Prabodh
PAGE No.:
DATE:

Chapter - 10
Object 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] ⇒ filed by an student ⇒ [Application of the student]

[class] ⇒ object instantiation ⇒ [Object]

Combine info to creat a valid object

The syntax of a class looks like this:

```
class Employee:
    # methods and variables
```

[classname is written in Pascacase]

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
Verbs → Methods → getSalary(), increment()

Class Attributes

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

Example:

```
class Employee:
    company = "Google"  → specific to each class

Abhi = Employee()  → object instantiation
Abhi.company
Employee.company = "YouTube"
```

↳ Changing class attributes

Instance Attributes

An attributes that belongs to the instance (Object) assuming the class from the previous example.

```
Abhi.name   = "Abhi"
Abhi.salary = "30k"   → adding instance attributes
```

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

Abhi attribute1 → ① Is attribute1 present in object?
② Is attribute1 present in class?

'Self' Parameter

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

```
Abhi.getSalary()  → Here self is Abhi.
   └→ equivalent to Employee.getSalary(Abhi)
```

The function getsalary is defined as:

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

Static Method

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

```
@staticmethod          -> decorator to mark greet
def greet():              as a static method
    print("Hello manisha")
```

__init__() Constructor

__init__() is a special method which is first run as soon as the object is created. __init__() method is also known as constructor

It takes self argument and can also take further arguments.

Example:

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

    def getSalary(self):
        ...

abhi = Employee("Abhi")
```

↳ Object can be instantiated using constructor like this!

Chapter -10
Practice Set

Que 1. Create a class programmer for storing information of few programmers working at microsoft.

Que 2. Write a class calculator capable of finding squar, curve and squareroot of a number.

Que 3. Creat a class with a class attribute a, create an object from it and set a directly using object.a = 0. Does this change the class attribute ?

Ques 4. Add a static method in problem 2 to greet the user with "Hello".

Ques 5. Write a class train which has methods to book a ticket, get status (no of seats) and get fare information of trains running under Indian Railways

Que 6 Can you change the self parameter inside a class to something else (say 'Abhi') . Try changing self to 'slf' or 'abhi' and see the effects.