COJ :: Packages & Access Modifiers

TalentSprint

Licensed To Skill

Version 1.0.4

Learning Objectives

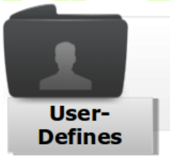
The content in this presentation is aimed to learn the following:

- Understanding Packages
- Types of Packages
- Usage of packages
- Working with predefined and user defined packages
- Use access modifier
- Explain the scope of various access modifiers

Packages

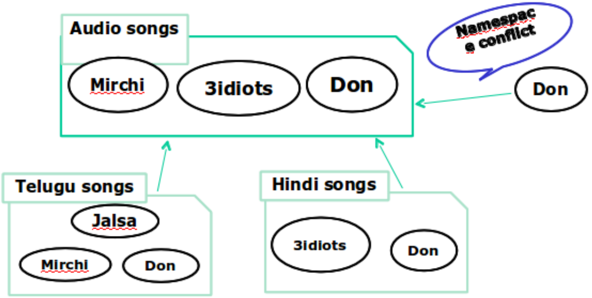
What is a Package?
A package is a grouping of related types (classes, interfaces, Exceptions and Annotations) providing access protection and Name Space Management.

Built-In packages are library packages. Example: java.lang, java.awt etc.



User-defined packages are user developed packages.

Why Packages



Packages

`Why Packages?`

- Using packages programmers can easily determine that these types are related.
- As packages follow naming conventions, a programmer knows that all graphical related methods will be present in a package named graphics.
- The names of your types won't conflict with the type names in other packages because the package creates a new namespace.

Packages

Creating Packages

Syntax to create a package:

package packagename;

Here package is a keyword used to create a package followed by packagename.

Packages

`Creating sub packages`

Syntax to create a sub-package:

package package1.package2;

Here package1 is a package, which contains a sub-package named package2.

Note

The package statement must be the first line in the source file.

Packages

Program to demonstrate packages:

```
package mypackage;
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Packages

How to compile the above program:

```
javac -d . HelloWorld.java
```

- "-d" stands for "directory" which explains the compiler the location where the class files should be created.
- .(dot) stands for the current directory.

Packages

`What happens when we use package statement?`

- When we use a package statement, the underlying operating system will create a directory with the same name of the package.
- As a programmer we should make sure that the **.class** files should be compiled to the same package directory.

Make sure that you set the CLASSPATH to the directory where the **.class** files are located. After setting the classpath, we can run our programs as follows:

java packagename.classname

Packages

`Using Packages?`

Import statement is used to get access to the classes present in packages.

`Importing a Single Package Member:`

Syntax

import packagename.Classname;

While importing the packages we need to mention the fully qualified package name as below.

Example

import java.lang.Integer;

Packages

`Importing an Entire Package:`

We can access multiple classes present in the same package using single import statement as follows:

Syntax:

import packagename.subpackage.$*$;

Asterisk(*) represents all the classes present in that package.

Example:

import java.lang.$*$;

Packages

While importing the packages we need to mention the fully qualified package name as above.

Note

It is a good programming practice to mention the fully qualified name of number of classes which are accessed from the same package.

Example

import java.lang.Integer;
import java.lang.Double;

Packages

Points to remember

The industry convention for creating a user-defined package is called as `Reverse Domain Naming Convention principle`.

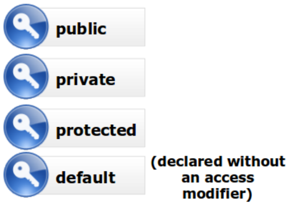
Syntax and Example

package domainname.companyname.
projectname.modulename;
package com.talentsprint.osp.inventory;

To access the sub-package classes one has to explicitly import it.
import javax.servlet.∗;
import javax.servlet.http.∗;

Access Modifier

Access specifiers specifies who can access them. There are four access specifiers used in java.



- public
- private
- protected
- default (declared without an access modifier)

Access Specifiers regulates the access to classes, constructors, methods and fields.

Access Modifier

`Access modifiers are restricted to two levels:`

- Class level access specifiers
- Member level access specifiers

Access Modifier

Access modifiers are restricted to two levels:

Class level access specifiers

For java classes only

Two access specifiers are allowed

public Can be accessed from anywhere

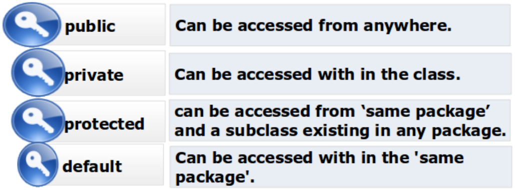
default Can only be accessed from ‘same package’.

Access Modifier

Access modifiers are restricted to two levels:

Member level access specifiers

- For java variables and java methods
- All four access specifiers are allowed

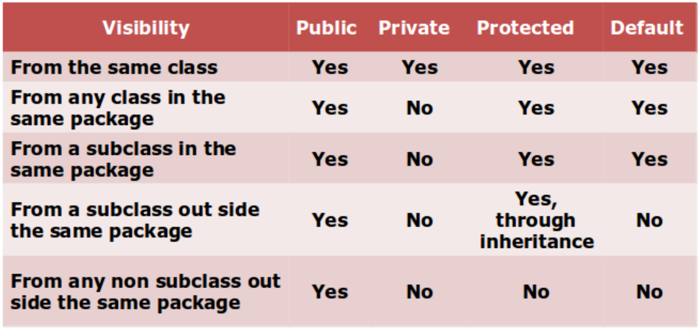


public	Can be accessed from anywhere.
private	Can be accessed with in the class.
protected	can be accessed from 'same package' and a subclass existing in any package.
default	Can be accessed with in the 'same package'.

Access Modifier

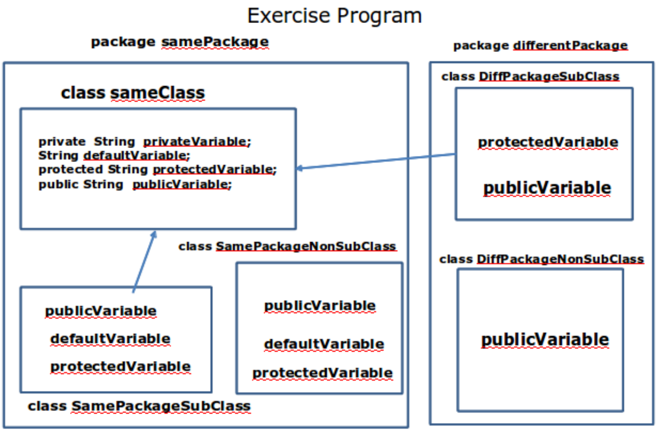
Tabular formulation of member level access:

Member level access specifiers



Visibility	Public	Private	Protected	Default
From the same class	Yes	Yes	Yes	Yes
From any class in the same package	Yes	No	Yes	Yes
From a subclass in the same package	Yes	No	Yes	Yes
From a subclass out side the same package	Yes	No	Yes, through inheritance	No
From any non subclass out side the same package	Yes	No	No	No

Access Modifier



Packages and Access Modifiers