Access Modifiers in Java

Access modifiers in Java specify the accessibility or scope of a **field**, **method**, **constructor**, or **class**. In other words, with access modifiers, we can **allow** fields, methods, constructors, and classes to be seen and used in other parts of the code. Alternatively, we can **prevent** them from being seen and used–all with one easy modifier.

This is great for protecting sensitive information and works well with the principles of OOP. We can effectively achieve abstraction through access modifiers, keeping the logic of one object hidden from others unless they need access. Access modifiers are also key in encapsulation because they help us package up our code and separate our concerns.

There are **four** types of access modifiers in Java:

- **Default**
- **Private**
- **Protected**
- **Public**

Let's look at each of these one at a time, with some examples.

#Private

If a field, method, constructor is declared with the **private** access modifier, then such a field (class, method, etc.) will **only** be accessible within its declared class.

For example:

```
class Data {
// private field
private String address;
}

public class Main {
public static void main(String[] main) {
// Create an object of the Data class
Data data = new Data();
// Trying to access a private field from another class
data.address = "Toronto"; // This line would cause an error
because the String address is private.
}
}
```

In the example above, we declared the `address` field as private in the `Data` class. Trying to read or write to this private field from another class causes an error. It's not allowed because the field is **private**.

#Default

As in so many other places, `default`, as an access modifier, means "you didn't declare anything, so this is what we're going with." **If we do not specify an access modifier, Java will automatically treat it as `default`.** With `default` access, we can access fields, methods, constructors, and classes that are declared with the ***only* within the same package**. We can ***not*** access them from outside the package. This provides more accessibility than the *private* modifier, but it's still more restrictive than `protected` and `public`.

Let's look at an example:

```
// Inside of package p1

// ClassA has a default access modifier since none was
provided.
class ClassA {
// the print method also has a default access modifier.
void print(){
System.out.println("Hello from ClassA");
}
}
```

```
// Inside of package p2

// Importing ClassA from a different package to be used in
this class
import p1.ClassA;

class ClassB {
public static void main(String args[]){
ClassA aObj = new ClassA();
aObj.print(); // We'd get an error here.
}
}
```

Again, we will get an error when we try to compile this code because `ClassB` isn't within the same package as `ClassA` and `ClassA` has a `default` access modifier. For more on the default access modifier, check out this blog here.

#Protected

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.** Let's tweak our previous example to test this:

```
// Inside of package p1

// ClassA has a default access modifier since none was
provided.
class ClassA {
// the print method also has a default access modifier.
protected void print(){
System.out.println("Hello from ClassA");
}
}
// Inside of package p2

// Importing ClassA from a different package, so that ClassB
can extend ClassA
import p1.ClassA;

// Because ClassB extends ClassA, ClassB is a subclass of
ClassA:
class ClassB extends ClassA {
public static void main(String args[]){
ClassB bObj = new ClassB();
bObj.print(); // ClassB inherited the protected print method
of ClassA, so this won't throw an error.
}
}
```

In essence, you use protected when you want to be accessible within the same package or subclasses in different package.

#Public

Everything goes! We can access fields, methods, constructors, and classes declared with the *public* modifier from **anywhere** within our codebase. We can access public modifiers from within the class as well as from outside the class and also within the package and outside the package. This modifier has the widest scope.

Example:
```
// public class
public class Vehicle {
// public variable
public int wheelCount;

// public method
public void display() {
System.out.println("I am a vehicle.");
System.out.println("I have " + wheelCount + " wheels.");
}
}

// Main.java
public class Main {
public static void main( String[] args ) {
// accessing the public class
Vehicle vehicle = new Vehicle();

// accessing the public variable
vehicle.wheelCount = 4;
// accessing the public method
vehicle.display();
}
}
```

None of the above will throw errors because we declared everything `public`.

#Summary

Based on our discussion, this table summarizes the effect of applying an access modifier to a field, method, constructor or class:

Access Modifier	within class	within package	outside package by subclass only	outside package

Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

InfoWarningTip
You may want to screenshot the table above or bookmark this page for future reference. Accessor methods are pretty simple, and we'll mostly be using `private` or `public` from here on out, but you never know when a little table like this one will come in handy.

#Knowledge Check

Can you create a sub-class for the following class?

```
class A {
private A() {
// First Constructor
}

private A(int i) {
// Second Constructor
}
}
```

Yes

No

Not enough information is given.

Can the field `i` of Class A be inherited by Class B in the code below?(Assume Class B is in a different package)

```
class A {
    protected int i;
}

class B extends A {
}
```

Yes

No

Not enough information is given.

In your own words, explain the answers to the two questions above. This question is unscored, but we will be checking in if we need more information about how you're doing in the course. Please be thorough:

Is the code below written correctly?

```
package pack1;

class A {

}

package pack2;
import pack1.A;
class B extends A {

}
```

Yes

No

Not enough information is given.

Can you create a sub-class for the following class?

```
class A {

    public A() {
        //First Constructor
    }
}
```

Yes

No

Not enough information is given.

Options

Lowest Visibility

PRIVATE

Second Lowest Visibility

DEFAULT

Second Highest Visibility

PROTECTED

Highest Visibility

PUBLIC

#Extra Resources

When you have time, we recommend watching this video for more code walkthrough examples about access modifiers:

https://youtu.be/T632kAJ_9VA