**Java Access Modifiers**

**What are Access Modifiers?**

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

class Animal {

public void method1() {...}

private void method2() {...}

}

In the above example, we have declared 2 methods: method1() and method2(). Here,

method1 is public - This means it can be accessed by other classes.

method2 is private - This means it can not be accessed by other classes.

Note the keyword public and private. These are access modifiers in Java. They are also known as visibility modifiers.

**Types of Access Modifier**

Before you learn about types of access modifiers, make sure you know about Java Packages.

There are four access modifiers keywords in Java and they are:

...

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. For example,

package defaultPackage;

class Logger {

void message(){

System.out.println("This is a message");

}

}

Run Code

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";

}

}

Run Code

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,

class Data {

private String name;

// setter method

public void setName(String name) {

this.name= name;

}

// getter method

public String getName() {

return this.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("Dr.Sm. Badhusha");

System.out.println(d.getName());

}

}

Run Code

Output:

The name is Dr.Sm. Badhusha

In the above example, we have a private variable named name. In order to access the variable from the outer class, we have used methods: getName() and setName(). These methods are called getter and setter in Java.

Here, we have used the setter method (setName()) to assign value to the variable and the getter method (getName()) to access the variable.

We have used this keyword inside the setName() to refer to the variable of the class. To learn more on this keyword, visit Java this Keyword.

**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();

}

}

Run Code

Output:

I am an animal

In the above example, we have a protected method named display() inside the Animal class. The Animal class is inherited by the Dog class. To learn more about inheritance, visit Java Inheritance.

We then created an object dog of the Dog class. Using the object we tried to access the protected method of the parent class.

Since protected methods can be accessed from the child classes, we are able to access the method of Animal class from the Dog class.

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

**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();

}

}

Run Code

Output:

I am an animal.

I have 4 legs.

Here,

The public class Animal is accessed from the Main class.

The public variable legCount is accessed from the Main class.

The public method display() is accessed from the Main class.