Java Lecture-9

Packages

- *Packages* are containers for classes.

- They are used to keep the class name space compartmentalized.

- For example, a package allows you to create a class named **List**, which you can store in your own package without concern that it will collide with some other class named **List** stored elsewhere.

- Inside a package unique name had to be used for each class to avoid name collisions.

- Packages are stored in a hierarchical manner and are explicitly imported into new class definitions.

- ❖ Java provides a package mechanism for partitioning the class name space into more manageable chunks.

- ❖ The package is both a naming and a visibility control mechanism.

- ❖ You can define classes inside a package that are not accessible by code outside that package.

- ❖ You can also define class members that are exposed only to other members of the same package.

- ❖ This allows your classes to have intimate knowledge of each other, but not expose that knowledge to the rest of the world.

Defining a Package

This is the general form of the **package** statement:

package *pkg*;

Here, ***pkg*** is the name of the package.

For example, the following statement creates a package called **MyPackage**:

package MyPackage;

- ❖ If you omit the **package** statement, the class names are put into the default package, which has no name.

- ❖ While the default package is fine for short, sample programs, it is inadequate for real applications.

- ❖ Java uses file system directories to store packages. For example, the **.class** files for any classes you declare to be part of **MyPackage** must be stored in a directory called **MyPackage**.

- ❖ More than one file can include the same **package** statement.

- ❖ The **package** statement simply specifies to which package the classes defined in a file belong.

- ❖ You can create a hierarchy of packages.

- ❖ To do so, simply separate each package name from the one above it by use of a period.

- ❖ The general form of a multileveled package statement is shown here:

  package *pkg1*[.*pkg2*[.*pkg3*]];

- ❖ A package hierarchy must be reflected in the file system of your Java development system. For example, a package declared as

  package java.awt.image;

  needs to be stored in **java\awt\image** in a Windows environment. Be sure to choose your package names carefully.

- ❖ You cannot rename a package without renaming the directory in which the classes are stored.

Finding Packages and CLASSPATH

- ❖ Packages are mirrored by directories.

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

The answer has three parts.

- ❖ First, by default, the Java run-time system uses the current working directory as its starting point. Thus, if your package is in a subdirectory of the current directory, it will be found.
- ❖ Second, you can specify a directory path or paths by setting the **CLASSPATH** environmental variable.
- ❖ Third, you can use the **-classpath** option with **java** and **javac** to specify the path to your classes.

A Short Package Example

```
package MyPack;
class Balance {
  String name;
  double bal;
 Balance(String n, double b) {
    name = n;
    bal = b;

}
 void show() {
    if(bal<0)
     System.out.print("--> ");
    System.out.println(name + ": $" + bal);

} }
```

```
class AccountBalance {
  public static void main(String args[]) {
   Balance current[] = new Balance[3];
   current[0] = new Balance("K. J. Fielding", 123.23);
    current[1] = new Balance("Will Tell", 157.02);
    current[2] = new Balance("Tom Jackson", -12.33);

for(int i=0; i<3; i++) current[i].show();

} }
```

- ❖ Call this file **AccountBalance.java** and put it in a directory called **MyPack.**
- ❖ Next, compile the file. Make sure that the resulting **.class** file is also in the **MyPack** directory.

- ❖ Then, try executing the **AccountBalance** class, using the following command line:

  java MyPack.AccountBalance

- ❖ Remember, you will need to be in the directory above **MyPack** when you execute this command.

- ❖ **AccountBalance** is now part of the package **MyPack**. This means that it cannot be executed by itself. That is, you cannot use this command line:

  java AccountBalance

- ❖ **AccountBalance** must be qualified with its package name.

Access Protection

- ❖ Packages act as containers for classes and other subordinate packages.

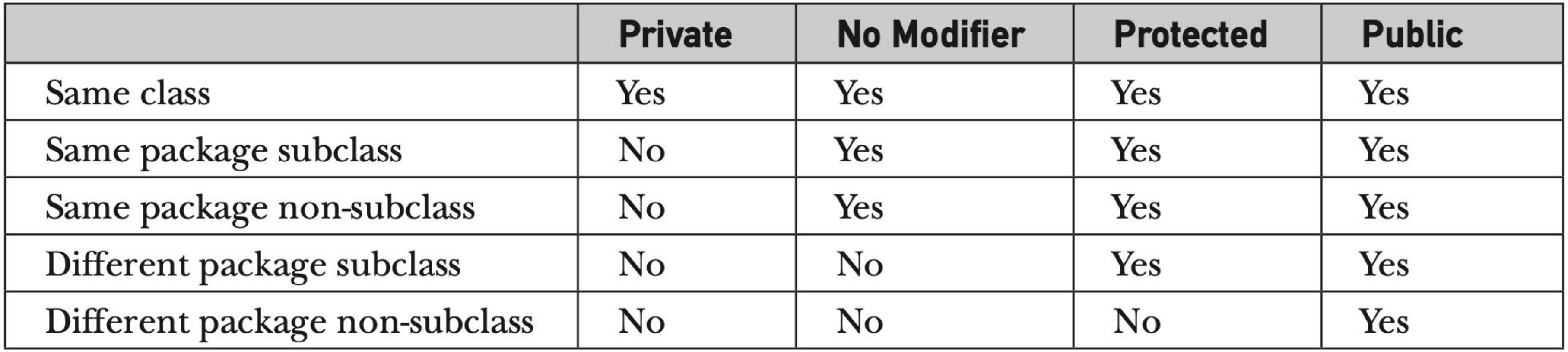
- ❖ Classes act as containers for data and code.

- ❖ The class is Java’s smallest unit of abstraction.

- ❖ Because of the interplay between classes and packages, Java 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

The three access modifiers, **private**, **public**, and **protected**, provide a variety of ways to produce the many levels of access required by these categories.



	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

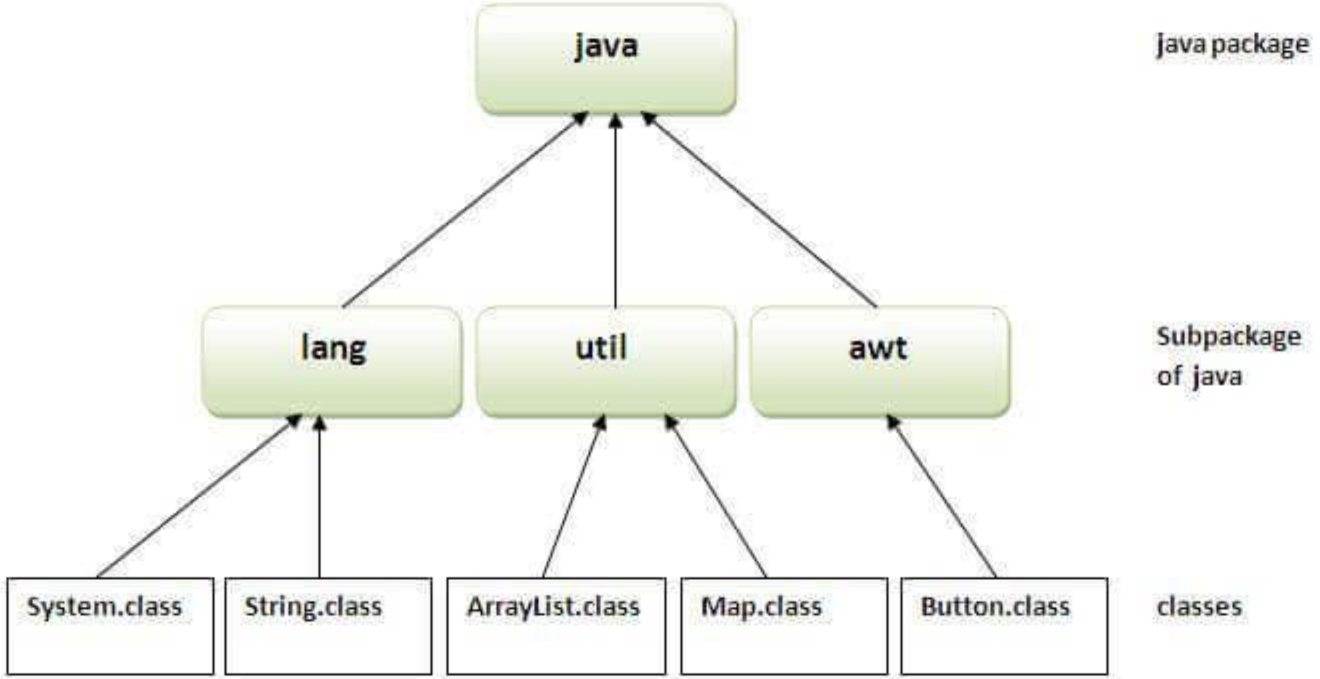
- ❖ A non-nested class has only two possible access levels: default and public.
- ❖ When a class is declared as **public**, it is accessible by any other code.
- ❖ If a class has default access, then it can only be accessed by other code within its same package.
- ❖ When a class is public, it must be the only public class declared in the file, and the file must have the same name as the class.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.


java
java package
lang
util
awt
Subpackage of java
System.class
String.class
ArrayList.class
Map.class
Button.class
classes