ACCESS MODIFIERS

access modifiers are used to set the accessibility (visibility) of classes, interfaces, variables, methods, constructors, data members, and the setter methods.

Need of Access Modifiers

* Access modifiers are mainly used for encapsulation.
* It can help us to control what part of a program can access the members of a class.
* So that misuse of data can be prevented.

**Note**: You cannot set the access modifier of getters methods.

|  |  |
| --- | --- |
| **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**

If we do not explicitly specify any access modifier for classes, methods, variables, etc, then by default the default access modifier is considered.

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.

**Note**: We cannot declare classes and interfaces private in Java. However, the nested classes can be declared private.

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";

}

}

Output:

It will throw an error

//Hence we need to resolve this issue with the getter() and setter() method

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());

}

}

### **Protected Access Modifier**

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

**Note**: We cannot declare classes or interfaces protected in Java.

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.

// 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