Object Oriented Programming ICT 1203

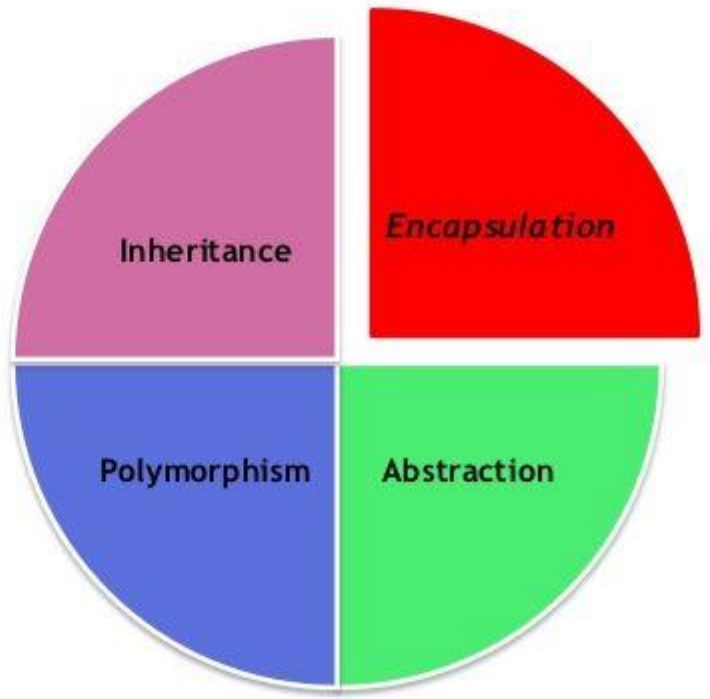
Dr. Jesmin Akhter

Associate Professor

IIT,JU

Encapsulation in JAVA

Introduction



Encapsulation is one of the four fundamental OOP concepts.

Introduction

- The process of binding data and corresponding methods (behavior) together into a single unit is called **encapsulation in Java**.
- Encapsulation is a mechanism of packaging the data (variables) and code acting on the data (methods) together as a single unit in order to protect the data from being accessed by outside of the package.

- The whole idea behind encapsulation is to hide the implementation details from the users.

- To understand encapsulation we first need to know about access modifiers in JAVA.

Introduction

- **Access modifiers in JAVA :**

There are four types of access modifiers in Java:

1. Private
2. Protected
3. Default
4. Public

If a data member is private, it means it can only be accessed within the class where it is declared. No outside class can access private data member of other class directly. The only way to access those private data is to set up public getter and setter methods in that class where the private data is declared.

Introduction

In encapsulation, private data and the methods by which they are accessible are hidden from other classes.

In this way data can only be accessed by public methods of their current classes. Therefore it is known as – **data hiding**.

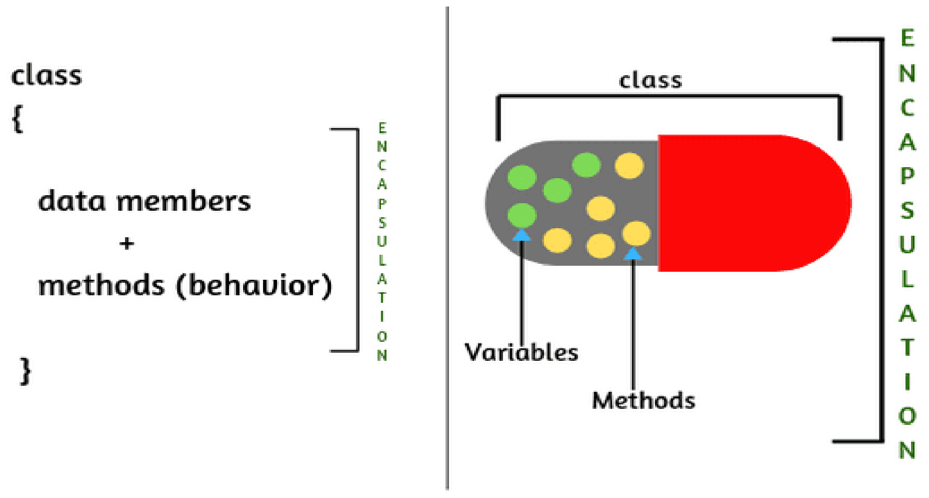


Fig: Encapsulation

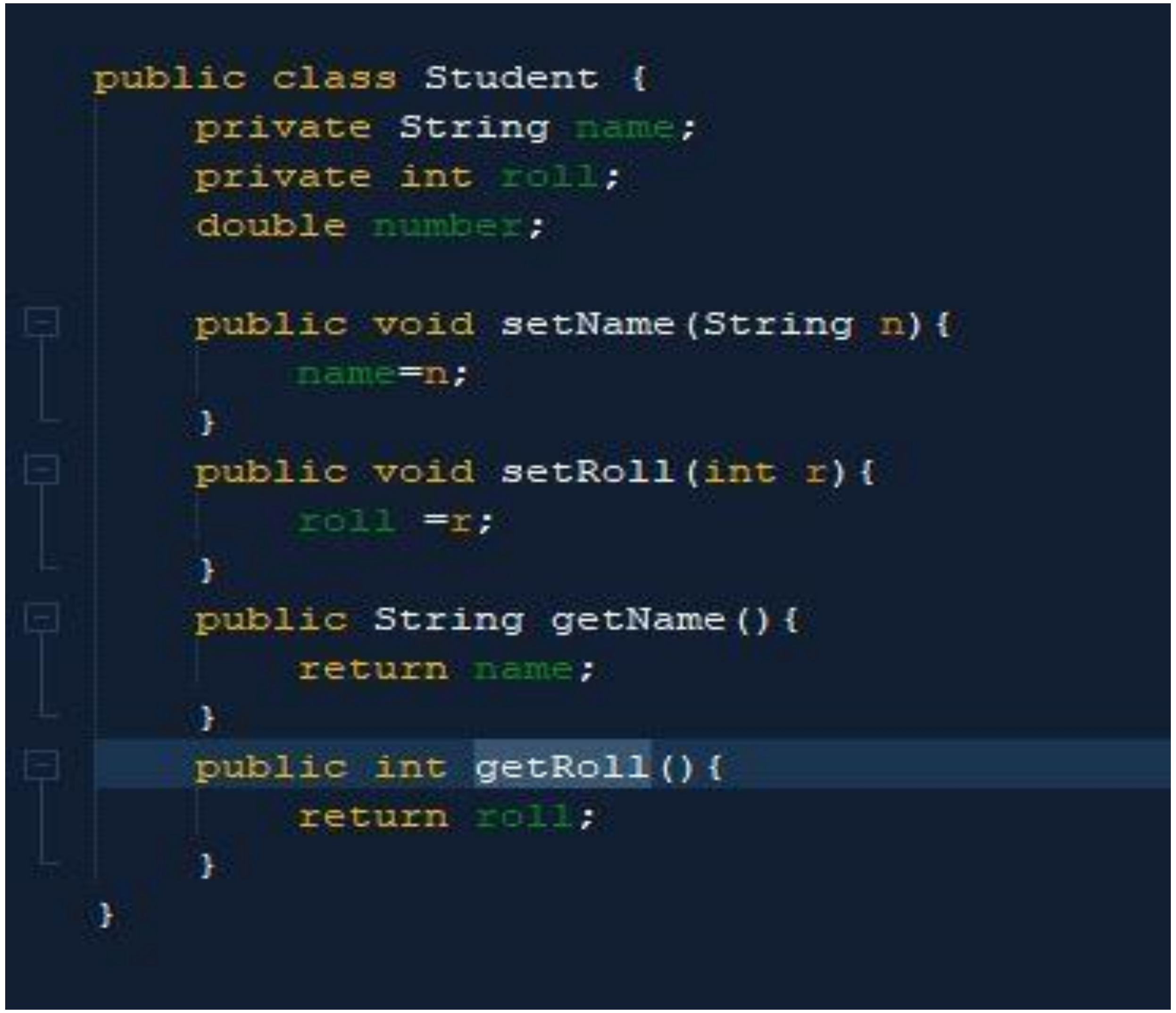
How to encapsulate

- Declare the variable of a class as private.
- Provide public setter and getter methods to modify and view the variables.

Note: Private member can only be accessed within the same class. An outside class can not access the data members of that class. If you need to access these variables, you have to use public “**getter**” and “**setter**” **methods**.

How to encapsulate

- Encapsulated class:



```
public class Student {
    private String name;
    private int roll;
    double number;

    public void setName(String n){
        name=n;
    }
    public void setRoll(int r){
        roll =r;
    }
    public String getName(){
        return name;
    }
    public int getRoll(){
        return roll;
    }
}
```

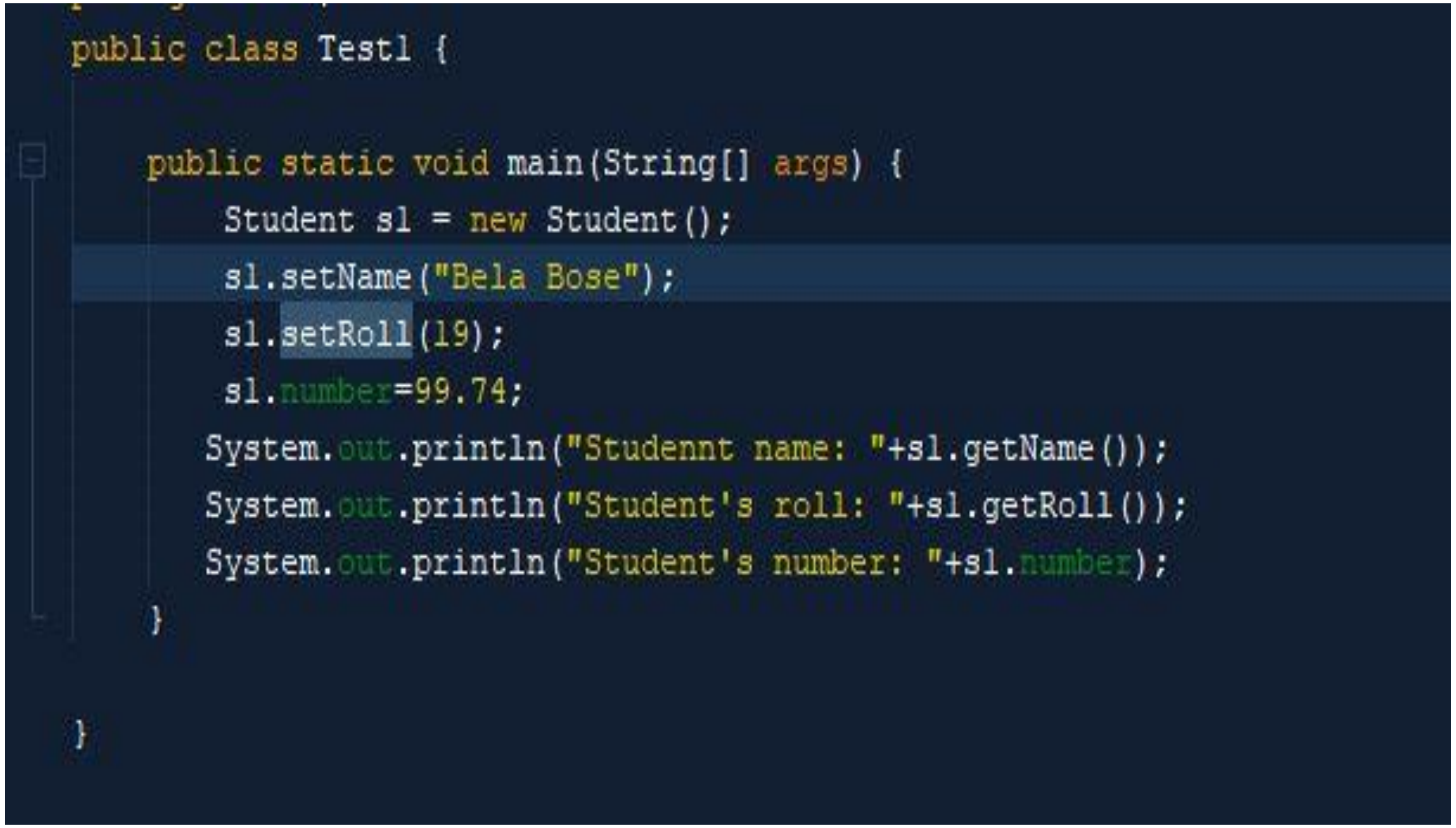
In this program the class **Student** is encapsulated as the variable (name, roll) are declared as **private**.

The get methods getName(), getRoll() and the set mathods setName(),setRoll() are set as public.

These methods are used to access **name** and **roll** variables from another class.

How to encapsulate

- Another class for users to access the private data form the encapsulated class:

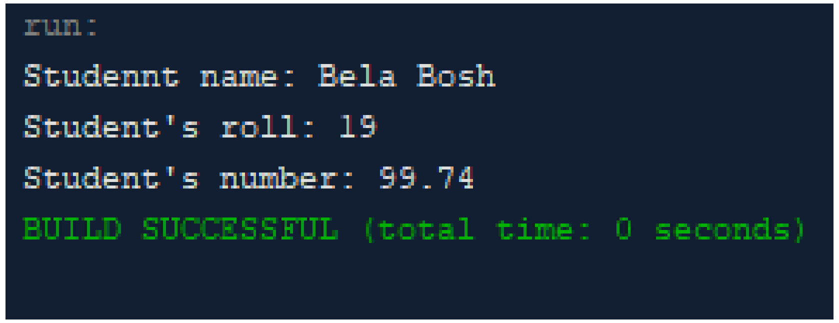


```
public class Test1 {

    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setName("Bela Bose");
        s1.setRoll(19);
        s1.number=99.74;
       System.out.println("Studennt name: "+s1.getName());
       System.out.println("Student's roll: "+s1.getRoll());
       System.out.println("Student's number: "+s1.number);
    }

}
```

Here, another class **Test1** works with data from the **encapsulated** class **Student**.

<span style="color:red">**Note**</span> : number variable is not declared as private at Student class, hence it can be accessed by another class Test1 directly.

Output:



```
run:
Studennt name: Bela Bosh
Student's roll: 19
Student's number: 99.74
BUILD SUCCESSFUL (total time: 0 seconds)
```

Advantages

- **Data hiding:** The user will have no idea about the inner code implement of the class. It will not be visible to the user that how the class is storing values in the variables.
- **Increased flexibility**: The variables of the encapsulated class can be made as read-only (using the get method only) or write-only (using the set method only) depending on our requirement.
- **Maintainability**: Hiding implementation details which reduces complexity as well as length of the program.
- **Reusability:** It allows a programmer to use the existing code again and again in an effective way.

Advantages

- **Testing of the code:** Ease of testing becomes easy. So it is better for unit testing.
- **Less error prone**: As Encapsulation allows modifying implemented code without breaking others code who have implemented the code.
- **Provides more control**: A class can have total control over what is stored in its fields.
- **Security**: With Java Encapsulation, you can hide (restrict access) to critical data members in your code, which improves security

Thank You