Object Oriented Programming

- Pratiksha Dhavale

4 pillars of OOP

Encapsulation

Abstraction

Inheritance

Polymorphism

Encapsulation

The meaning of Encapsulation, is to make sure that "sensitive" data is hidden from users. To achieve this, you must:

- declare class variables/attributes as private
- provide public get and set methods to access and update the value of a private variable

Example

```
public class Person {
    private String name;

    // Getter
    public String getName() {
      return name;
    }

    // Setter
    public void setName(String newName) {
      this.name = newName;
    }
}
```

Abstraction

Data abstraction is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interfaces

The abstract keyword is a non-access modifier, used for classes and methods:

- Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

Inheritance

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- subclass (child) - the class that inherits from another class
- superclass (parent) - the class being inherited from

To inherit from a class, use the extends keyword.

Example

```
class Vehicle {
  protected String brand = "Ford";        // Vehicle attribute
  public void honk() {                    // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar object
    myCar.honk();

    // Display the value of the brand attribute (from the Vehicle class) an
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
}
```

Polymorphism

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

In Java Polymorphism is mainly divided into two types:

- Compile-time Polymorphism
- Runtime Polymorphism

Compile time polymorphism

It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.

When there are multiple functions with the same name but different parameters then these functions are said to be **overloaded**.

Run time polymorphism

Method overriding