Object-Oriented Programming

April 2025

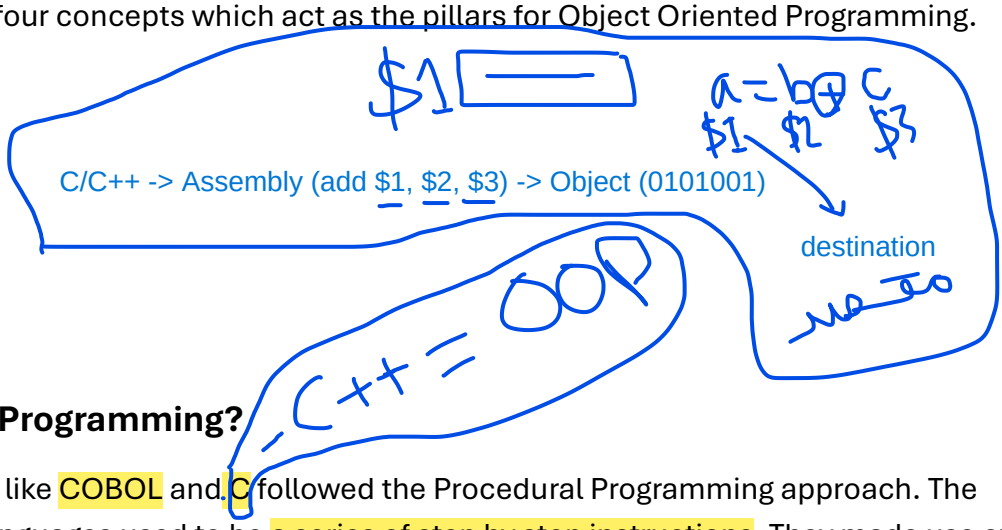
Prepared by: Mohamed Emad El-Dien

Introduction

Complex real life requirements need to be model into instructions for the computer to understand. We have a few paradigms like Procedural Programming, Object oriented programming, Functional Programming etc. which can help in modeling the business requirements into computer programs. We will be learning about one such paradigm called Object oriented programming. OOP is a philosophy and is language neutral.

We will be taking a look at the four concepts which act as the pillars for Object Oriented Programming.

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism



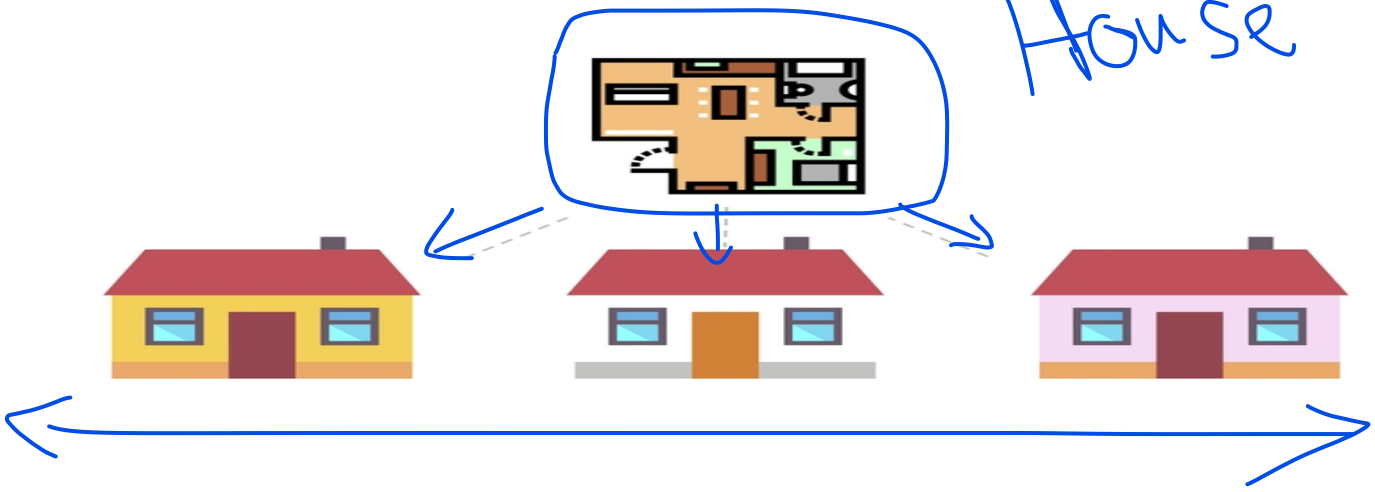
What is Object Oriented Programming?

Older programming languages like COBOL and C followed the Procedural Programming approach. The program written using these languages used to be a series of step by step instructions. They made use of procedures/subroutines for making the program modular. This programming paradigm focused on logic more than data and the program used to combine both of them together.

Modern programming languages like Java, C# etc. follow the Object Oriented approach. In object oriented programming, importance is given to data rather than just writing instructions to complete a task. An object is a thing or idea that you want to model in your program. An object can be anything, example, employee, bank account, car etc.

Class, Object.. what's that?

For getting started with object oriented programming we would have to know what is a class and object and the difference between them. A class is a blueprint for creating an object. It is similar to the blue print of a house.



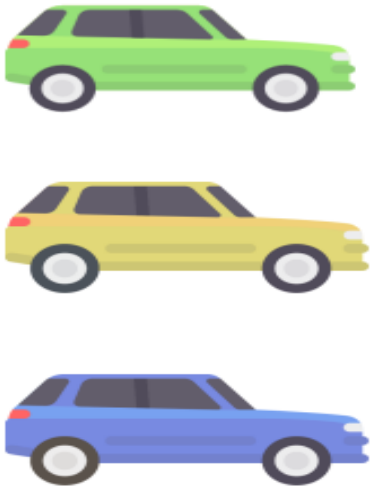
A class defines *attributes and behavior*. If we are to model a car in our application then we could define attributes of the car like, model, fuel, make and behaviors like start, break, accelerate etc.. If you notice, the attributes and behavior that we are specifying are not specific to just one model of car. We are trying to generalize a car by stating that the car which we are going to model in our program will have these number of attributes and behavior. There might be others as well but our scope and interest for the business requirement are limited to these attributes. This helps us create a blue print of the car and later while we use this class for creating objects we create car objects with specific details.

```
public class Car{
    private string _color;
    private string _model;
    private string _makeYear;
    private string _fuelType;

    public void Start(){
        ..
    }

    public void Stop(){
        ..
    }

    public void Accelerate(){
        ..
    }
}
```



Example, using the same class "Car" we can create different objects having variation in model, fuel type and make year while having the same common behavior.

Object 1	**Object 2**
Model Volkswagen Polo	Model Volkswagen Vento
Fuel Petrol	Fuel Diesel
Make 2017	Make 2017
Start() Break() Accelerate()	Start() Break() Accelerate()

In this way, Object oriented programming allows you to easily model real world complex system behavior. With OOP, data and functions (attributes and methods) are bundled together within the object. This prevents the need for any shared or global data with OOP, which is a core difference between the object oriented and procedural approaches.

Abstraction

Abstraction lets you focus on what the object does instead of how it is done. The idea behind abstraction is knowing a thing on a high level. Abstraction helps in building independent modules which can interact with each other by some means. Independent modules also ease maintenance.

Abstraction means to represent the essential feature without detailing the background implementation or internal working detail.

We try to selectively focus on only those things that matter to us or in the case of programming, to our module. Modifying one independent module does not impact the other modules. The only knowledge one needs to know is what a module gives you. The person who uses that module does not need to bother about how the task is achieved or what exactly is happening in the background.

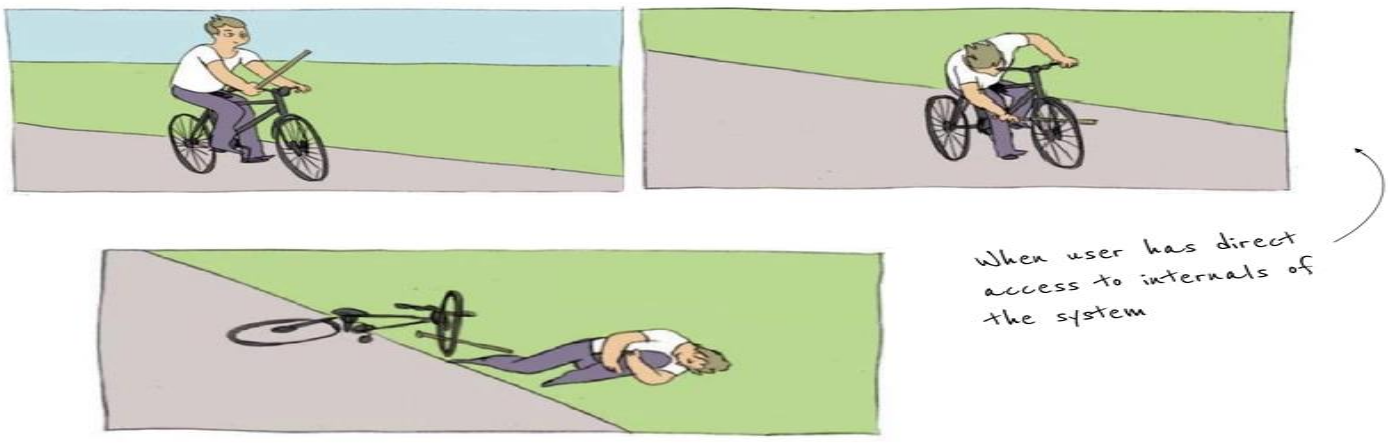
Abstraction is everywhere. Everyday objects that we use has abstractions applied at various levels. One example is applying breaks in your car or bike. The breaking system is abstracted and you are provided with a paddle for stopping your vehicle. Making changes to acceleration system does not affect the braking system they are independent. You also do not have to bother about the internal working of the brakes, you only have to press the brake pedal and be it disc brake or drum brake, the vehicle stops.

Encapsulation

The second concept Encapsulation is closely related to Abstraction. Encapsulation is all about exposing a solution to a problem without requiring the consumer to fully understand the problem domain.Encapsulation is binding the data and behaviors together in a single unit. This prevents the client or the user of the module from knowing about the inside view where the behavior of the abstraction is implemented.

The data is not accessed directly. It is accessed through the exposed functions. Hiding the internalsÂ of the objectÂ protects its integrityÂ by preventing users from setting the internal data of the component into an invalid or inconsistent state.

Encapsulation isÂ not just about hiding complexity but conversely exposing complexity in a fail-safe manner.



The data is not accessed directly. It is accessed through the exposed functions. Hiding the internalsÂ of the objectÂ protects its integrityÂ by preventing users from setting the internal data of the component into an invalid or inconsistent state.

Encapsulation isÂ not just about hiding complexity but conversely exposing complexity in a fail-safe manner.

Inheritance

Inheritance is a powerful feature of Object oriented programming languages.Inheritance helps in organizing classes into a hierarchy and enabling these classes to inherit attributes and behavior from classes above in the hierarchy.
Inheritance describes an “IS A relationship. This is how we talk in the real world. Example. A Parrot is a bird. US Dollar is a type of currency. But the phrase, Bank is a bank account is not correct. This relation is obvious when you try to describe some entity in the given business/problem statement.

With inheritance, you can define a common implementation/behavior and later for specialized classes you can alter or change it to something specialized.
Inheritance does not work backward. The parent won’t have properties of the derived class.

Inheritance is a mechanism for code reuse and can help in reducing duplication of code.

Do not force inheritance. You would just be writing unnecessary code. It is important to note that while trying to model the requirement don’t go adding multiple levels of inheritance. This is not needed. You need to try to identify common attributes and behavior in the entities you modeled and based on that you can go ahead re-factoring the code defining a suitable parent class. Common implementation can then be moved to this class.

Polymorphism

Polymorphism is the concept that there can be many different implementations of an executable unit and the difference happen all behind the scene without the caller awareness.
PolymorphismÂ allows computer systems to be extended with new specialized objects being created while allowing current part of the system to interact with a new object without concern for specific properties of the new objects.

For example, if you needed to write a message on a piece of paper, you could use a pen, pencil, marker or even a crayon. You only require that the item you use can fit in your hand and can make a mark when pressed against the paper. So the action of writing would help you make a mark on the paper and what marking or writing instrument to use is a matter of decision. Another example is a plane and space shuttle both can be termed as Flying objects. But the way both fly are different I.e. there is a difference in implementation. But from a viewer's perspective both the objects can fly.

Inheritance is one means of achieving polymorphism where behavior defined in the inherited class can be overridden by writing a custom implementation of the method. This is called method overriding also known as Run Time polymorphism.

```
public class WritingInstrument{

    public void Write(){
        // Makes visible mark
    }
}


public class Pen : WritingInstrument{

    public void Write(){
        // Makes visible mark with ink
    }
}
```

There is also one more form of polymorphism called method overloading where inheritance does not come into the picture. The method name is same but the arguments in the method differ.

```
public class WritingInstrument{

    public void Write(){
        // Makes visible mark
    }

    public void Write(int fontSize){
        // Makes visible mark of given font
        // size
    }

}
```

References:

- https://stackoverflow.com/questions/16014290/simple-way-to-understand-encapsulation-and-abstraction
- http://www.adobe.com/devnet/actionscript/learning/oop-concepts/polymorphism-and-interfaces.html
- https://stackoverflow.com/questions/3322318/explain-polymorphism?rq=1

Credit:

Data from: https://dev.to/charanrajgolla/beginners-guide---object-oriented-programming

Icons from: http://www.freepik.com/