**Package:**

The standard of defining package is domain.company.package e.g. com.javatpoint.bean or org.sssit.dao.

**Access Specifiers**

**Why?**

To set access levels for classes, variables, methods, and constructors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifiers | Within Class | Within Package | Outside package ( By subclass only) | Outside package |
| Private | Yes | No | No | No |
| Default | Yes | Yes | No | No |
| Protected( for members of class alone. Cannot apply for class) | Yes | Yes | Yes | No |
| Public | Yes | Yes | Yes | Yes |

**Private: (object encapsulation)**

Class and interfaces cannot be private.

Methods, variables, and constructors that are declared private can only be accessed within the declared class itself.

**Exception:** Variables that are declared private can be accessed outside the class, **if public getter methods are present in the class.**

public class Logger {

private String format;

public String getFormat() {

return this.format;

}

public void setFormat(String format) {

this.format = format;

}

}

Here, the *format* variable of the Logger class is private, so there's no way for other classes to retrieve or set its value directly.

So, to make this variable available to the outside world, we defined two public methods: *getFormat()*, which returns the value of format, and *setFormat(String)*, which sets its value.

Protected :

**Protected access** gives the subclass a chance to use the helper method or variable, while preventing a nonrelated class from trying to use it.

Public :

The main() method of an application has to be public. Otherwise, it could not be called by a Java interpreter (such as java) to run the class.

### **Access Control and Inheritance**

The following rules for inherited methods are enforced −

* Methods declared public in a superclass also must be public in all subclasses.
* Methods declared protected in a superclass must either be protected or public in subclasses; they cannot be private.
* Methods declared private are not inherited at all, so there is no rule for them

Interface: The fields in an interface are implicitly public static final and the methods in an interface are by default public.

methods and fields in an interface cannot be declared protected.