Kennesaw State University
Department of Information Technology
IT5413 Software Design & Development
Fall 2017

Object-Oriented Programming

By Karis Kim

Table of Contents

1. Introduction

Object-Oriented Programming (OOP) is pervasive in software design and implementation, and Object-Oriented Technology is also extending its reaches in database environments. So, what is OOP, how did OOP evolve into the dominant paradigm, how is it different from previous procedural languages and why is it better in terms of cost and software quality? This paper will summarize the answers to those 4 questions.

2. What is Object-Oriented Programming?

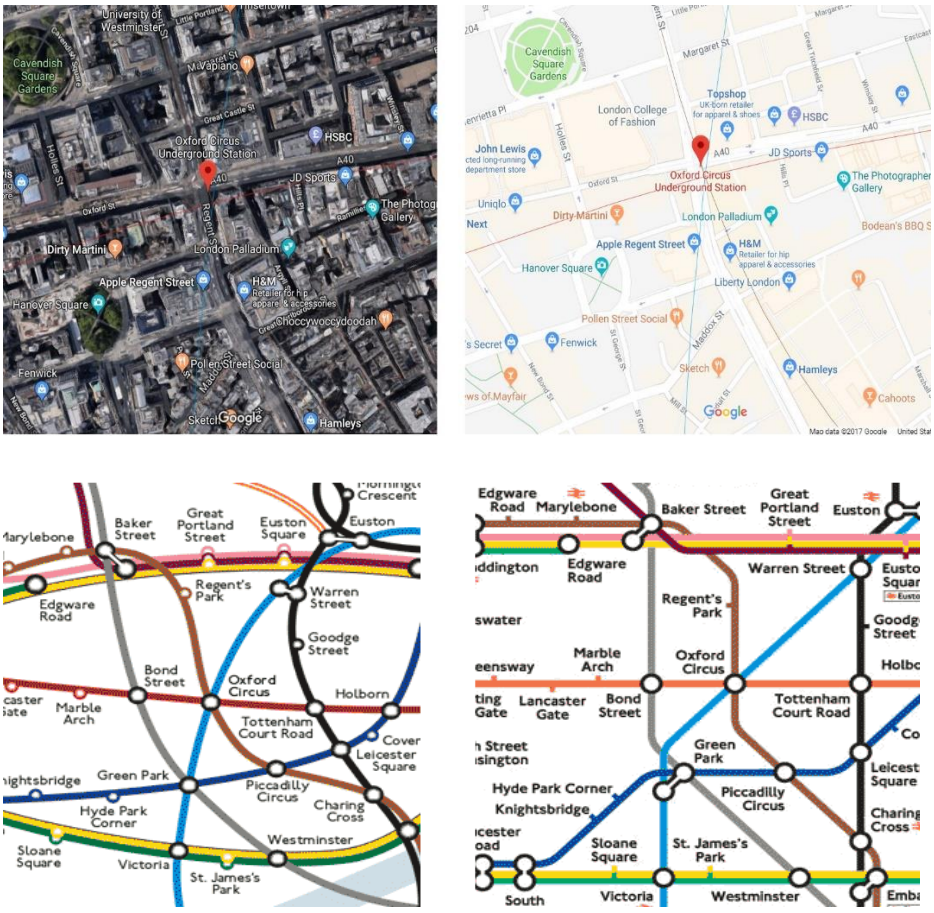
OOP is a key paradigm in today's programming technology. In the OOP approach, programs are grouped together in basic units called objects with their own attributes and functions, so OOP codes are easier to understand, correct and modify. OOP languages like Java have classes, which some refer to as a "factory" that contains "blueprints" to mass produce objects. In a class, variables of the object are declared with the operations (methods) that the object will perform and a set of function definitions that implements each of the operations ("A Brief History of Object-Oriented Programming," n.d.). There are four main concepts that define OOP: abstraction, encapsulation, inheritance, and polymorphism.

2.1 Abstraction

One of four key characteristic concepts in OOP, abstraction is a process to simplify the design of complex programs by representing objects with their essential features without all the background details. Abstractions are described as "tools for describing the common structure of similar phenomena" (Nygaard, 1986). Abstractions allow the programmer to encapsulate and hide all but the relevant data about an object, which reduces complexity and increases efficiency (Rouse, 2014). As summarized in a definition of abstraction by Techopedia, abstraction and the other three key OOP concepts, encapsulation, inheritance and polymorphism, are very closely

related, and "[a]bstraction defines an object in terms of its properties, functionality, and interface (means of communicating with other objects)," as "[t]he state of the object is encapsulated while the detailed data structures associated with the object are kept behind the scenes." Figure 1 illustrates the concept of abstraction with Google Map satellite image of Oxford Circus Station in London and three maps of the same location; the three maps illustrate abstractions in which only relevant essential features are represented without all the background details.

Figure 1.



2.2 Encapsulation

Encapsulation is one of the main concepts of OOP, where data attributes and methods are wrapped (or encapsulated) as a single program unit and the implementation details are hidden

inside the class. By declaring the variables of a class private, data hiding is accomplished and can only be accessed through the use of setter and getter methods. The hiding of implementation details from the rest of the system is useful in preventing unauthorized access in teams of many developers or users, and is a key aspect of OOP. Yaiser's article about encapsulation best summarizes the concept as:

> *Put simply, encapsulation is about hiding complexity. The first reason is to provide a simplified and understandable way to use your object without the need to understand the complexity inside. In the same way, hiding the complex functionality of your object from the user allows anyone to use it and to find ways to reuse it in the future regardless of their knowledge of the internal workings. The second reason for hiding complexity is to manage change… object can change very significantly internally while the remainder of your application can stay exactly the same (2011).*

2.3 Inheritance

Inheritance is a fundamental concept of OOP language that enables software reuse and promotes cost efficiency and software quality improvement. When creating a new class, you can "inherit" an existing class' members and add or modify capabilities from the existing class, rather than having to declare completely new members. The existing class is known as the superclass and the new class is known as the subclass, in which the inheritance is represented by an "is-a" relationship between the subclass and its superclass. The subclass "is a" member of a more general superclass; for example, a truck (subclass) *is a* vehicle (superclass). Inheritance allows you to create multiple subclasses like Truck, Sedan, SUV, and Bus, by reusing the properties from the superclass Vehicle that would be shared by all the subclasses, without having to recreate them for each subclass. Inheritance is possible in OOP languages because a subclass

object is an object of its superclass. In procedural programming, without the benefit of OOP, inheritance would not be possible.

2.4 Polymorphism

Polymorphism is a characteristic concept of OOP languages that is closely related with inheritance hierarchies. According to Deitel and Deitel, polymorphism "enables you to "program in the general" rather than "program in the specific" …to write programs that process objects that share the same superclass (either directly or indirectly) as if they're all objects of the superclass" (2012, p.395). With polymorphism, "subclasses of a class can define their own unique behaviors and yet share some of the same functionality of the parent class" ("Polymorphism," n.d.). So, polymorphism enables a method of the superclass to be implemented in different ways in the subclass objects. Singh (2017) illustrated this concept using a superclass Animal with a method Sound( ): For example, the Animal class method Sound( ) may be implemented in subclass Cat as meow, subclass Cow as moo, and subclass Bird as chirp by using method override. In addition to method overriding, method overloading is another critical concept of polymorphism. Overloading is achieved when the same method can produce various types of data.

3. How did OOP develop?

3.1 History of OOP

In the 1960s, the concept of OOP debuted in the world with a programming language for simulations known as Simula. Simula was created in Norway by Dahl, Myhrhaug and Nygaard. According to Weisfeld (2005), "The introduction of Simula-67 brought with it the first true programming object, classes, and a form of inheritance; therefore, Simula is an important

milestone in any discussion on O-O programming languages… Simula had a novel way of presenting the object, so that each object has its own behavior and data." Simula was a language for simulations and object-oriented programming was better suited than procedural programming to represent real life objects.

In the 1970s, Smalltalk, conceived by Alan Kay, was introduced to the world by Xerox company's Palo Alto Research Center (PARC) as what many call the first truly or pure Object-Oriented language and "gave O-O development a certain amount of legitimacy in the marketplace" (Weisfeld, 2005). Weisfeld states that in the programming environment of Smalltalk, "everything is really an object that enforces the O-O paradigm. It is virtually impossible to write a program in Smalltalk that is not O-O" (2005).

In the 1980s, OOP gained widespread acceptance in the mainstream market through C++. Bjarne Stroustrup at Bell Labs "integrated object-oriented programming into the C language" and created C++ ("A Brief History of Object-Oriented Programming," n.d.). Weisfeld and others in the field consider C++ to be the most important OOP language because it was the first to be widely used in the marketplace.

Then in the 1990s, James Gosling at Sun Microsystems worked with features of C++ to create what eventually became known as Java. With the explosive growth of internet and the growth of Java as a language for internet applications, OOP language also gained worldwide popularity. Many other OOP languages have been developed to join the market competition, and developers now can choose from many OOP languages, such as C#.NET, VB.NET, PHP, Python, Ruby, etc. A quick google search will yield that almost all programming languages now have adopted or support OOP concepts, even if the language started out as a procedural language like COBOL and LISP.

4. What makes OOP different?

4.1 Procedural Programming vs. OOP

According to Wikipedia, Procedural Programming languages appeared in the 1960s with Fortran, Algol, COBOL, and BASIC, then in the 1970s with Pascal and C, followed by Ada in 1980. In Procedural Programming, also known as Imperative Programming, programs are structured in a top-down, step-by-step manner comprised of procedures/functions, routines, and subroutines. With the Procedural Programming approach, a program is "divided into functions that perform a specific task. Data is global and all the functions can access the global data ("Evolution of OOP," n.d.). The focus of Procedural Programming is on the order of or procedure of actions to perform specific functions on/with data attributes, while the focus of Object-Oriented Programming is on organizing specific functions into objects that groups its own data attributes with functions. In Procedural Programming, since the data is global and all functions can access it, it is less secure than in OOP where data can be encapsulated and hidden from unauthorized users. Unlike OOP, Procedural Programming does not have overloading. In Procedural Programming, editing a program would require the developer to go through every line in the entire program to edit every single line of code that is related to/with the change. As Eliason articulated, "as more and more changes may be needed to the code, it becomes increasingly difficult to locate and edit all related elements in the program" (Eliason, 2017). In OOP, modularization allows the developer to go to the object class to be changed and find all related variables and methods grouped together for the change. So, the main differences between Procedural Programming and OOP approaches involve the focus and structure, data visibility/access, reusability, and maintenance.

5. Why does OOP reduce cost and improve software quality?

5.1 Cost Reduction

OOP designs have objects that are modularized into more manageable classes. With inheritance, multiple subclasses can be created by reusing properties from a superclass, and with polymorphism, a single action can be performed in multiple different ways with overriding and overloading ("Polymorphism in Java," n.d.). So, designs can be reused and recycled, which saves time and resources, thereby promoting cost reduction in the development of code. Since OOP classes are divided up into easily understandable bundles, maintenance, particularly in large program environments, can be much faster, thereby promoting cost reduction in the maintenance of code, in addition to development.

5.2 Better Software Quality

Encapsulation enables data hiding, which can prevent unauthorized access and secure the information, leading to better software quality. Hirshfield and Ege (1996) wrote that encapsulating data and actions together in an object promotes better design and understanding of a program. Inheritance, basing new classes on existing classes that have already been tested and debugged, can lead to better software quality also. With reuse and recycle ability in OOP, the cost and time saved in development can be allocated to testing and verification of software, which promotes higher software quality (The Saylor Foundation, n.d.). Moreover, OOP objects are "self-contained, and each bit of functionality does its own thing while leaving the other bits alone… this modality allows an IT team to work on multiple objects simultaneously while minimizing the chance that one person might duplicate someone else's functionality" (Half, 2017). That modularity means testing, trouble-shooting, maintenance and updating is also easier, and that in turn could lead to better software quality as well.

6. Concluding Remarks about OOP

Surprisingly, OOP has been around as long as Procedural Programming, but the OOP momentum for global popularity appears to have been fueled by the growth of Java paired with the massive expansion of the internet. The difference between OOP and Procedural Programming approaches increases in significance as the size of the program becomes larger/longer. OOP languages like Java can produce codes that are Procedural, and if the said code is simple and short enough, the lack of OOP would scarcely be missed. My personal experience confirms the common assertion that there is a steep learning curve with OOP, and since OOP is most suitable and beneficial for large scale programs rather than smaller programs, I would disagree with the blanket statements claiming that OOP languages are simply better. Research indicates that Procedural Programming is still widely used, and it would be safe to conclude this research by stating that even with all the "bells and whistles" and benefits of OOP, the best programming approach is the one that best meets the needs of a particular project at hand, and clients/developers would be better served in thoroughly researching all options and programming paradigms.

7. References

A Brief History of Object-Oriented Programming. (n.d.). Retrieved December 01, 2017, from http://web.eecs.utk.edu/~huangj/CS302S04/notes/oo-intro.html

Abstraction (software engineering). (2017, November 01). Retrieved December 07, 2017, from https://en.wikipedia.org/wiki/Abstraction_(software_engineering)#Abstraction_in_object_oriented_programming

Deitel, P. J., & Deitel, H. M. (2012). *Java how to program* (9th ed.). Boston: Pearson.

Eliason, K. (2017, August 17). Difference Between Object Oriented and Procedural Programming. Retrieved December 01, 2017, from https://neonbrand.com/website-design/procedural-programming-vs-object-oriented-programming-a-review/

Encapsulation (networking). (2017, September 14). Retrieved December 07, 2017, from https://en.wikipedia.org/wiki/Encapsulation_(networking)

Evolution of OOP - Step by Step Programming. (n.d.). Retrieved December 01, 2017, from https://sites.google.com/site/simplestjava/evolution-of-oop

Half, R. (2017, July 13). 4 Advantages of Object-Oriented Programming. Retrieved December 07, 2017, from https://www.roberthalf.com/blog/salaries-and-skills/4-advantages-of-object-oriented-programming

Hirshfield, S., & Ege, R. (1996, March). Object-oriented programming. *ACM Computing Surveys (CSUR), 28(1),* 253-255. Retrieved December 07, 2017, from https://dl.acm.org/citation.cfm?id=234415

Nygaard, K. (1986, October). Basic Concepts in Object-Oriented Programming. *SIGPLAN Notices, 21(10),* 128-132. Retrieved December 7, 2017, from https://pdfs.semanticscholar.org/ba2b/e42b631dda8c51d9c90e26755353ce53bab4.pdf

Object-oriented programming. (2017, November 26). Retrieved November 26, 2017, from https://en.wikipedia.org/wiki/Object-oriented_programming

Polymorphism. (n.d.). Retrieved November 29, 2017, from https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html

Polymorphism in Java - javatpoint. (n.d.). Retrieved December 07, 2017, from https://www.javatpoint.com/runtime-polymorphism-in-java

Procedural programming. (2017, November 26). Retrieved December 01, 2017, from https://en.wikipedia.org/wiki/Procedural_programming

Rouse, M. (2014, June). What is abstraction? - Definition from WhatIs.com. Retrieved December 07, 2017, from http://whatis.techtarget.com/definition/abstraction

Singh, C. (2017, September 12). Polymorphism in Java with example. Retrieved November 29, 2017, from https://beginnersbook.com/2013/03/polymorphism-in-java/

Stroustrup, B. (Modified 2016, December 21). The C Programming Language. Retrieved December 01, 2017, from http://www.stroustrup.com/C .html

Stroustrup, B. (1988, May). What is Object-Oriented Programming? *IEEE Software, 5(3),* 10-20. Retrieved December 6, 2017, from http://ieeexplore.ieee.org/abstract/document/2020/

The Saylor Foundation. (n.d.). Advantages and Disadvantages of Object-Oriented Programming (OOP). Retrieved December 7, 2017, from https://www.saylor.org/site/wp-content/uploads/2013/02/CS101-2.1.2-AdvantagesDisadvantagesOfOOP-FINAL.pdf

Weisfeld, M. (2005, March 31). The Evolution of Object-Oriented Languages. Retrieved November 29, 2017, from https://www.developer.com/java/other/article.php/3493761/The-Evolution-of-Object-Oriented-Languages.htm

What is Abstraction? - Definition from Techopedia. (n.d.). Retrieved December 07, 2017, from https://www.techopedia.com/definition/3736/abstraction

What is a Procedural Language? - Definition from Techopedia. (n.d.). Retrieved December 07, 2017, from https://www.techopedia.com/definition/8982/procedural-language

Yaiser, M. (2011, October 31). Object-oriented programming concepts: Encapsulation. Retrieved December 07, 2017, from http://www.adobe.com/devnet/actionscript/learning/oop-concepts/encapsulation.html