Packages and Access Protection

Packages

Packages are encapsulation of holding classes. Java provides packages for partitioning the classes into more manageable chunks.

Classes that are defined in a package are not accessible by classes outside of the package.

Class members that we define in general such as member variables, methods are generally declared in such a way that are visible to other members of the same package.

Why this way? So that the members of the same class have intimate knowledge of each other.

If we don't include a class in a package then the classes are put into **default package** by default.

As we have already seen that java uses file systems or directories to keep and manage the packages. The hierarchy of packages are maintained by separating with a . symbol.

We have seen general syntax like

java.something.something.something .... and so on.

The general form of multilevel package statement is:

package pkg1[.pkg2[.pkg3]]

and the general form of import statements occur following the package statements.

import pkg1 [.pkg2].classname | *

for example **import java.io.File**

and something we wall have seen quite a few times:

import java.util.Scanner

Access Protection

So packages are containers of classes and even smaller packages and in turn the classes are containers for data and code.

Class is the smallest unit of abstraction that is dealt in java.

Access modifiers are just keywords that control access to the properties and methods of the class.

We have seen **two access modifiers** so far, **private and public** and which we have seen them in action too.

There is also another **access modifier called protected**. When we don't apply any access modifier then the member of the class(method of property) has **default access**

The following table shows how the members of class can communicate with each other upon various access modifiers applied to them:

Members of	Private	Protected	Public	No Modifier
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	Yes	Yes	No
Different package non-subclass	No	No	Yes	No