Core Principles of OOP:

1. **Abstraction**: Focus on essential qualities while ignoring irrelevant details.
2. **Encapsulation**: Bundling data with methods that operate on that data; restrict access to some components.
3. **Inheritance**: Mechanism for creating a new class based on an existing class (IS-A relationship).
4. **Polymorphism**: Ability to process objects differently based on their data type or class.

Classes and Objects:

Class: Blueprint for creating objects.

Object: Instance of a class.

Members: Properties (attributes) and methods (behaviors).

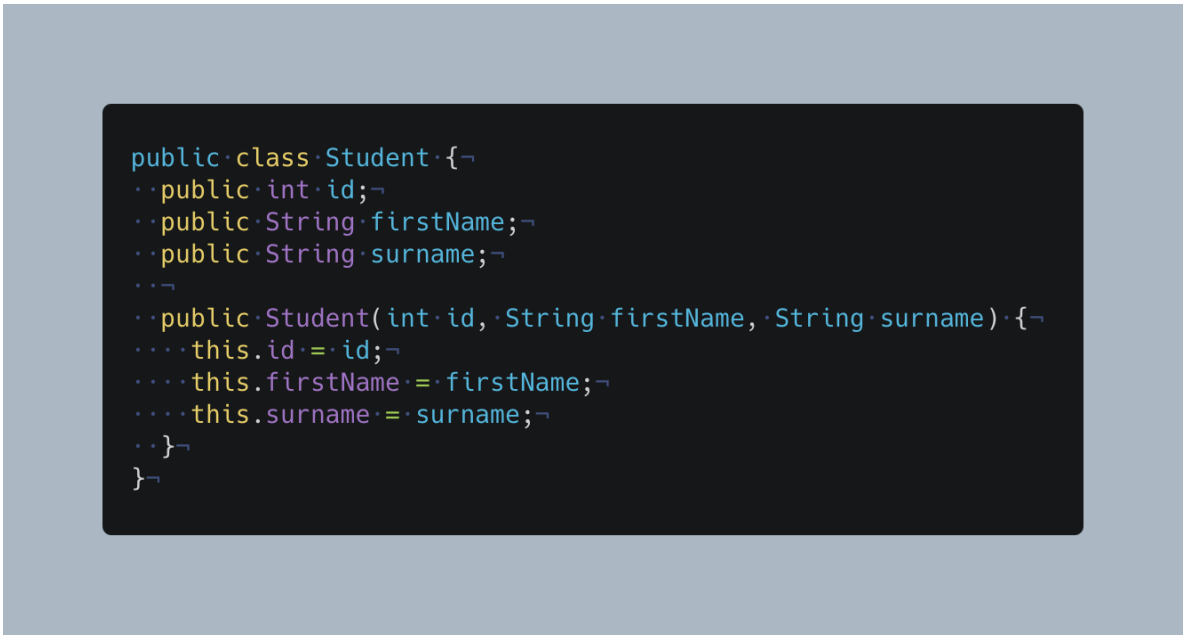
Constructors:

Definition: Special method for initializing objects.

Types:

- No-argument constructor: Initializes with default values.
- Parameterized constructor: Initializes with specified values.

Example:



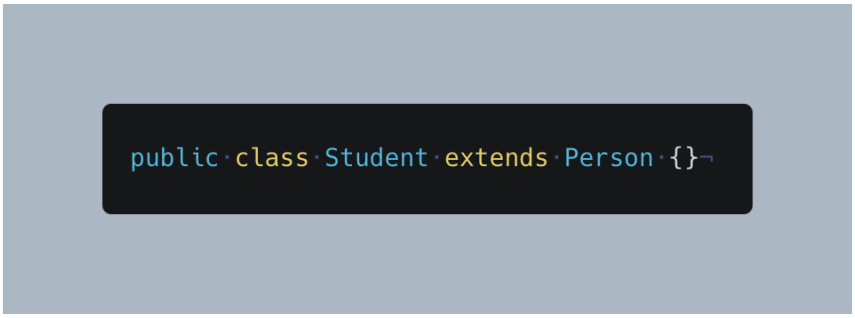
```
public class Student {
    public int id;
    public String firstName;
    public String surname;

    public Student(int id, String firstName, String surname) {
        this.id = id;
        this.firstName = firstName;
        this.surname = surname;
    }
}
```

Inheritance:

Definition: New class inherits properties and methods from an existing class.

Java Syntax:



Encapsulation:

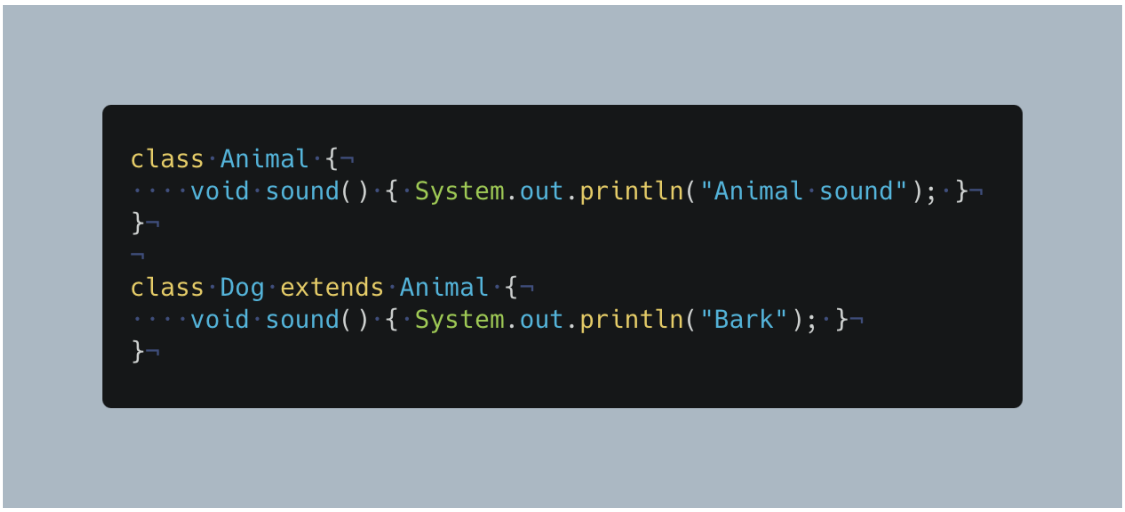
Access Modifiers:

- **public**: Accessible from any class.
- **private**: Accessible only within the defining class.
- **protected**: Accessible by subclasses and classes in the same package.

Polymorphism:

Method Overriding: Same method name, different implementation in subclasses.

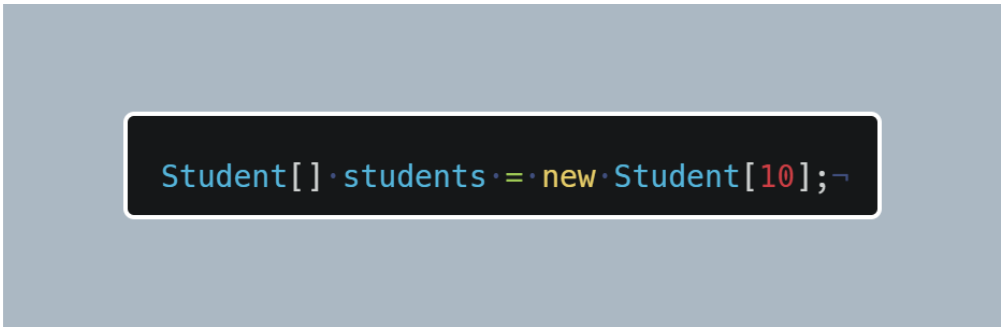
Example:



Collections and Arrays:

Definition: Used to store multiple objects.

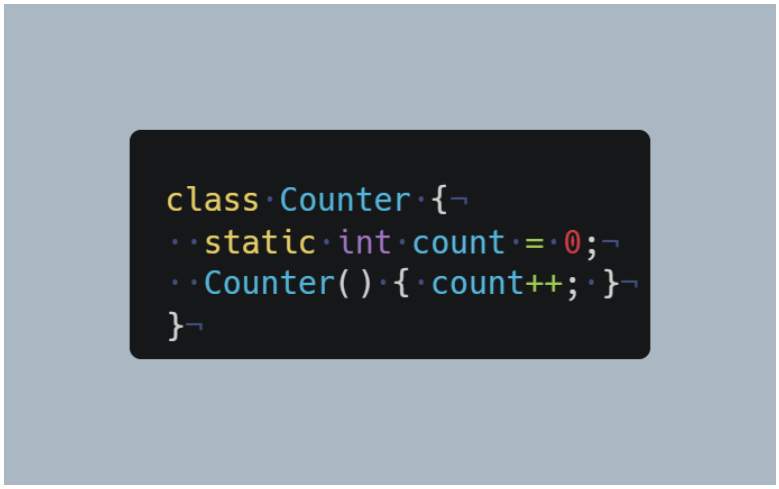
Example:



Static Members:

Definition: Belong to the class rather than any instance.

Example:

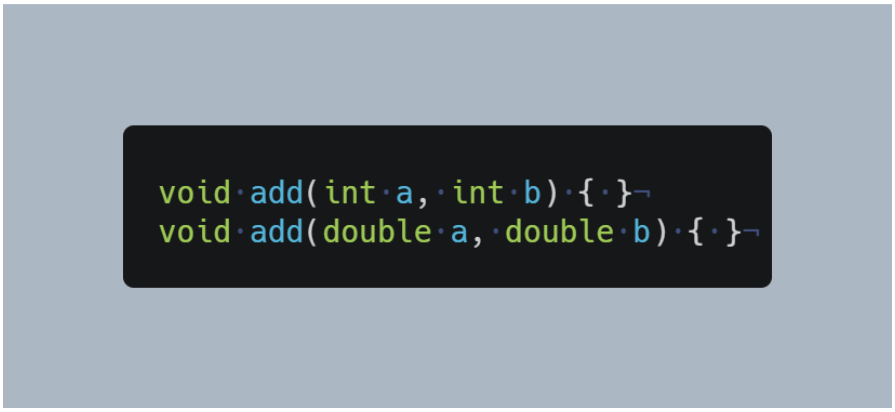


Every time a new counter class is instantiated, the count variable will increase by 1.

Method Overloading:

Definition: Multiple methods with the same name but different parameters.

Example:



Best Practices:

- **DRY Principle:** Don't Repeat Yourself.
- Use meaningful names for classes and methods.
- Keep methods focused on a single task.