*Access specifiers*

Whenever we are writing our own classes compulsory we have to provide some information about our class to the jvm. Like

1. Whether this class can be accessible from anywhere or not.

2. Whether child class creation is possible or not.

3. Whether object creation is possible or not etc.

We can specify this information by using the corresponding modifiers. The only applicable modifiers for Top Level classes are:

1. Public

2. Default

3. **Protected**

4. **Private**

*1.Public*

* If a class declared as public then we can access that class from anywhere. Within the package or outside the package.
* It applies to outer class, inner class, global variables, static variables, methods and constructors.
* Local variables can’t be public because they have a limited scope

*Example*

We declared the class as public in velocity. package1 let access the class in outside the package i.e. velocity. package2

**package** com.velocity.package1;

**public** **class** Test {

**public** **void** methodOne(){

System.***out***.println("test class method one is executed");

}

}

**package** com.velocity.Pacakge2;

**import** com.velocity.package1.Test;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

Test test= **new** Test();

test.methodOne();

}

}

*Output*

test class method one is executed

*2. Default*

* If a class declared as the default then we can access that class only within the current package hence default access is also known as "package level access".
* If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package. It cannot be accessed from outside the package.
* It applies to global variables, static variables, methods, constructors and inner class.

*Example*

**package** com. velocity. package1;

**class** Test {

**public** **void** methodOne(){

System.***out***.println(" Default test class method one is executed");

}

}

**package** com. velocity. Pacakge2;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

Test test= **new** Test () ;//CE Test cannot be resolved to a type

test.methodOne();

}

}

*3.* ***Protected***

* If a member declared as the protected then we can access that member within the current package anywhere but outside package only in child classes.
* But from outside package we can access protected members only in child classes and should be by child reference only that is we can't use parent reference to call protected members from outside package.
* If you do not make the child class, it cannot be accessed from outside the package.
* It applies to global variables, static variables, methods, constructors, inner class and outer class

I make the method with protected access specifier in package1 (**package** com. velocity. package1) when I try to access outside the package it’s not accessible it gives the compile error.

*Example*

**package** com. velocity. package1;

**public** **class** Test {

**protected** **void** methodOne(){

System.***out***.println(" Default test class method one is executed");

}

}

**package** com. velocity. Pacakge2;

**import** com. velocity. package1.Test;

**public** **class** Test1 {

**public** **static** **void** main (String [] args) {

Test test= **new** Test ();

test.methodOne();//The method method One() from the type Test is not visible

}

}

outside package we can access protected members only in child classes for the we have to make the child class in pacakge2

*Example*

**package** com. velocity. package1;

**public** **class** Test {

**protected** **void** methodOne(){

System.***out***.println(" protected method one is executed ");

}

}

**package** com.velocity.Pacakge2;

**import** com.velocity.package1.Test;

**public** **class** Test1 **extends** Test {

**public** **static** **void** main(String[] args) {

Test1 test= **new** Test1();

test.methodOne();

}

}

Output

protected method done is executed

*4. Private members*

* If a member declared as the private then we can access that member only with in the current class.
* Private methods are not visible in child classes
* It cannot be accessed from outside the class
* It applies to global variables, static variables, methods, constructors and inner class
* Local variable can’t be private.

private then we can access that member only with in the current class.

*Example*

**package** com.velocity.package1;

**public** **class** Test {

**private** **void** methodOne(){

System.***out***.println(" private methodone is executed");

}

**public** **static** **void** main(String[] args) {

Test test=**new** Test();

test.methodOne();

}

}

*Output* private method one is executed

It cannot be accessed from outside the class

*Example*

**package** com.velocity.package1;

**public** **class** Test {

**private** **void** methodOne() {

System.***out***.println(" private methodone is executed");

}

}

**package** com.velocity.Pacakge2;

**import** com.velocity.package1.Test;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

Test test= **new** Test();

test.methodOne();//The method methodOne() from the type Test is not visible

}

}