**Access Specifiers-**

* As the name suggests access specifier’s in Java helps to restrict the scope.
* Access specifier’s plays very important role while performing the operation on variable, methods, classes, etc.
* In other words, it is used to restrict the access of variable, methods, classes, etc.

There are four types of access specifiers as

* Private
* Default
* Protected
* Public

**Private-**

* The access level of a private specifiers is only within the class.
* It can not access from outside class or outside package as scope is very limited.\
* It apply to global variable, method, constructor and inner class only.
* Class cannot be private.
* Local variables cannot private.

**public class** Test{

**private** **int** data=10;

**private** **void** msg(){

System.out.println("Velocity");}

}

**public** **class** Example{

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

   Test test=**new** Test ();

   System.out.println(test.data);//Compile Time Error

   test.msg();//Compile Time Error

   }

}

**Default-**

* The access level of a default specifiers is only within the package.
* It cannot be accessed from outside the package.
* When the access specifiers is not specified then it will be treated as default members.
* It apply to global variable, local variable, constructors, method, inner class and outer class.
* No need to use keyword default like private.

Example: Within package

**public class** Test{

**int** data=10;   //Initializing without default keyword //correct

**default** **void** msg(){ //defining with default keyword //correct but we avoid

System.out.println("Velocity");}

}

**public** **class** Example{

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

   Test test=**new** Test ();

   System.out.println(test.data);//it will work as its in same package

   test.msg();//it will work as its in same package

   }

}

**Protected-**

* The access level of a protected specifiers is within the package and outside the package if inheritance is happened while calling.
* It apply to constructor, global variables, inner class and methods.
* It cannot apply to local variables and outer class.

Example: in different package

**package** com.test;

**public** **class** Test{

**protected** **void** msg(){System.out.println("Velocity");}

}

**package** com.example;

**import** com.test;

**class** Example **extends** Test{

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

   Example  example  = **new** Example  ();

   example .msg();

  }

}

**Public-**

* It can access anywhere in the class or outside the class or same package or different package.
* It apply to class, method, constructor, global variable, static variable, inner class, outer class.
* Local variables cannot public because they have limited scope within the method only. If we make it public then getting error only. “Illegal modifier for parameter”

**Note-**

1. We can apply default access Specifiers or final on local variable.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | private | default | protected | public |
| Within class | Yes | Yes | Yes | Yes |
| Within package | No | Yes | Yes | Yes |
| Outside package by subclass only | No | No | Yes | Yes |
| Outside package non-subclass | No | No | No | Yes |

**Why we use access specifiers?**

If we have a business requirement where we need to perform the employee CRUD operations and all the methods need to be called from getEmployeeData() only.

Scenario-1

**package** com.wipro.velocity;

**public** **class** Employee {

**public** **void** addOperation() {

System.***out***.println("Add operation");

}

**public** **void** editOperation() {

System.***out***.println("Edit operation");

}

**public** **void** getOperation() {

System.***out***.println("Get operation");

}

**public** **void** deleteOperation() {

System.***out***.println("Delete operation");

}

**public** **void** getEmployeeData() {

addOperation();

editOperation();

getOperation();

deleteOperation();

}

}

Here we are directly call any method from outside class because scope is public. Hence requirement is not fulfilled here.

Scenario-2

class Employee{

**package** com.wipro.velocity;

**public** **class** Employee {

**private** **void** addOperation() {

System.***out***.println("Add operation");

}

**private** **void** editOperation() {

System.***out***.println("Edit operation");

}

**private** **void** getOperation() {

System.***out***.println("Get operation");

}

**private** **void** deleteOperation() {

System.***out***.println("Delete operation");

}

**public** **void** getEmployeeData() {

addOperation();

editOperation();

getOperation();

deleteOperation();

}

}

Here we cannot directly call any method except getEmployeeData() because scope is private. So it cannot be directly accessible from outside. We need to access it from by calling getEmployeeData ().

In this way, we use the access specifiers.