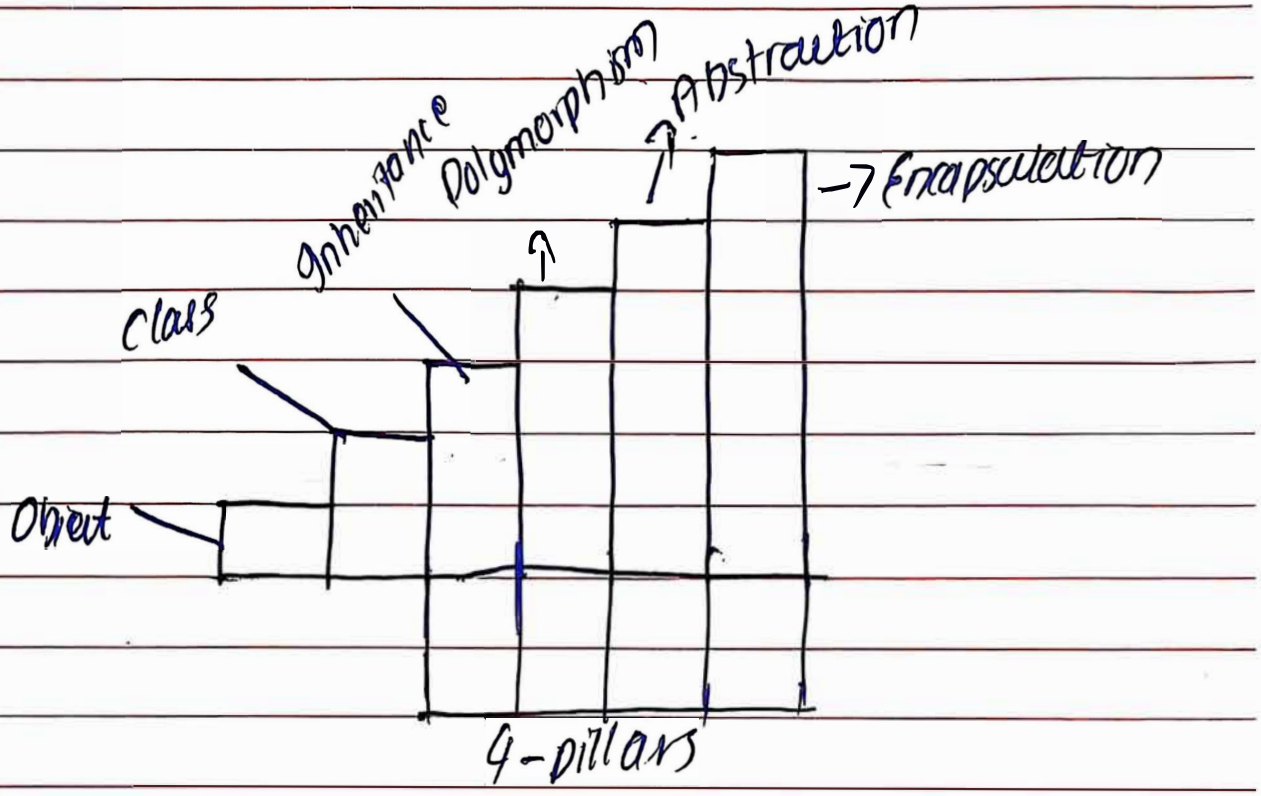
DATE _____

UNISH

OOP (WOOP)

~~Attributes~~

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects.



1) Class: A blueprint for creating objects. A class defines as a object according to its attributes (properties) and methods (functions) that an object can have

2) Object: An instance of a class. An object has attributes and methods defined by its class.

DATE _____

3) Inheritance: A way to create a new class from existing class. The new class (child class) inherits the attributes and methods of existing class (parent class)

4) Encapsulation: Bundling the data (instructions) and methods (functions) that work on the data into a single unit (class). This also restricts the direct access to some of an object's components, which can prevent accidental modification.

5) Polymorphism: The ability to present the same interface for different underlying data types. It allows methods to do different things based on the object it is acting upon.

6) Abstraction: Hiding the complex implementation details and showing only the essential features of the object

DATE ____ ____

Imagine we are playing ~~Legos~~ LEGOS.

OOP is like playing with LEGOs, but instead of building with plastic bricks, we use code.

OOP is a way to organize and built programs by creating "blueprints" for things, just like the instructions to built a LEGO car, house etc,. In OOP blueprints are called ~~classes~~ objects.

Taking an example of a Pet Store Game

A pet store game may have pets like cats, dog, birds. Each birds has things that describe it, like name, color, age and these pets can eat, sleep, make sounds.

In OOP, you will create a class called Pet that'll be a "blueprint" for all the pets in the store. Then, you can create objects (actual pets) like a Dog named SHERU cat named TUTTUT using that blueprint.

DATE _____

1. Class (Blueprint for pets)

=> Think of pet class like a general instruction manual for any type of pet in the pet store. It defines what a pet should have (name, color, type) and what it can do (make sounds, eat, sleep)

2. Objects (Actual pets in your store)

=> When we create dog and cat, we are making actual pets using the pet class (blueprint). Each one is a little different — Dog is Sheru, who is brown — Cat is Tut Tut, who is orange. All can eat but make their sounds depending what type of pet are they.

3. Methods (things pet can do)

=> Methods like [eat()] and [make-sound()] are the actions that a pet can perform. All pets can eat but their sounds depends on what they are.

4. Inheritance (getting traits from their parents)

=> If you want a specific class, like Dog or Cat, that have special behaviours of their own and some can have unique behaviours others dont have, inheritance can be used here. "Some dogs can fetch"

DATE _____

5) Encapsulation (keeping details inside)

=> when we call [dog.eat()], you dont need to know how it does it internally, it just does

6) Polymorphism (many forms):

=> If both dogs and cats have a [make_sound] method, polymorphism let us call the same methods [make_sound] and get different results (woof and meow) depending on the pet.