

BASICS OF OOP

AZNA AJENG | ASTRID CHAERIDA | NAURA KHANSA P

OBJECT ORIENTED PROGRAMMING

OOP in Java is based on declaring classes, creating objects from them and interacting between these objects.

This simplifies software development and maintenance as it is modeled and organized around objects and data rather than actions and logic.

CREATING OBJECTS USING OOP

1

DECLARING CLASSES

A class in Java is a blueprint which includes all your data which contains variables and methods

```
ex:
Class Test {
    member variables; // class body
    methods();
}
```

2

CREATING OBJECTS

It is an instance of a class which can access your data.

Object Structures :

- States (Data Fields/ Attributes)
- Behavior (Methods)

```
ex:
// Declaring and Initializing Object:
Object obj = new Object();
```

3

CONSTRUCTORS

Is a block of code that initializes a newly created objects either by default or parameterized

```
public class Test {
    int appld;
    String appName;
    //parameterized constructor with two
    parameters
    Test (int id, String name) {
        this.appld = id;
        this.appName = name;
    }
}
```

ENCAPSULATION

Process of binding the datas by adding a "private" syntax.

In order to access and modify the private variable values, we need to provide a getter and setter method.

```
public class Artist {
    private String name;

    //getter method
    public String getName() { return name; }

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

    // inside main
    Artist s = new Artist();

    //setting value in the name member
    s.setName("V");

    //getting value of the name member
    System.out.println(s.getName()); } }
```

ASSOCIATION

Is the relationship between objects.

This refers to how objects are related to each other and how they are using each other's functionality. Composition and aggregation are two types of association.

COMPOSITION - Object owns another object and another object cannot exist without the owner object. Consider the case of Human having a heart. Here Human object contains the heart and heart cannot exist without Human.

AGGREGATION - Both objects can exist independently. For example, a Team object and a Player object. The team contains multiple players but a player can exist without a team.

STATIC VARIABLE, METHOD, AND CONSTANT

1

STATIC MODIFIER (VARIABLE AND METHOD)

A variable can be used without first creating an instance of the class. A static class member is associated with the class itself, rather than an object. All class instances share the same copy of the variable.

For example, class myClass contains a static variable `days_in_week` :

```
public class myClass {
    static int days_in_week = 7;
}
```

It can be used elsewhere without explicitly creating a myClass object :

```
public class MyOtherClass{
    static void main(String[] args) {
        System.out.println
            (myClass.days_in_week); }
}
```

2

FINAL MODIFIER IN STATIC (CONSTANTS)

The final modifier means that the variable's value cannot change. Once the value is assigned, it cannot be reassigned.

```
static final int DAYS_IN_WEEK = 7;
```

Note that we declared in all caps once we added the final modifier.