JAVA ACCESS MODIFIERS

Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are:

- Visible to the package. the default. No modifiers are needed.

- Visible to the class only (private).

- Visible to the world (public).

- Visible to the package and all subclasses (protected).

Default Access Modifier - No keyword:

Default access modifier means we do not explicitly declare an access modifier for a class, field, method, etc.

A variable or method declared without any access control modifier is available to any other class in the same package. The fields in an interface are implicitly public static final and the methods in an interface are by default public.

Example:

Variables and methods can be declared without any modifiers, as in the following examples:

```
String version = "1.5.1";

boolean processOrder() {
    return true;
}
```

Private Access Modifier - private:

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

Private access modifier is the most restrictive access level. Class and interfaces cannot be private.

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

Using the private modifier is the main way that an object encapsulates itself and hide data from the outside world.

Example:

The following class uses private access control:

```
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.

Public Access Modifier - public:

A class, method, constructor, interface etc declared public can be accessed from any other class. Therefore fields, methods, blocks declared inside a public class can be accessed from any class belonging to the Java Universe.

However if the public class we are trying to access is in a different package, then the public class still need to be imported.

Because of class inheritance, all public methods and variables of a class are inherited by its subclasses.

Example:

The following function uses public access control:

```
public static void main(String[] arguments) {
    // ...
}
```

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.

Protected Access Modifier - protected:

Variables, methods and constructors which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class.

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected, however methods and fields in a interface cannot be declared 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.

Example:

The following parent class uses protected access control, to allow its child class override *openSpeaker()* method:

```
class AudioPlayer {
    protected boolean openSpeaker(Speaker sp) {
        // implementation details
    }
}

class StreamingAudioPlayer {
    boolean openSpeaker(Speaker sp) {
        // implementation details
    }
}
```

Here, if we define *openSpeaker()* method as private, then it would not be accessible from any other class other than *AudioPlayer*. If we define it as public, then it would become accessible to all the outside world. But our intension is to expose this method to its subclass only, thats why we used *protected* modifier.

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 without access control (no modifier was used) can be declared more private in subclasses.

- Methods declared private are not inherited at all, so there is no rule for them.