Encapsulation

Definition:

Encapsulation is the process of **wrapping data (variables)** and **methods (functions)** together into a single unit (class).
It is also called **data hiding**, since the internal details are hidden from outside access.

Key Points:

- Achieved by declaring variables as **private**.
- Access provided through **public getters and setters**.
- Protects data from unauthorized access and modification.
- Promotes **modularity and maintainability**.

Example :

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


   // Setter methods
   public void setName(String name) {
      this.name = name;
   }
   public void setAge(int age) {
      this.age = age;
   }
```

```
    // Getter methods
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
}

public class EncapsulationExample {
    public static void main(String[] args) {
        Student s1 = new Student();

        s1.setName("Ram");
        s1.setAge(33);

        System.out.println("Student Name: " + s1.getName());
        System.out.println("Student Age: " + s1.getAge());
    }
}
```

Abstraction

Definition:

Abstraction is the process of **hiding implementation details** and **showing only the essential features** of an object.
It focuses on **what an object does**, not **how it does it**.

Key Points:

- Achieved using **abstract classes** and **interfaces**.
- Helps reduce **complexity**.
- Increases **security** by hiding unnecessary details.
- Supports **polymorphism** (different implementations).

Example (Abstract Class):

```
abstract class Shape {
    abstract void draw();  // abstract method
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing Circle");
    }
}

class Square extends Shape {
    void draw() {
        System.out.println("Drawing Square");
    }
}

public class AbstractionExample {
    public static void main(String[] args) {
        Shape s1 = new Circle();
        Shape s2 = new Square();
```

```
        s1.draw();
        s2.draw();
    }
}
```

Example (Interface):

```
interface Animal {
    void sound();  // abstract method
}

class Dog implements Animal {
    public void sound() {
        System.out.println("Dog barks");
    }
}

class Cat implements Animal {
    public void sound() {
        System.out.println("Cat meows");
    }
}

public class InterfaceExample {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        Animal a2 = new Cat();

        a1.sound();
        a2.sound();
    }
}
```

Difference between Encapsulation and Abstraction

Feature	Encapsulation	Abstract Class	Interface
Main Purpose	Data hiding	Design hiding	Design hiding
Achieved by	Private vars + getters/setters	`abstract` keyword	`interface` keyword
Can have data?	Yes (private vars)	Yes (fields + methods)	Only constants (public static final)
Methods	Normal methods	Abstract + Concrete	Abstract (by default), default & static
Inheritance	Not related	Single inheritance	Multiple inheritance
Example use	Bank account balance	Shape hierarchy	Common behavior (e.g., Flyable, Runnable)