Encapsulation Learning Guide (Step-by-Step)

1. Basic Concepts

- Definition: Data + Methods in one unit, with data hiding.
- Advantages: Data security, code maintainability, data validation.
- Keywords: private, public, getter, setter.

2. Java Syntax & Simple Example

- Private variables: hide data.
- Getter/Setter methods: access data.

```
class Student {
    private String name;
    private int age;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getAge() { return age; }
    public void setAge(int age) { if(age > 0) this.age = age; }
}
```

3. Access Modifiers

- private: only inside class
- default/package-private: same package
- protected: same package + subclass
- public: everywhere

4. Data Hiding & Validation

```
public void setAge(int age) {
    if(age > 0 && age < 100) this.age = age;
    else System.out.println("Invalid age");
}
```

5. Constructor with Encapsulation

```
class Student {
    private String name;
```

```
    private int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

6. Encapsulation + Method Usage

```
class Employee {
    private double salary;

    public void setSalary(double salary) {
        if(salary > 0) this.salary = salary;
    }

    public double getSalary() { return salary; }
}
```

7. Advanced Practice Concepts

- Multiple objects with encapsulation.
- Validation in setters.
- Combination with other OOP concepts (inheritance, polymorphism).

8. Extra Concepts

1. Immutable Class: private final variables, only getters.
2. Encapsulation + Inheritance: subclass cannot access private variables directly.
3. Encapsulation + Polymorphism: getter/setter can be overridden.
4. Best Practices: all variables private, public getters/setters, validation in setters, use immutable objects.
5. Encapsulation vs Abstraction: Encapsulation hides data; Abstraction hides implementation.

9. Summary Chart

Concept	Keyword/Example	Notes
Private Variable	private int age;	Hidden from outside
Getter Method	public int getAge()	Access hidden data
Setter Method	public void setAge(int age)	Modify hidden data with validation

Concept	Keyword/Example	Notes
Constructor	public Student(String name, int age)	Initialize data
Advantages	Security, Maintainability, Validation	Keeps code safe & clean
Access Modifiers	private, protected, public	Control visibility
Immutable Class	private final int id;	Cannot modify after creation
Inheritance Access	Use getter/setter	Subclass cannot access private vars directly
Polymorphism	Override getter/setter	Flexible behavior
Best Practices	Private vars + getter/setter + validation	Secure and maintainable code
Encapsulation vs Abstraction	Encapsulation hides data, Abstraction hides implementation	Key difference