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***).

Java 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.

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.
- It can be applied on the data member, method and constructor. It can't be applied on the class.

Given the example code below, suppose the Java classes are actually in different files:

Example-2:

```
package randomPackage;
public class A{
        float f1;
        protected int i1;
}

                                        // note that class B belongs to the same package as class A
package randomPackage;
public class B{
        public void someMethod()        {
                // create an instance of class A
                A anInstance = new A();

                anInstance.f1 = 19;
                anInstance.i1 = 12;  // Visible to the package
        }
}

public class C extends A{
        public void someOtherMethod(){
                i1 = 89; // Visible to the all subclasses
        }
}
```

Example-2:

```
//A.java
package pack;
  public class A{
  protected void msg(){System.out.println("Hello");}
}

//B.java
package mypack;
import pack.*;

class B extends A{
  public static void main(String args[]){
    B obj = new B();
    obj.msg();
  }
}
```

Example 1: Program that illustrates protected variable use in java using shape class

```
public class Shape{
  protected int sides;

  public Shape(){
    sides = 3;
  }
     public int getSides(){
    return sides;
  }
  public printSides(){
     System.out.println("This object has " + sides + " sides." );
  }
}

public class Square extends Shape{
  public Square(int nSides){
    sides = nSides;  // don't need to call super class constructor due to protected type of variable.
  }
}
```

```
class ProtectedVariableDemo{
  public static void main(String args[]){
     Square sObj = new Square(10);

     sObj.printSides();
  }
}
```