Java Access Protection

Classes and packages are means of encapsulating and containing the name space and scope of the variables and methods.
Packages behaves as containers for classes and other subordinate packages.
Classes act as containers for data and code.

Class Members Visibility

The Java's smallest unit of abstraction is class. Because of the interplay between the classes and packages, Java addresses the following four categories of visibility for class members :

- Subclasses in same package
- Non-subclasses in same package
- Subclasses in different packages
- Classes that are neither in same package nor in subclasses

The three access modifiers are :

- public
- private
- protected

provides a variety of ways to produce many levels of access required by these categories. The upcoming table sums up the interaction
While the access control mechanism of Java may seem complicated, we can simplify it as follows.
Anything declared as **public** can be accessed from anywhere.
Anything declared as **private** can't be seen outside of its class.

Class Member Access

When a member doesn't have an explicit access specification, then it is visible to the subclasses as well as to the other classes in the same package. This is the default access. And If you want to allow an element to be seen outside your current package, but only to the classes that subclass your class directly, then declare that element protected.

	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

This table applies only to the members of classes. A non-nested class has only two possible access levels i.e., **default** and **public**.
When a class is declared as **public**, then it is accessible by any other code. If a class has default access, then it can only be accessed by the other code within its same package. When a class is public, it must be only the public class declared in the file that must have the same name as the class.

Java Access Protection Example

Here the upcoming example allows all the combinations of access control modifiers. This example has two packages and five classes.
Always remember that the classes for the two different packages need to be stored in directories after their respective packages (in this case **pkg1** and **pkg2**).
The source for the first package defines the three classes i.e., **Protection, Derived**, and **SamePackage**. The first class defines the four variables of type **int** in each of the legal protection modes. The variable **n** declared with the default protection, the variables **n_priv, n_prot,** and **n_publ** is **private, protected,** and **public** respectively.
Each subsequent class in the following example will try to access the variables in an instance of this class. The lines that will not compile due to the access restrictions are commented out. Before each of these lines is a comment that listing the places from which this level of protection would allow access.
The second class named **Derived**, is a subclass of **Protection** in the same package, **pkg1**. This grants **Derived** access to every variable in the class **Protection** except for **n_priv**, the private one. The third class named **SamePackage**, is not a subclass of the class **Protection**, but is in the same package and also has access to all but not **n_priv**.

This is **Protection.java** file:

```
package pkg1;

public class Protection
{
	int n = 1;
	private int n_priv = 2;
	protected int n_prot = 3;
	public int n_publ = 4;

	public Protection()
	{
		System.out.println("base constructor");
		System.out.println("n = " + n);
		System.out.println("n_priv = " + n_priv);
		System.out.println("n_prot = " + n_prot);
		System.out.println("n_publ = " + n_publ);
	}
}
```

This is **Derived.java** file:

```
package pkg1;

class Derived extends Protection
{
	Derived()
```

```
        {
                System.out.println("derived constructor");
                System.out.println("n = " + n);

                /* class only
                *  System.out.println("n_priv = "4 + n_priv); */

                System.out.println("n_prot = " + n_prot);
                System.out.println("n_publ = " +n_publ);
        }
}
```

This is **SamePackage.java** file:

```
package pkg1;

class SamePackage
{
        SamePackage()
        {
                Protection pro = new Protection();
                System.out.println("same package constructor");
                System.out.println("n = " + pro.n);

                /* class only
                *  System.out.println("n_priv = " + pro.n_priv); */

                System.out.println("n_prot = " + pro.n_prot);
                System.out.println("n_publ = " + pro.n_publ);
        }
}
```

Following is the source code for the other package named **pkg2**. The two classes defined in the package **pkg2** cover the outer two conditions that are affected by the access control. The first class named **Protection2**, is a subclass of **pkg1.Protection**. This grants access to all of **pkg1**. Variables of the class **Protection** except for **n_priv** (because it is private) and **n**, the variable declared with the default protection.
Always remember that the default only allows access from within the class or the package, not extra-package subclasses. Finally, the class **OtherPackage** has access to **n_publ** only which was declared as **public**.

This is **Protection2.java** file:

```
package pkg2;

class Protection2 extends pkg1.Protection
{
```

```
        Protection2()
        {
                System.out.println("derived other package constructor");

                /* class or package only
                *  System.out.println("n = " + n); */

                /* class only
                *  System.out.println("n_priv = " + n_priv); */

                System.out.println("n_prot = " + n_prot);
                System.out.println("n_publ = " + n_publ);
        }
}
```

This is **OtherPackage.java** file:

```
package pkg2;

class OtherPackage
{
        OtherPackage()
        {
                pkg1.Protection pro = new pkg1.Protection();

                System.out.println("other package constructor");

                /* class or package only
                *  System.out.println("n = " + pro.n); */

                /* class only
                *  System.out.println("n_priv = " + pro.n_priv); */

                /* class, subclass or package only
                *  System.out.println("n_prot = " + pro.n_prot); */

                System.out.println("n_publ = " + pro.n_publ);
        }
}
```

If you want to try these two packages, here are two test files you can use. The one for package **pkg1** is shown here:

```
/* demo package pkg1 */

package pkg1;

/* instantiate the various classes in pkg1 */
```

```
public class Demo
{
	public static void main(String args[])
	{
		Protection obj1 = new Protection();
		Derived obj2 = new Derived();
		SamePackage obj3 = new SamePackage();
	}
}
```

The test file for **pkg2** is shown below :

```
/* demo package pkg2 */

package pkg2;

/* instantiate the various classes in pkg2 */
public class Demo
{
	public static void main(String args[])
	{
		Protection2 obj1 = new Protection2();
		OtherPackage obj2 = new OtherPackage();
	}
}
```