Absolute Beginner's Guide to Object-Oriented Programming

Have you started to learn to code, if your answer is yes then the most puzzling question for you probably will be *what exactly Object-Oriented Programming is* also known as, OOP?

It's no denying fact that the technology industry has a fascination for terms. Every now and then you will find a new term (for example 3D Touch and AR) being trending and everybody is adding their thoughts about it- whether they know it or not.

Now, because you are here I can safely assume that you must have heard or was taught that object oriented programming is a better way to code but without knowing the term itself in detail could land you into serious problems later on.

Let's begin our guide without wasting a single second now! 😛

Although it's recommended to stick with the web version, if you want a pdf copy of this click here.

What will you learn?

Before beginning to read any guide or tutorial you must know what you will have learned after you have finished reading.

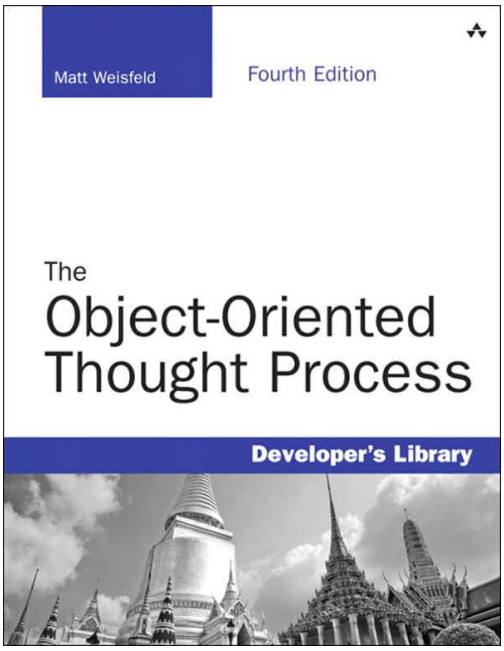
As the title suggests the article will be teaching you in's and out of object oriented programming. This guide will be the only thing you need to understand OOP. Because everyone will be of different programming backgrounds I will explain every concept with examples in real life and you

will never stumble ever again!

It doesn't matter what level of coder you are appropriate knowledge of these terms will help you a lot in the long run.

If you want you can bookmark this article for future reference.

Recommendation



If you are really interested in making your career as a programmer you must go through this book: The Object-Oriented Thought Process(US visitors) |The Object-Oriented Thought Process (Indian visitors) | The Object-Oriented Thou
Process (for non-US/ non-Indian visitors)

Introduction

(*You must read this whole, to understand why OOP was invented and to also know some basic programming rules***)**

In the very beginning of their inventions, computers were programmed in

binary language i.e., 0's and 1's. With the advent of cheaper and larger storage devices programming languages began to take shape and instead of now instructing the computer in bits, coders could write their code in English.

These programming languages were relatively easy to code in but as capacity and capabilities of computers continued to evolve the need for more complex programs began. It was this stage that the languages started showing limitations:

- There was no way a piece of code could be reused. If you had to insert same things in a program, you had to switch to duplication.
- The control of program was transferred via goto statement. This was very dangerous as several goto statements in the program made understanding the flow of program incredibly problematic.
- All variables (if you don't know a variable is, remember it is actually a named memory location) in the program were global. Meaning, once a variable has been defined any function will have access to it.

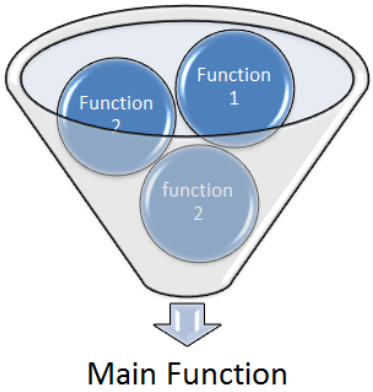
To overcome these difficulties structures programming languages were invented.

According to Wikipedia, the first structured programming language appeared in early 1970's. Structured programming language introduced Functions. Large programs could now be divided into smaller units called function thus making coding overall a simpler job. Functions solved three major problems of the basic programming languages.

- **Reusability** – With the invention of functions, whenever a part of the code was required again the function would be 'called' and thus avoiding duplication.
- **Global Variables** – Each function could have its own variable (that would be called local variable) and the scope (part of a program where a

variable is accessible) would be the within the function only.
- **Eradication of goto** – New control instructions were defined that eliminated the need to use goto statements

It's very important to understand how these functions worked. There would be several functions in the program so for the computer it will be difficult as to what function should it execute first and thus came the 'main' function. In any case, (whether there are other functions or not) the main function is mandatory. The main function will be the only function that executes and all other functions would from the main function in accordance with their order.



By separating different processes in different functions, It becomes easier to maintain and develop complex programs.

This type of programming paradigm dominated for more than two decades from1970's to 1990's. As the evolution of something is continuous, demand for richer programs began and complexity began to show up in structured programming language.

The limitations offered by structured programming languages can be enlisted as:

- Solving a problem was so unlike as you would do in real life and does

require a greater degree of mental effort.
- Maintenance of huge programs was still very difficult.
- A contrivance to reuse a piece of code was still not perfect.

That's when Object Oriented Programming language began to take shape

Object-Oriented Programming language

OOP focuses on real life. How do to solve a problem in real life? We focus on the object and the characteristics it possesses defines the procedure as to how would we solve the problem. Let's take an example.

How do you make a cup of coffee? First, you would think about the object (here coffee) and its behavior. Then you would select tools (coffee machine) and finally proceed with procedures. This is how you solve a problem in real life.

Everything is object oriented. The programmers are also real people and hence if the approach to solving problem also relates to the real world, solving of a program would be a much easier task.

The most fundamental difference between these two fundamental paradigms namely OOP and structure oriented programming is the design they are developed upon. OOP programs focuses more on objects whereas Structured programs focuses on process. What OOP basically does is that it ties data and functions (procedures in real life) into a single unit.

Now that we know what OOP actually is, how do we define it?

According to Margaret Rouse from Tech Target,"Object Oriented Programming is a programming language model organized around objects rather than actions and data rather than logic."

One of the essential features of OOP is data abstraction. Let's take an example. How do you drive a car? You will rotate the steering wheel and change gears accordingly. Do you need to know how is fuel inside the engine making the wheels move? Do you need to know how is the rotatory motion of steering wheels propels the cars? The answer is no. OOP does exactly that. So, when an end user uses a calculator app he or she needn't know how is the code implemented in PC.

Verdict and limitations of OOP

So, what did you learn?

Remember this when you solve problems (real life or while coding) rather than thinking on how to divide the problem into functions try to divide it in terms of objects – this will drastically improve productivity and this was the whole idea behind OOP.

Nothing is perfect. OOP also comes with its own set of limitations. Sumita Arora clearly marks them out as:

- Designing of OOP programs is very tricky.
- You need to have a proper plan before beginning to code.
- The programmer needs to think in terms of object and apply brains to correlate.

OOP has its own vocabulary that you will need to understand to continue to code. First, let us enlist them and later we will try to explain them.

They are:

- Objects
- Classes
- Inheritance

- Polymorphism
- Containership
- Reusability

Tada! First, let's target objects!

Objects

As discussed earlier, OOP divides the programs in terms of objects rather than in procedures thus making manipulation easy. Basically, an object is anything that has **unique** characteristics and nature. For example Mac Book Pro or Taylor Swift.

Classes

It's said that an object is an instance of a class. A class is a collection of objects that share similar characteristics. For example, both Surface book and Mack book Pro are objects as both have unique characteristics but there is a class classed "Laptop" in which both belongs as both share some common characteristics.

Inheritance

OOP was built in such a way as to render greater reusability options. Inheritance allows you to make another class by inheriting (like children inherit from parents) other class (called the parent or base class). Examples are the best way to understand things.

Let us take a class called Samsung Galaxy. We would want to create two other classes Samsung Galaxy S7 and Samsung Galaxy S5. Both of these classes would inherit from class Samsung Galaxy and in addition to it they will also possess some unique features. Like both phones would run on Android and possess a decent camera, but in additional to properties from class Samsung

Galaxy, they will also have unique properties like Galaxy S7 will have a low light optimized camera and Galaxy S5 will have a eye scroller – notice these two properties are what S7 and S5 do not have in common. Samsung Galaxy would be called the base (parent) class whereas Galaxy S7 and Galaxy S5 would be called derived(child) class.

OOP's inheritance works in similar fashion.

Polymorphism

By definition, it is the ability to take more than one form. Did you understand it? Let's extend our example. Both of these phones would be put in Amazon's Samsung section so, how do we locate them? We would use filters and select the year in which they were released in. We would easily find S7 in 2016 and S5 in 2014.

Did you notice what did we do here? We used a common filter (function in case of programs) and by the use of that we got different results. (Same filter upon different conditions produce different results, thus the filter could be said to be exhibiting polymorphism.)

Containership

I will start away with a real world example. In a typical online market like Amazon you would find different items divided into categories like Clothing and books and in each category Clothing are various sub-categories called Men's shirts or Women's shoes. Men's shirts and women shoes can be said to be *contained in* Clothing. This is called containership. You will get to know

more of this as you move along with more advanced codes.

Reusability

Reusability is a core feature of OOP. Once a program is developed and tested it would be distributed to various developers. OOP programs are designed in such a way to facilitate easy sharing rendering faster and better apps.

Proper approach to code

Because coding Object-oriented programs are a mammoth task. The programs are tricky therefore it's very important that we make a proper plan.

Follow these:

1. Identify the supreme actors – important players.
2. Seek for the relation that exists between them (actors).
3. Recognize actions carried out by players.

Let us take an example.

(Originally example by Yahavant Kanetkar in his book Let Us C++)

Consider a college. The college will have more than one department with each department offering one or more than one program, like a degree for B. Sc. For every single program there will be several courses. Students will take admission for a particular course and each course will be taught by a professor who can teach multiple courses.

Can you now identify the actors? The easiest one to pick out is that a student is an actor. (Find other actors on your own 😛 and comment down)

Now it's time to seek for the relation between actors by looking at their actions.

Students *purchase* form to take admission in a particular program. Students will submit application form and fees. The college will examine application forms and would either approve or reject candidature. Once approved, the respective department will admit the student and would prepare a time table for each program offered.

Professors would conduct classes according to the timetable and the department will prepare an examination schedule would further conduct an examination as per the schedule. Professors would evaluate answer sheets and accordingly allots grades. Department will collect the grades and then announce the result.

So, what exactly did we do? We focused on objects (actors – student, department, etc.) then let their characteristics define the method to solve our problem (here, college administration). This example clearly indicates how OOP is related to real world.

Post Script

> *"Practice makes a man perfect"*

I hope that your confusions regarding object-oriented programming must have been cleared. If you have any doubts of any kind do not forget to comment down below. Also, do not forget to share this in your favorite sites and subscribe to our newsletter for regular updates.

Sources and Citations:

- Object-oriented programming – Wikipedia
- Lesson: Object-Oriented Programming Concepts
- Quora