("a="+obj.a);  
//cannot access private of other class  
}  
}
```



# Built-in Packages



# Java Foundation Packages



- Java provides a large number of classes grouped into different packages based on their functionality.
- The six foundation Java packages are:
  - **java.lang**
    - Contains classes for primitive types, strings, math functions, threads, and exception
  - **java.util**
    - Contains classes such as vectors, hash tables, date etc.
  - **java.io**
    - Stream classes for I/O
  - **java.awt**
    - Classes for implementing GUI – windows, buttons, menus etc.
  - **java.net**
    - Classes for networking
  - **java.applet**
    - Classes for creating and implementing applets





- Steps and examples for creating and using packages



- Create a folder **pack1** inside D drive
- Create a file A.java

```
package pack1;  
public class A  
{  
    public static void main(String args[] )  
    {  
        System.out.println("Hello");  
    }  
    public void show()  
    { System.out.println("show in A");  
    }  
}
```



## Method 1

- Take *path before pack1* folder in command prompt here it s D drive.

- **Compile** using

D:\>**javac** pack1/A.java

- **To run**

D:\>**java** pack1/A

Or

D:\>**java** pack1.A



## **Method 2**

- Set **classpath** in command prompt to path to folder before the package pack1

```
C:\Users\USER>set CLASSPATH=;D:\
```

### **To compile**

```
C:\Users\USER>javac -cp . D:\pack1\A.java
```

### **To run**

```
C:\Users\USER>java pack1.A
```

**Hello**



### Method 3:

- Using **-classpath** option
- Compile using

```
C:\Users\USER>javac D:\pack1\A.java
```

Or

```
C:\Users\USER>javac -classpath . D:\pack1\A.java
```

- Run using

```
C:\Users\USER>java -classpath D:\ pack1.A
```

## E.g using import statement



- Create a folder **pack2** inside D drive
- Create a file B.java in it

```
package pack2;  
import pack1.*;  
class B{  
public static void main(String args[])  
{  
    A obj = new A();  
    obj.show();  
    System.out.println("main in class B");  
}  
}
```

```
D:\>javac pack1\A.java
```

```
D:\>javac pack2\B.java
```

```
D:\>java pack2.B  
show in A  
main in class B
```

# Reference



- **Herbert Schildt, Java: The Complete Reference, 8/e, Tata McGraw Hill, 2011.**