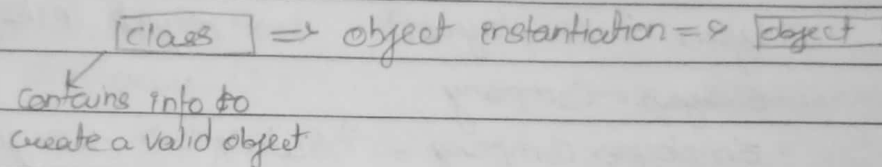
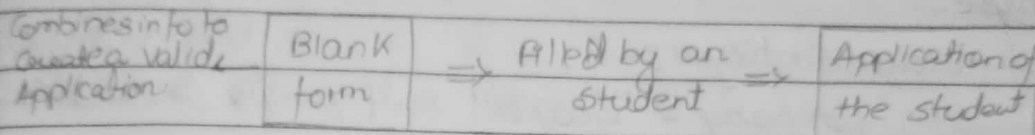


* CHAPTER-10 : OBJECTED ORIENTED PROGRAMMING

- Solving a problem by creating objects is one of the most popular approaches in programming. This is called Object Oriented Programming.
- This Concept focuses on using reusable code
 ↳ Implements DRY principle

→ CLASS

A class is a blueprint for creating objects.



- The Syntax of a Class looks like this :

Class Employee : # methods & variables	classname is written in Pascal Case
---	-------------------------------------

→ Object

An object is an instantiation of a class. When class is defined, a template (info) is defined. Memory is allocated only after object instantiation.

Objects of a given class can invoke the methods available to it without revealing the implementation details to the user.
 → Abstraction & Encapsulation!

- Modelling a problem in OOPs.
We identify the following in our problem.

Noun → Class → Employee
Adjective → Attributes → name, age, salary
Verb → Methods → getSalary(), increment()

- Class Attributes

An attribute that belongs to the class rather than a particular object.

Example:

Class Employee:

Company = "Google" → [Specific to Each Class]

Shayan = Employee() → object instantiation

Shayan.Company

Employee.Company = "Youtube" → changing class attribute

- Instance Attributes

An attribute that belongs to the instance (object)

Assuming the class from the previous example:

Shayan.name = "Shayan"

Shayan.salary = "30k" ⇒ Adding instance attributes

NOTE: Instance attributes take preference over class attributes during assignment & retrieval.

Shayan.attributes → ① Is attribute present in object?

② Is attribute present in class?

- 'self' Parameter
self refers to the instance of the class. It is automatically passed with a function call from an object.

harvey.getSalary() → here self is harvey

↳ Equivalent to Employee.getSalary(harvey)

The function getsalary is defined as:

```
class Employee:
    company = "Google"
    def getSalary(self):
        print("Salary is not there")
```

- static method
Sometimes, we need a function that doesn't use the self parameter. We can define a static method like this:

@ static method

```
def greet():
    print("Hello User")
```

→ decorator to mark greet as a static method.

- __init__ () Constructor

__init__ is a special method which is first run as soon as the object is classified, also known as Constructor. It takes argument (self) & can also take further arguments.

For Example :

```
class Employee:
    def __init__(self, name):
        self.name = name
```

```
def getSalary(self):
```

...

Shayan = Employee("Shayan")

↳ object can be instantiated using constructor like this !