Intro –V  
what Is programming and paradigm – N   
unstructured and structured –V  
classes and objects –N  
pillars of OOPS and inheritance and polymorphism –V  
abstraction and encapsulation –N  
glimpse of implementation detaILS IF TIME PERMITS

**MY Part 1:**

1. What is programming and programming paradigm?

- Let me highlight the todays topic :

Object Oriented Progamming Paradigm,

So, firstly, for considering this topic, we must be 100% familiar with what is Programming and what is programming paradigm? Only then we can switch to specific case of it, and thats OOPs.

Anyone can quickly tell the audience : What is programming?

According to wiki pedia:

**Computer programming** (often shortened to **programming**) is a process that leads from an original formulation of a [computing](https://en.wikipedia.org/wiki/Computing) problem to [executable](https://en.wikipedia.org/wiki/Executable) [computer programs](https://en.wikipedia.org/wiki/Computer_program).

Programming involves activities such as analysis, developing understanding, generating [algorithms](https://en.wikipedia.org/wiki/Algorithm), verification of requirements of algorithms including their correctness and resources consumption, and implementation (commonly referred to as **coding**[[1]](https://en.wikipedia.org/wiki/Computer_programming" \l "cite_note-1)[[2]](https://en.wikipedia.org/wiki/Computer_programming" \l "cite_note-2)) of algorithms in a target [programming language](https://en.wikipedia.org/wiki/Programming_language)

Dont worry, This is definitely what programming is but generally we can interprete as, for any given task, we thinks how to do this, in steps and we do that means we execute our steps, and get our task done, it may may not go wrong, may be done better or anything, but this is programming in general sense.

In computer science, programming is formulating the solution in any programming language.

Now, tell me what is a programming paradigm?

- A **programming paradigm** is a style, or “way,” of **programming.**

Now, lets explore various programming paradigms rather I would say evolution of programming paradigms.

\*\* **VIDHI WILL TAKE OVER \*\***

**My Part 2:**

We have seen unstructured and structured programming, now let me enlist the features and facilities what we have achieved and what we are lacking!

What we have achieved by unstructured programming

1. Our problem is being solved nothing else.

We havent achieved any kind of efficiency and any other faciltity from unstrucred programming, but hard coding the whole stuff over there get our task done

What we have achieved by Strucured programming:

1. Reusability of code in terms of function
2. and Readability, easy maintainance and enhancements

Modular programmig aka strutured programing was good enough until softwares used be be on not that big scale.

As the business world change, large enterprice level softwares has to be designed at that time developers were not able to connect the programs with the real world scenarios or real life entities.

Apart from that, What they lacked still:

1. Privacy and securiy
2. Inheritance ( reusability in a broader sense)
3. Polymorphism ( making functions and operators behave different in different cases)

Object Oriented programming allows us to achieve all this properties

Object has attributes which it possess and methods which it can perform.

We can start it with introdunction of some most important key words such as

1. Classes
2. Objects
3. Instances
4. Instantiation

Class – How object will look like, we can say a blueprint/prototype of an object

Object or we can say instance is an individual object of a certain class

Instanctiation is the process of making instances

Now, Lets continue the example of withdrawing money from ATM machine and thinkinh the solution in Object Oriented manner

Consider the scenario:

An **account holder** withdraws money from the **bank account** using **credit card**.

Can you identify the objects ?

Yes, An account holder can be an object, bank account can be and similarly credit card can be an object

Consider the class of account holder

Various attributes it can have

and various methods it can perform

All bundled into a single object, very similar to real world account holder

Each of the objects can be connected with each other as required and can communicate, gives a solid structure to this system.

\*\* **Now, vidhi will continue about inheritance and polymorphism\*\***

my part 3:

Abstraction and Encaptulation:

Basic definitions of abstraction and encapsulation.

I would love to dive into more details of abstraction:

Abstaction : Hiding implementation details.

Focus on the definitions,

Here, we considered abstraction only in the sense that it abstacts the implementation details, but its way more broader concept.

We use a software-> we dont care about code. Code is hidden to us.

Code-> in higher level language, we dont care how its gonna get our task done from machine that understands only machine level language. Again abstaction.

Code -> assembly -> machine level. How machine level language i.e. bit stream of 0s and 1s interact with hardware and its components,

Through ICs and sequential and combinational circuits, that are ultimately based upon logic gates, which works on the electron flow. And we dont care about that flow and quantum mechanics of electron. Still we uses that software, this is abstraction, so, dont cosider only the first layer of abstraction.

Abstraction is layered from outer interface to quantum mechanics.

How do u move your hand?

You dont care about, how u made the wish to move hand, which is being transfered through neurons as electrical impulses via synapses and brain in return sends the signal to muscles responsible for that movement.

This all details, you dont care about, but still you use it. This is the real meaning of abstraction.

Now, get back to the workshop topic.

Data/ information hiding:

Public / private / protected

Question on data hiding

Its explainanation

Discussion about memory allocation and garbage collection

­