Twitter Thread by Vikas Rajput



Vikas Rajput
@vikasrajputin

OOPs Concept in Java

a thread...

OOP Concepts are very common in Software Engineering.

All the OOP languages like Java, Python, C++, .Net, etc. have this concept.

In this thread, we'll try to understand it from Java's standpoint.

There are six different concepts in OOP:

1. Object
2. Class
3. Inheritance
4. Encapsulation
5. Abstraction
6. Polymorphism

Let's understand them one by one...

Object:

Look around you, whatever you see can be considered an Object.

For eg: The chair you are sitting on, the table, your computer, your house, windows, your pet, you yourself!

An object has Behaviour (things it does or performs) and Attributes (things that describe it).

Example:

Note: B = Behavior, A = Attribute

Chair:

B - Moves, Adjust Height, Recline, etc.
A - Built Material, Color, Make & Model, Brand, Price, etc.

You:
B - Eat, Play, Watch Netflix, Sleep, etc.
A - Height, Weight, Age, DOB, Name, SSN, etc.

Class:

The collection of all related objects is called Class.

Consider class as a general category which contains all the related objects inside it.

Eg:
Objects like Wheel chair, Office Chair, Wooden Chair can be apart of "Chair" class.

Other Class Examples:

"Table" Class can have different objects like Study Table, Office Table, Dining Table, etc.

"Vehicle" Class has Car, Truck, Bus, Plane as Objects.

"Car" Class can have Sedans, SUVs, HatchBacks, and Mini SUVs as Objects.

and many more like this...

Inheritance:

The way we inherited a few qualities from our parents like few look like their father, few skin types are the same as their mother, hairs, nails, height, body, etc.

Similarly,
A class can also inherit the qualities from a parent class.

For eg:

Phone Class can have two Child Classes:
TelePhone
MobilePhone

Both these child classes can inherit the "calling" behavior.

Encapsulation:

It means wrapping data into a single unit and securing it.

For eg:

Drug Capsule: Wraps different medicines into a single unit and protect them from outside environment.

Bank Locker: Wraps your valuables into a single unit(locker) and protects it via passcode.

Use of Encapsulation?

Encapsulation hides the data and wraps it in a single unit. It helps in two ways - a) It protects the data from any misuse by hiding it and b) Provides easy access to relevant data with a single unit by wrapping it.

Abstraction:

Hiding complexity from the user and showing only the relative stuff.

For Eg:
Car - All the complexity like engine, machinery, etc is hidden from you and only relevant part is shown like the brakes, accelerator, and gearbox.

Other Abstraction Example:

ATM Machine:
User is only given a option to insert card, take cash, enter PIN,etc. While the other complexities like communication with the bank server, validating PIN, checking balance, etc is hidden from user.

Use of Abstraction?

Abstraction helps in making a User-Friendly system and also protects it from doing any harm to our system by hiding the complexity. So that user can only focus on relative things.

Confused between Abstraction and Encapsulation?

Abstraction hides the Implementation details (complexity) and secures implementation, not data.

Encapsulation hides the data and secures it from unwanted use.

Polymorphism:

It means many forms. With the same name, it provides different forms.

For eg:
In Chess, we've 6 pieces - king, rook, bishop, queen, knight, and pawn. All of them "move" differently i.e. Bishop moves diagonally, Rooks move horizontal and vertical, etc.

All Chess piece performs a common behavior "move" but they all do it differently. The behavior name "move" is the same but it's still differently done by different Objects.

Use of Polymorphism:

We can implement standards that are common and that everybody understands, easier to remember.

In the above example "move" is a common behavior so everybody knows that the "move" function will move the chess piece as per the piece type.

Conclusion:

We can use the OOP concept to map any real-world situation into programming very easily. That's why OOP concepts are designed close to reality.

A better understanding of these concepts is going to decide how good programmers we will become in the future.