Week - 1

Day - 2

LLD course

Page No.

Date 13 05 2025

* Topic : OOPs Pillars

* Content Covered :

- Brief history of programming
- Procedural vs OOP explained
- Real-world understanding of OOP pillars
- Deep dive into Abstraction & Encapsulation.

* History of programming :

● Machine Language :

- Directly executed by CPU.
- Binary coding language. (0/1)
- ex : 01001100 1100101
- Prone to errors ×
- Tedious process ×
- Not scalable ×

● Assembly level language :

- used mnemonics / keywords.
- ex : MOV A, 61 H
- Tightly coupled with hardware ×
- Prone to errors ×
- Not scalable ×
- Tedious ×

● Procedural programming :

- Introduced -
  Functions , loops, blocks (if-else, switch)
- Code was like a "set of instructions", a procedure, step by step execution.

- Not suitable for enterprise level complex applications.

OO Programming :

- Limitations of procedural programs:
  - Real world modelling ✓
  - Data security ✓
  - Highly scalable & reusable applns. ✓

- OOPs solved all these limitations.

i] Real world modelling :

- Bringing real world objects and relations in the applications

Objects ⟺ Interactions
↳ Behaviours
↳ characteristics

- For ex:

Car ⟹ characteristics
↳ engine
↳ Brand
↳ model
↳ wheels

Behaviours
↳ start ()
↳ stop ()
↳ gearshift(
↳ break ()

Owns

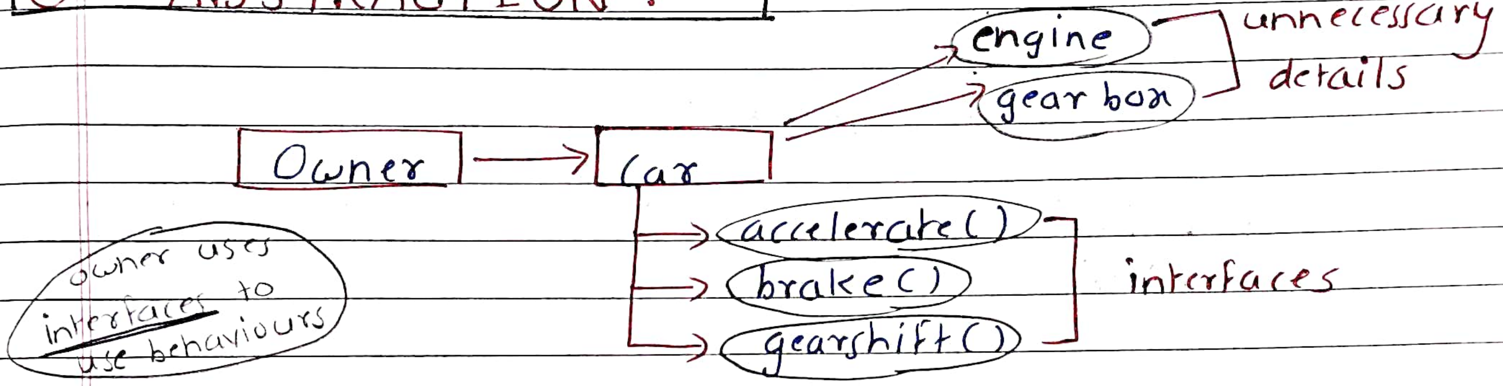
```
Owner {
    Car car;
    String name;
    void drive();
}
```

- Implementing above things is way too complex and not scalable in procedural prog.

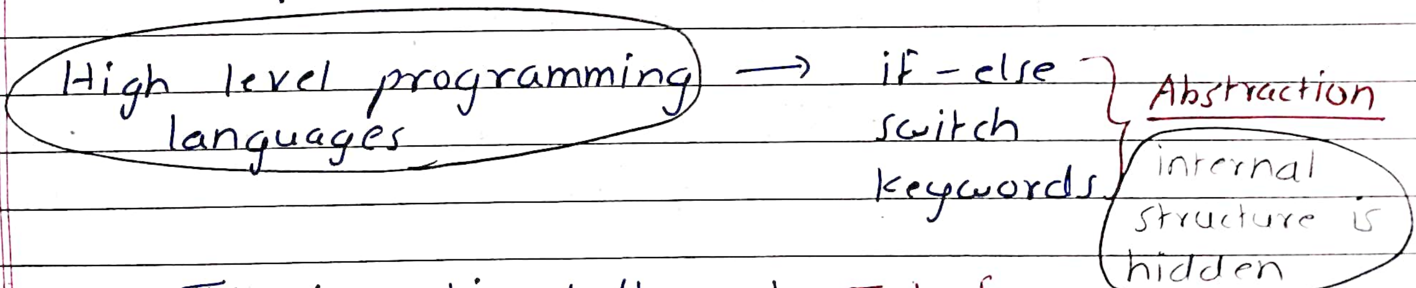
* PILLARS OF OOPs :

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

ABSTRACTION :

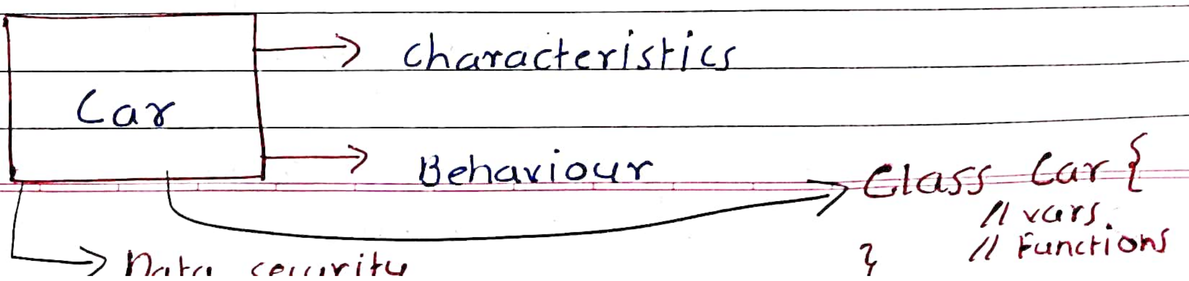


- Abstraction hides unnecessary details from a client, and showcases only what is necessary.
- example:

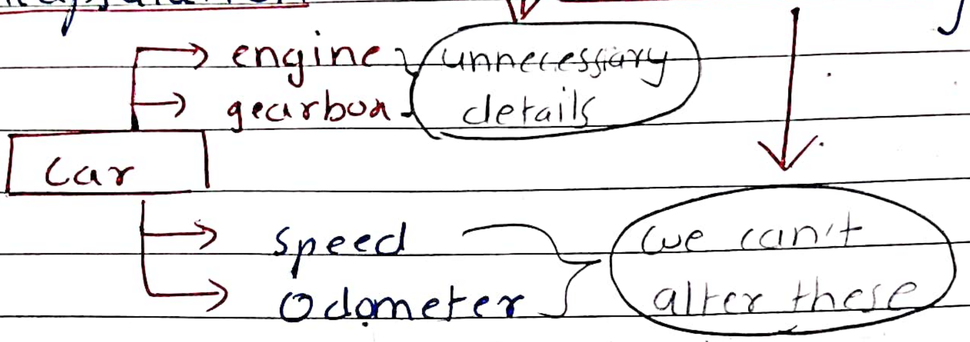

High level programming languages
if - else
switch
keywords
Abstraction
internal structure is hidden

- It is achieved through Interface

ENCAPSULATION :



- Abstraction $\Longrightarrow$ Data hiding
- Encapsulation $\Longrightarrow$ Data security



- Hence, Encapsulation does:

i) Binding characteristics & behaviours in single unit:

```
class car {
        // variable
        // methods
}
```

ii) Data security : through Access modifiers

- Access modifiers in java:
  - public - private - protected - default.

- It says not all the characteristics are accessible to everyone.

- It also includes the concept of :
  getters & setters
  to access the private members indirectly.
  which gives us the control over the variables of class through validations or pre-update checks.