Object Oriented Programming

Course Instructor: Sadaf Anwar

Packages

Points to be covered today

- Packages Introduction
- Defining a package
- Finding packages and CLASSPATH
- Access Protection
- Import Packages

Introduction

Packages in Java are groups of similar types of **classes**, **interface** and **sub packages**. It is a way of grouping a variety of classes or interfaces collectively.

- Define classes inside a package that are not accessible by code outside that package
- Define class members that are exposed only to other members of the same package
- This allows classes to have intimate knowledge of each other
  - – Not expose that knowledge to the rest of the world

Benefits of organizing classes

1. In packages, classes can be declared uniquely compared with classes in other packages.
2. Java Packages provide a way to **'hide'** classes thus preventing other programs or packages from accessing classes that are meant for internal use only.
3. Packages provide **access protection**.
4. Java package removes **naming collision**.

Syntax and Defining a package

- `Syntax: package pkg;`

Simple Example

```
package mypack;
public class Simple{
 public static void main(String args[]){
   System.out.println("Welcome to package");
  }
}
```

Packages Example

```java
package mypackage;

class Balance {
    String name;
    double bal;
    Balance(String n, double b) {
        name = n;
        bal = b;
    }
    void show() {
        System.out.println(name + ": $" + bal);
    }
}
public class AccountBalance {
    public static void main(String[] args) {
        Balance [] current = new Balance[3];
        current[0] = new Balance( n: "K. J. Fielding",  b: 123.23);
        current[1] = new Balance( n: "Will Tell",  b: 157.02);
        current[2] = new Balance( n: "Tom Jackson",  b: -12.33);
        for (Balance b : current) {
            b.show();
        }
    }
}
```

javac -d . AccountBalance.java

java mypackage.AccountBalance

How to access package from another package?

There are three ways to access the package from outside the package.

1. import package.*;
2. import package.classname;
3. fully qualified name.

Packages Syntax of multileveled

- The general form of a multilevel package statement
  - ***package pkg1[.pkg2[.pkg3]]***
  - ***package java.util.concurrent***
- import statements occur immediately following the package statement and before any class definitions
- The general form of the import statement
  - ***import pkg1 [.pkg2].(classname | *)***
  - ***import java.util.Scanner***
  - import statement is optional, class can be used with name that includes full package hierarchy

1) Using packagename.*

```
1.  package pack;
2.  public class A{
3.    public void msg(){
4.  System.out.println("Hello");
5.        }
6.  }
```

```
1.  package mypack;
2.  import pack.*;
3.
4.  class B{
5.    public static void main(String args[]){
6.      A obj = new A();
7.      obj.msg();
8.    }
9.  }
```

2)Using packagename.classname

```
1. package pack;
2. public class A{
3.   public void msg(){System.out.println("Hello");}
4. }
```

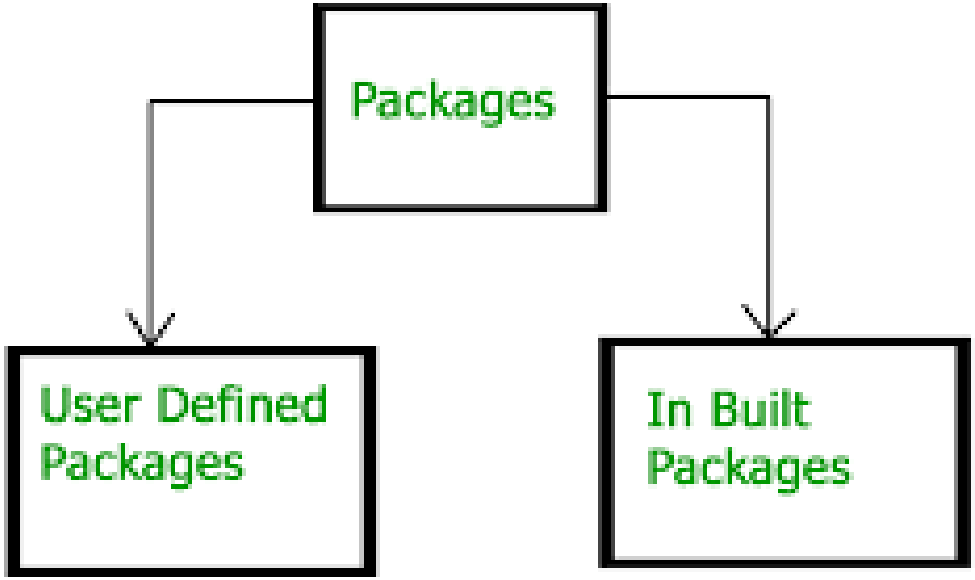
```
1.package mypack;
2.import pack.A;
3.
4.class B{
5.  public static void main(String args[]){
6.   A obj = new A();
7.   obj.msg();
8.  }
9.}
```

3) Using fully qualified name

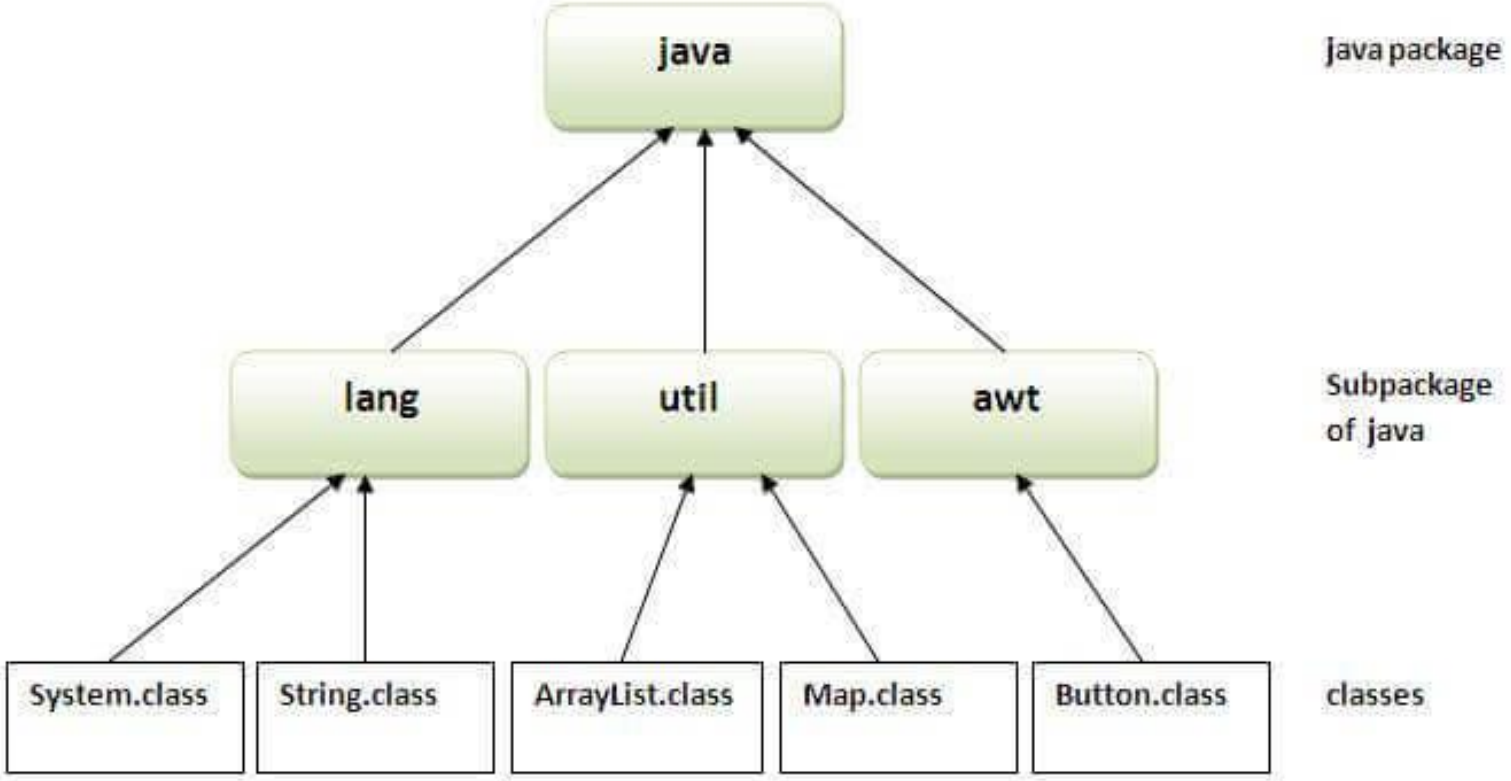
```
1. package pack;
2. public class A{
3.   public void msg(){System.out.println("Hello");}
4. }
```

```
1.package mypack;
2.class B{
3.  public static void main(String args[]){
4.   pack.A obj = new pack.A(); //using fully qualified name
5.   obj.msg();
6.  }
7.}
```

Types of Packages



Built in Packages



Import a Class

- Syntax:

1. import package.name.Class; // Import a single class
2. import package.name.*; // Import the whole package

Simple Example of Built in class

```
import java.util.Scanner;
class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new
Scanner(System.in);
    System.out.println("Enter username");


    String userName = myObj.nextLine();
    System.out.println("Username is: " +
userName);
  }
```

Import a Package

```
import java.util.*;
```

How packages work?

- Package names and directory structure are closely related. For example if a package name university.*staff*.*cs*, then there are three directories, *university*, *staff* and *CS* such that *cs* is present in *staff* and *staff* is present *university*.
- university.*staff*.*SE*
- university.*staff*.*EE*
- university.*staff*.*Maths*

Sub package

- Packages that are inside another package are the **subpackages**.
- These are not imported by default, they have to imported explicitly.

```
1. package letmecalculate;
2. public class Calculator {
3.    public int add(int a, int b){
4.   return a+b;
5.    }
6.    public static void main(String
   args[]){
7.   Calculator obj = new Calculator();
8.   System.out.println(obj.add(10, 20));
9.    }
10.}
```

```
1. import letmecalculate.Calculator;
   public class Demo{
2. public static void main(String
   args[]){
3.  Calculator obj = new Calculator();
4. System.out.println(obj.add(100,
   200)); }
5. }
```

```
1. package letmecalculate.multiply;
2. public class Multiplication {
3. int product(int a, int b){
4.  return a*b;
5. }
6.  }
```

Access Protection

- The three access modifiers provide a variety of ways to produce the many levels of access required
  - private, public, and protected
- The following applies only to members of classes

	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

Points to remember:

- 1. Sometimes class name conflict may occur. For example: Lets say we have two packages abcpackage and xyzpackage and both the packages have a class with the same name, let it be JavaExample.java. Now suppose a class import both these packages like this:

```
import abcpackage.*;
import xyzpackage.*;
```

```
abcpackage.JavaExample obj = new
abcpackage.JavaExample();
xyzpackage.JavaExample obj2 = new
xyzpackage.JavaExample();
```

- This way you can avoid the import package statements and avoid that name conflict error.

2.. A class can have only one package declaration but it can have more than one package import statements. For example:

```
package abcpackage; //This should be one
import xyzpackage;
import anotherpackage;
import anything;
```

3.The wild card import like package.* should be used carefully when working with subpackages. For example: Lets say: we have a package **abc** and inside that package we have another package **foo**, now **foo** is a subpackage.

- classes inside abc are: Example1, Example 2, Example 3
  classes inside foo are: Demo1, Demo2
- So if I import the package **abc** using wildcard like this:

```
import abc.*;
```

- So to import all the classes present in package and subpackage, we need to use two import statements like this:
- `import abc.*;`
- `import abc.foo.*;`

Summary

Questions?