Reynolds CHAMP

Date: Page:

* 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.

Combines info to create a valid Application	Blank form	⇒ Filled by an student ⇒	Application of the student

Class ⇒ object instantiation ⇒ object

Contains info to create a valid object

- The Syntax of a Class looks like this :

```
Class Employee :
        # methods & variables
```

Classname is written in PascalCase

→ Object

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

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

Scanned with CamScanner

Date: Page:

Modelling a problem in OOPs.

We identify the following in our problem.

Noun → Class → Employee
Adjective → Attributes → name, age, salary
Verbs → Methods → get_salary(), 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 & retrival

Shayan.attributes → ① Is attribute present in object?
② Is attribute present in class?

Scanned with CamScanner

Date: Page:

- 'self' Parameter

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

harry.getSalary() → here self is harry
↳ Equivalent to Employee.getSalary(harry)

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():                    → decorator to mark greet
    print("Hello User")           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")
```

↳ objected can be instantiated using Constructor like this!

Scanned with CamScanner