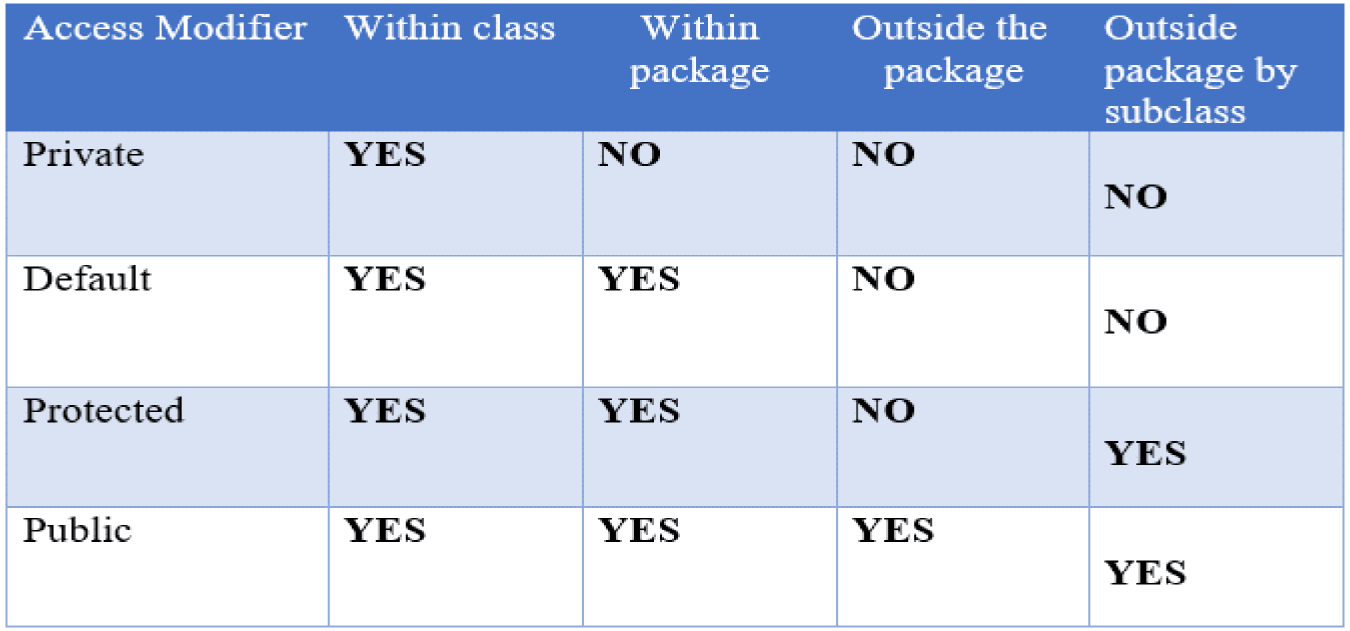
INTRODUCTION TO PROGRAMMING LANGUAGE II(JAVA)

Fariha Zahin
Lecturer
CSE, Southeast University

Access Modifiers in Java

Define the visibility or access level of classes, methods, and variables.
Four types of access modifiers:
1.private
2.default (no keyword)
3.protected
4.public



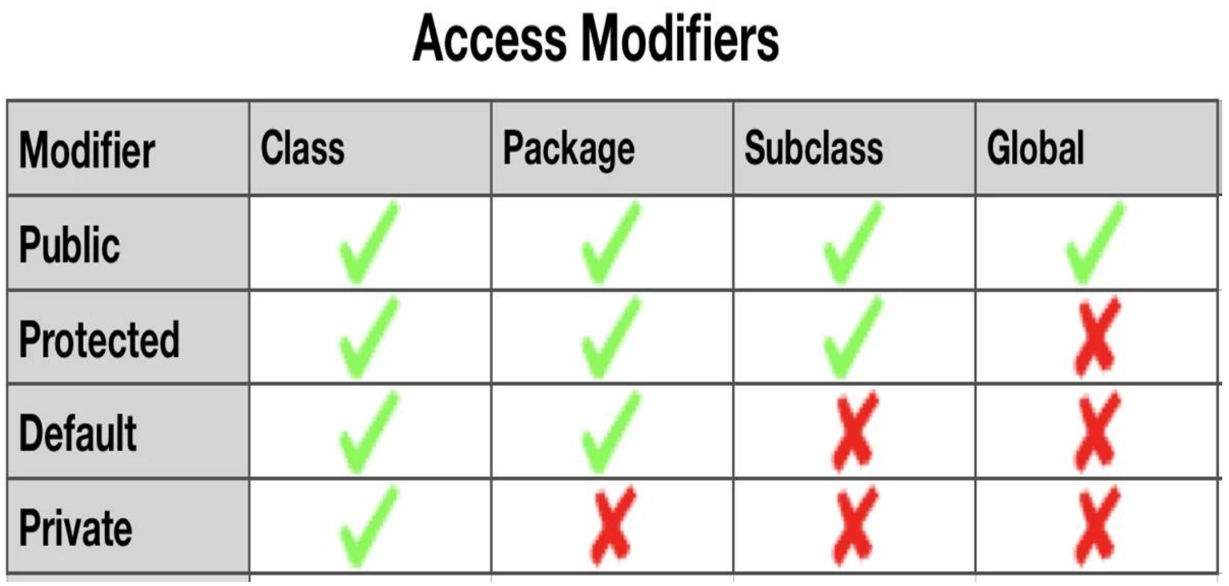
Access Modifier	Within class	Within package	Outside the package	Outside package by subclass
Private	YES	NO	NO	NO
Default	YES	YES	NO	NO
Protected	YES	YES	NO	YES
Public	YES	YES	YES	YES

Public Modifier
Accessible from anywhere in the project

Private Modifier
Accessible only within the same class

Protected Modifier
Accessible within the same package and in subclasses (even if they are in different packages)

Default Modifier
No modifier means package-private access.
Accessible only within the same package



Access Modifiers

Modifier	Class	Package	Subclass	Global
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
Default	✓	✓	✗	✗
Private	✓	✗	✗	✗

Encapsulation

Encapsulation is the process of wrapping data (variables) and methods into a single unit called a class
It restricts direct access to data members
Achieved by:
1. Declaring variables as private
2. Providing public getter and setter methods

Why Use Encapsulation?

- Control over data access
- Protects from unauthorized access
- Increases code maintainability and flexibility
- Enhances security of data

Getter and Setter Methods in Java

What is a Getter?

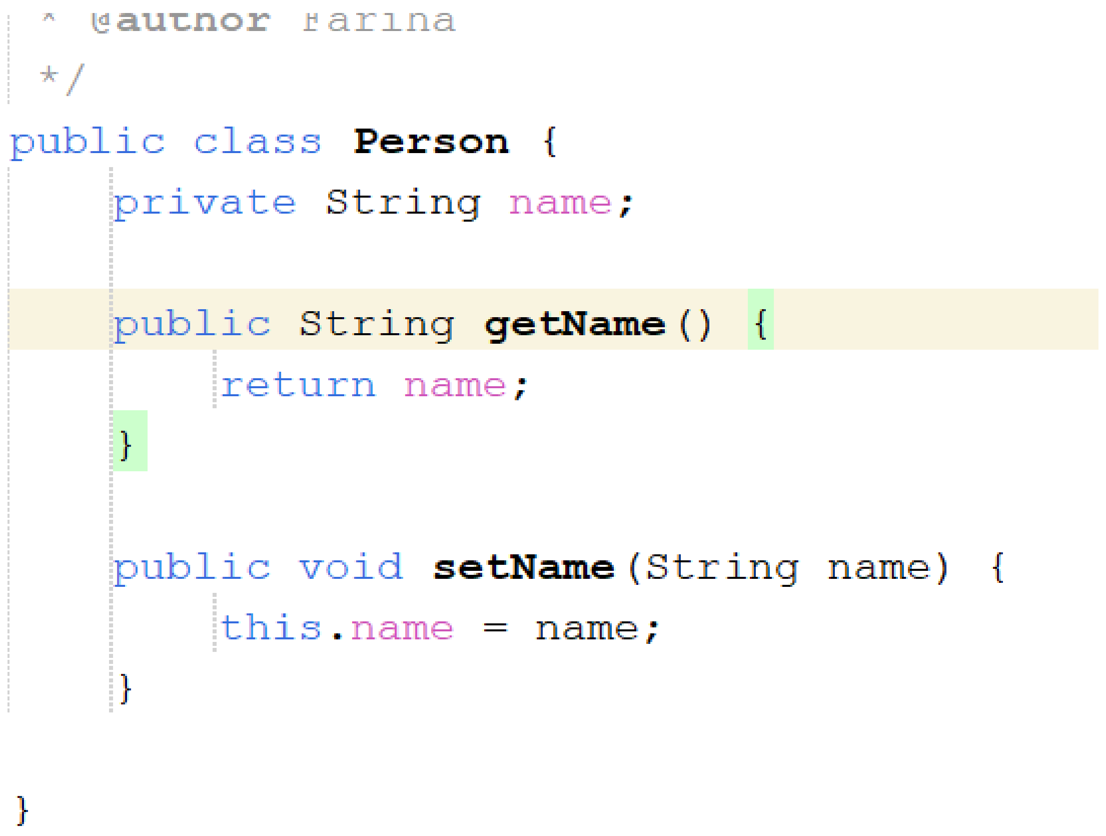
•A *getter* is a public method used to **retrieve the value** of a private variable.
•It allows **read access** to private data.

What is a Setter?

A *setter* is a public method used to **update or set the value** of a private variable.
It allows **write access**, often with validation.

Why Use Getters and Setters?

To implement **encapsulation**.
To **control access** to the data fields.
To **validate data** before changing it.
To **hide implementation details**.

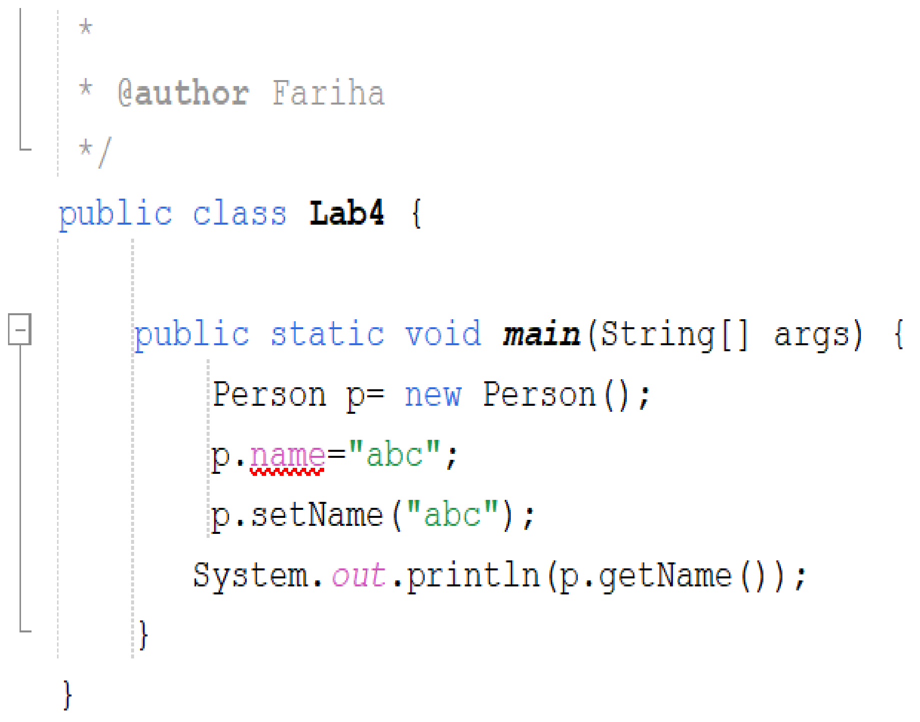


```
 * @author Fariha
 */
public class Person {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```



```
 *
 * @author Fariha
 */
public class Lab4 {

    public static void main(String[] args) {
        Person p= new Person();
        p.name="abc";
        p.setName("abc");
       System.out.println(p.getName());
    }
}
```

Accessing Private Members through Getters and Setters

- **Private members** are *not directly inherited*.
- Subclasses or other classes **cannot access private fields/methods** directly.
- They **can access** them via **public/protected methods** (getter/setter).