Java Access Modifiers

Types of Access Modifier

Modifier	Description
Default	declarations are visible only within the package (package private)
Private	declarations are visible within the class only
Protected	declarations are visible within the package or all subclasses
Public	declarations are visible everywhere

Default Access Modifier

```java
package defaultPackage;
class Logger {
    void message(){
        System.out.println("This is a message");
    }
}
```

Here, the `Logger` class has the default access modifier. And the class is visible to all the classes that belong to the `defaultPackage` package. However, if we try to use the `Logger` class in another class outside of defaultPackage, we will get a compilation error.

Private Access Modifier

When variables and methods are declared `private`, they cannot be accessed outside of the class. For example,

```
class Data {
    // private variable
    private String name;
}

public class Main {
    public static void main(String[] main){

        // create an object of Data
        Data d = new Data();

        // access private variable and field from another class
        d.name = "Programiz";
    }
}
```

In the above example, we have declared a private variable named `name`. When we run the program, we will get the following error:

```
Main.java:18: error: name has private access in Data
        d.name = "Programiz";
         ^
```

The error is generated because we are trying to access the private variable of the `Data` class from the `Main` class.

You might be wondering what if we need to access those private variables. In this case, we can use the getters and setters method. For example,

```java
class Data {
    private String name;

    // getter method
    public String getName() {
        return this.name;
    }
    // setter method
    public void setName(String name) {
        this.name= name;
    }
}
public class Main {
    public static void main(String[] main){
        Data d = new Data();

        // access the private variable using the getter and setter
        d.setName("Programiz");
        System.out.println(d.getName());
    }
}
```

Output:

```
The name is Programiz
```

Protected Access Modifier

When methods and data members are declared `protected`, we can access them within the same package as well as from subclasses. For example,

```
         class Animal {
    // protected method
    protected void display() {
        System.out.println("I am an animal");
    }
}

class Dog extends Animal {
    public static void main(String[] args) {

        // create an object of Dog class
        Dog dog = new Dog();
         // access protected method
        dog.display();
    }
}
```

Output:

```
I am an animal
```

Public Access Modifier

When methods, variables, classes, and so on are declared `public`, then we can access them from anywhere. The public access modifier has no scope restriction. For example,

```
// Animal.java file
// public class
public class Animal {
    // public variable
    public int legCount;

    // public method
    public void display() {
        System.out.println("I am an animal.");
        System.out.println("I have " + legCount + " legs.");
    }
}

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

        // accessing the public variable
        animal.legCount = 4;
        // accessing the public method
        animal.display();
    }
}
```

Output:

```
I am an animal.
I have 4 legs.
```