OBJECT ORIENTED PROGRAMMING WITH REAL WORLD EXAMPLES WHITEPAPER

STACKPROGRAMR.IN

DIPESH A. PARAB

01/06/2019

Object Oriented Programming

Introduction

Hello, I'm Dipesh Parab from Mumbai. I am software developer usually code in Java and Python. If you have done programming before you must have come across something called Object Oriented Programming or Oops. This Oops is not the one we use to show recognition of a mistake, often as part of an apology. It is rather very interesting topic in programming. So our objective for today is to learn about this Object Oriented Programming.

First thing first, Dipesh what is Object Oriented Programming? Why is it so important?
Let's, see what some of the recognized sources have to say about it.

Object-oriented programming is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields, and code, in the form of procedures. **- Wikipedia**
Object Oriented programming is a programming style that is associated with the concept of Class, Objects and various other concepts revolving around these two, like Inheritance, Polymorphism, Abstraction, Encapsulation etc. **- studytonight.com**
The programming paradigm where everything is represented as an object is known as truly object-oriented programming language. **- javatpoint.com**

Well if you understand anything from it then congratulations. You are smart and fast learner. My objective here is not only to communicate with those who understand technical topics with ease but also to make everyone understand what Object oriented programming is and how we can relate it with real life examples.

From above you must have seen few things which are classes, objects and four principles of object-oriented programming - encapsulation, abstraction, inheritance, and polymorphism.

I will explain you these topics with some examples to first establish basic understanding. Then we will move to technical aspect of it.

Real World Examples

We will divide it into two parts:

1. Classes and Objects

Let me give you one real world example. Let's say we have vehicle class with attributes as type, class, brand, model, color, no_of_wheels, no_of_seats, engine_type, engine_capacity, steering_type, etc. and methods such as start, drive, break, stop, horn,etc. If i want to make any car out of it then I can easily make. It means my car will be the object or instance of a class vehicle.

Let's make a new vehicle then -

My vehicle attributes are:

type = car ,class = sedan, brand = suzuki , model = dzire zxi, color = blue, no_of_wheels = 4, no_of_seats = 4, engine_type = petrol, engine_capacity = 1.2cc, steering_type = leather

my vehicle methods are:

start()
drive()
break()
horn()
stop()

My 'suzuki dzire' will start when i start it. I can drive my car, apply breaks, i can blow horn whenever needed and stop it. So i can make any object instance from class vehicle. Because class vehicle is my blueprint for creating n number of car objects.

2. Principals of Object oriented programming

The four principles of object-oriented programming are abstraction, encapsulation, inheritance, and polymorphism.

Abstraction:

Abstraction means showing only the important part or the part which is relevant to the user without exposing internal mechanism. While driving a car you only need to know how to start or stop it, how to drive it, apply a break or blow a horn. You don't need to know how all this mechanism is working internally. So this concept is abstraction which is used in oops.

Encapsulation:

Encapsulation = en + capsule

Encapsulate means binding something together inside one unit just like medicine is bind inside a capsule.

In Oops, encapsulation ensure two things

1. Data hiding and security from external interference
2. Classes separated and prevent them from having tightly coupled with each other

In a vehicle, there are different encapsulated mechanisms which do not affect one another. Like, wheel steering mechanism won't affect AC mechanism or door mechanism won't affect engine mechanism. They are encapsulated as separated entities.

Inheritance:

Inheritance is similar to the way a children inherit some attributes from their parents. Yes, in oops a class can be a parent of another class which is also known as base class. A class which inherits attributes and methods is known as child class.

Let's say vehicle is our parent class. There is another class called supercars. supercars class will inherit vehicle class so it do not have to define all the vehicle attributes again. It can define any other attributes required to make a supercars. This way supercars class is a child of vehicle class.

You may know that parents can have multiple children and those children may have multiple children.

Similarly, in oops there are three levels of inheritance.

1. Single inheritance - from vehicle to supercars
2. Multiple inheritance - from vehicle to supercars, superbikes
3. Multilevel inheritance - from vehicle to supercars, from supercars to racecars

Polymorphism:

Have you ever used similar method to perform different task. Let's take our car example again. When you accelerate the engine then depending upon which gear is engaged different amount power and movement is delivered to the car. so you are doing same functionality of accelerating a car but based on gear levels car will move differently at different power and pace.

Coding Example

For this demonstration I will use python as my programming language. You can learn or practice with any object-oriented programming language you want.

Let's create class as we created above:

```
class vehicle:
 def __init__(self, vehicle_type, vehicle_class, brand,model, color, no_of_wheels, no_of_seats, engine_type, engine_capacity, stearing_type):
   self.vehicle_type = vehicle_type
   self.vehicle_class = vehicle_class
   self.brand = brand
   self.model = model
   self.color = color
   self.no_of_wheels = no_of_wheels
   self.no_of_seats = no_of_seats
   self.engine_type = engine_type
   self.engine_capacity = engine_capacity
   self.stearing_type = stearing_type

 def start_vehicle(self):
   print("Car started")

 def stop_vehicle(self):
   print("Car stopped")

 def drive_vehicle(self):
   print("Car started driving")

 def apply_break(self):
   print("Breaks applied")

 def blow_horn(self):
   print("Horn blown")

#Here i have made my first vehicle from class vehicle.....we can add comments in code using '#' sign. It will not be a part of program execution
mycar1 = vehicle("Car","Sedan","Suzuki","zxi","Blue",4,4,"Petrol","1.2cc","Leather")

#using print() function to display my new car details
print("My first vehicle details:")
print("\nVehicle Type :",mycar1.vehicle_type,
    "\nClass :",mycar1.vehicle_class,
    "\nBrand :",mycar1.brand,
    "\nModel :",mycar1.model,
    "\nColor :",mycar1.color,
    "\nNo. of Wheels :",mycar1.no_of_wheels,
    "\nNo. of Seats :",mycar1.no_of_seats,
    "\nEngine Type :",mycar1.engine_type,
    "\nEngine Capacity :",mycar1.engine_capacity,
    "\nStearing Type :",mycar1.stearing_type)
```

Output:

```
My first vehicle details:

Vehicle Type : Car
Class : Sedan
Brand : Suzuki
Model : zxi
Color : Blue
No. of Wheels : 4
No. of Seats : 4
Engine Type : Petrol
Engine Capacity : 1.2cc
Stearing Type : Leather
```

By creating new object instance, i can make any type of vehicle such as bus, bike, truck with the help of my class as a blueprint.

Feedback

With this knowledge, you have put your first step to the introduction of object-oriented programming languages such as Java, Python, C++, etc.

If you have any query, you can contact me on my email - dipeshanandparab@gmail.com

Till the next time, keep learning, keep coding and keep exploring new stuff.

```
def say_thank_you():
        print("Thank You!")

say_thank_you()
```

Thank You!

- **Dipesh Anand Parab**