https://www.geeksforgeeks.org/access-modifiers-java/

1.Default Access Modifier-

When no access modifier is specified for a class, method, or data member – It is said to be having the default access modifier by default. The data members, classes, or methods that are not declared using any access modifiers i.e. having default access modifiers are accessible only within the same package.

In this example, we will create two packages and the classes in the packages will be having the default access modifiers and we will try to access a class from one package from a class of the second package.

```
// Java program to illustrate default modifier
package p1;

// Class Geek is having Default access modifier
class Geek
{
    void display()
    {
        System.out.println("Hello World!");
    }
}

// Java program to illustrate error while
// using class from different package with
// default modifier
package p2;
import p1.*;

// This class is having default access modifier
class GeekNew
{
    public static void main(String args[])
    {
        // Accessing class Geek from package p1
        Geek obj = new Geek();

        obj.display();
    }
}
```

Output:

```
Compile time erro
```

2. Private Access Modifier

The private access modifier is specified using the keyword **private**. The methods or data members declared as private are accessible only **within the class** in which they are declared.

- Any other **class of** the **same package will not be able to access** these members.
- Top-level classes or interfaces can not be declared as private because
    - private means “only visible within the enclosing class”.
    - protected means “only visible within the enclosing class and any subclasses”

Hence these modifiers in terms of application to classes, apply only to nested classes and not on top-level classes

In this example, we will create two classes A and B within the same package p1. We will declare a method in class A as private and try to access this method from class B and see the result.

```
package p1;

// Class A
class A {
    private void display()
    {
        System.out.println("GeeksforGeeks");
    }
}
// Class B
class B {
    public static void main(String args[])
    {  A obj = new A();
        // Trying to access private method
        // of another class
        obj.display();
    }
}
```

Output:

```
error: display() has private access in A
        obj.display();
```

3 Protected Access Modifier

The protected access modifier is specified using the keyword **protected**.

The methods or data members declared as protected are **accessible within the same package or subclasses in different packages.**

In this example, we will create two packages p1 and p2. Class A in p1 is made public, to access it in p2. The method display in class A is protected and class B is inherited from class A and this protected method is then accessed by creating an object of class B.

```
// Protected Modifier
package p1;
// Class A
public class A {
    protected void display()
    {
        System.out.println("GeeksforGeeks");
    }
}
package p2;
import p1.*;
// Class B is subclass of A
class B extends A {
    public static void main(String args[])
    {
        B obj = new B();
        obj.display();
    }
}
```

Output:

```
GeeksforGeeks
```

4.Public Access modifier

The public access modifier is specified using the keyword **public**.

- The public access modifier has the **widest scope** among all other access modifiers.
- Classes, methods, or data members that are declared as public are **accessible from everywhere** in the program. There is no restriction on the scope of public data members.

```
// Java program to illustrate
// public modifier
package p1;
public class A
{
public void display()
    {
        System.out.println("GeeksforGeeks");
    }
}


package p2;
import p1.*;
class B {
    public static void main(String args[])
    {
        A obj = new A();
        obj.display();
    }
}
```

Output:

```
GeeksforGeeks
```

	Default	Private	Protected	Public
Same Class	Yes	Yes	Yes	Yes
Same Package Subclass	Yes	No	Yes	Yes
Same Package Non-Subclass	Yes	No	Yes	Yes
Different Package Subclass	No	No	Yes	Yes
Different Package Non-Subclass	No	No	No	Yes