perplexity

## Access Modifiers in Java: Structured Notes

Access modifiers in Java define the **visibility** and **scope** of classes, methods, variables, and constructors. There are four main access modifiers in Java: `private`, `default` (package-private), `protected`, and `public`.

### Table: Access Levels of Modifiers

| Modifier | Within Class | Within Package | Subclass (Other Package) | Other Packages |
|---|---|---|---|---|
| **private** | Yes | No | No | No |
| **default** (no modifier) | Yes | Yes | No | No |
| **protected** | Yes | Yes | Yes | No |
| **public** | Yes | Yes | Yes | Yes |

### Description of Each Modifier

### 1. private

- **Scope:** Only within the declared class.
- **Usage:** Hide implementation details, encapsulation.

### 2. default (No modifier)

- **Scope:** Package-private, visible to all classes in the same package.
- **Usage:** Group classes logically without exposing them outside the package.

### 3. protected

- **Scope:** Classes in the same package + subclasses (even in different packages).
- **Usage:** Allow inheritance but restrict access from unrelated classes.

### 4. public

- **Scope:** Everywhere.
- **Usage:** Make classes or members accessible from any part of the program.

## Examples

### 1. private Example

```
class Animal {
    private int age; // Only accessible within Animal class

    private void sleep() {
        System.out.println("Sleeping...");
    }
}
```

### 2. default Example (No modifier)

```
class Bird { // default access
    void fly() { // default access
        System.out.println("Flying...");
    }
}
```

`Bird` *and its* `fly()` *method can be accessed only within the same package.*

### 3. protected Example

```
class Plant {
    protected String type; // accessible in same package and subclasses

    protected void grow() {
        System.out.println("Growing...");
    }
}

class Flower extends Plant {
    void displayType() {
        System.out.println(type); // accessible due to 'protected'
    }
}
```

### 4. public Example

```
public class Main {
    public int number; // accessible from anywhere

    public void show() {
        System.out.println("Showing...");
    }
}
```

### Key Points

- If no access modifier is specified, the default is package-private (visible within the package).
- `protected` is more open than default, as it allows access from subclasses in other packages.
- `private` is the most restrictive and is commonly used for encapsulation.
- `public` is the least restrictive, making classes or members globally accessible.

### Proper Use Cases

| Modifier | When to Use |
| --- | --- |
| **private** | For sensitive data or helper methods, encapsulation. |
| **default** | For package-level grouping, helper classes(methods) not needed elsewhere. |
| **protected** | For inheritance, when subclasses need access to superclass members. |
| **public** | API classes/methods intended for use everywhere. |

These are the core concepts, distinctions, and examples you need to understand and use Java access modifiers effectively.