**Access Modifiers**

Access modifiers in Java are keywords used to define the visibility or accessibility of classes, methods, variables, and constructors. They control which other classes can access and modify these components. Java provides **four primary access modifiers**:

1. **public**
2. **protected**
3. **private**
4. **default** (no modifier specified)

These access modifiers control access based on the relationship between classes and the location of those classes in different packages.

**1. public Access Modifier**

* **Visibility**: A public member (class, method, variable, etc.) can be accessed from anywhere in the program, regardless of the package or class.
* **Use Case**: Use public when you want to provide access to the member from other classes and packages.

Example:

public class MyClass {

public int publicVar = 100;

public void publicMethod() {

System.out.println("This is a public method.");

}

}

class Test {

public static void main(String[] args) {

MyClass obj = new MyClass();

System.out.println(obj.publicVar); // Accessible

obj.publicMethod(); // Accessible

}

}

**Explanation**:

* The publicVar and publicMethod() in the MyClass class are both declared as public, so they can be accessed from any other class, even outside the package.

**2. protected Access Modifier**

* **Visibility**: A protected member is accessible within its own package and by subclasses (including those in different packages).
* **Use Case**: Use protected when you want a member to be accessible by subclasses but not by other unrelated classes.

Example:

class Animal {

protected String name;

protected void makeSound() {

System.out.println("Animal makes a sound");

}

}

class Dog extends Animal {

void display() {

System.out.println("Dog name is: " + name); // Accessible due to protected modifier

makeSound(); // Accessible due to protected modifier

}

}

public class Test {

public static void main(String[] args) {

Dog dog = new Dog();

dog.display();

}

}

**Explanation**:

* The name and makeSound() are protected in the Animal class, so they are accessible within the same package (if used in another class in the same package) and in the Dog subclass.

**3. private Access Modifier**

* **Visibility**: A private member is only accessible within the class in which it is declared. It is not accessible from outside that class, including from subclasses.
* **Use Case**: Use private to hide the internal implementation details of a class from other classes, enforcing encapsulation.

Example:

class MyClass {

private int privateVar = 50;

private void privateMethod() {

System.out.println("This is a private method.");

}

void accessPrivate() {

System.out.println(privateVar); // Accessible within the same class

privateMethod(); // Accessible within the same class

}

}

public class Test {

public static void main(String[] args) {

MyClass obj = new MyClass();

// System.out.println(obj.privateVar); // Error: privateVar has private access

// obj.privateMethod(); // Error: privateMethod() has private access

obj.accessPrivate(); // Accessing private members through a public method

}

}

**Explanation**:

* The privateVar and privateMethod() in the MyClass class are private, meaning they cannot be accessed outside of the MyClass class. However, they can be accessed within the class itself.

**4. Default (Package-Private) Access Modifier**

* **Visibility**: When no access modifier is specified, the default access level is package-private, meaning the member is accessible only within classes in the same package.
* **Use Case**: Use the default access modifier when you want to limit access to within the same package, but not outside of it.

Example:

class MyClass {

int defaultVar = 200; // Default access modifier (package-private)

void defaultMethod() {

System.out.println("This is a default method.");

}

}

class AnotherClass {

void test() {

MyClass obj = new MyClass();

System.out.println(obj.defaultVar); // Accessible within the same package

obj.defaultMethod(); // Accessible within the same package

}

}

public class Test {

public static void main(String[] args) {

MyClass obj = new MyClass();

System.out.println(obj.defaultVar); // Accessible within the same package

obj.defaultMethod(); // Accessible within the same package

}

}

**Explanation**:

* Since defaultVar and defaultMethod() are not explicitly declared with an access modifier, they have package-private access. This means they can be accessed from other classes within the same package, but not from outside the package.

**Comparison of Access Modifiers**

| **Access Modifier** | **Class** | **Package** | **Subclass (same package)** | **Subclass (different package)** | **World** |
| --- | --- | --- | --- | --- | --- |
| public | Yes | Yes | Yes | Yes | Yes |
| protected | No | Yes | Yes | Yes | No |
| private | No | No | No | No | No |
| Default (no modifier) | No | Yes | Yes | No | No |

* **public**: Accessible everywhere.
* **protected**: Accessible within the same package and by subclasses.
* **private**: Accessible only within the class.
* **Default**: Accessible only within the same package.

**Summary of Use Cases:**

* **public**: Use when you want to provide access from any class or package.
* **protected**: Use when you want to allow access within the same package and to subclasses (even those in different packages).
* **private**: Use when you want to encapsulate details and prevent access outside the class.
* **Default (package-private)**: Use when you want to limit access to classes within the same package.

By carefully choosing the right access modifier, you can enforce good encapsulation practices and ensure that the right parts of your code are visible and accessible to others while keeping the rest hidden or protected.