Executive Summary- Object Oriented Programming

Object oriented programming is used for creating systems because of the benefits that come from using an OOP language such as Java. The attractive features of an object oriented language mostly boil down to three things, which will be elaborated upon: inheritance, encapsulation, and polymorphism. As with anything, object oriented programming has its setbacks, and these are related, in part, to what makes OOP attractive to use in the first place.

Inheritance

To begin with, inheritance, is a trait that must be understood in order to see why using object oriented programming is so attractive. Object oriented programming may be class based or prototype based. Java, is a class based language. In OOP, classes act as a skeleton, or blueprint that may be used to create objects with certain attributes or features. To take an example, we might create a class called employee. Employees come with some commonalities, no matter the type, therefore we may say an employee will have an ID number, department and some kind of salary. Employee then may be divided into two sub-classes called salary employee and hourly employee, both of which have the same traits as the class employee as well as some differences (these may be things that need to be added to the sub-class, or adapted for the sub-class).

This is where object oriented programming is advantageous, because it allows for the sub-classes we want to create to inherit from the super-class, hence allowing programmers some reprieve from having to code and create classes, methods, objects etc. that already exist in some form and need only be adapted to the new class. With class hierarchy, classes that have a direct super-class as well as classes that have an indirect super-class may exists. While this is a huge benefit of object oriented programming, this can also be problematic for OOP languages that allow multiple inheritance.[1] It is important to bear in mind that inheritance allows code to be reusable (inheritance allows us to "recycle"code) as well as allows for more maintainability.

Encapsulation

Encapsulation is the second trait that makes object oriented programming advantageous to use when building software systems. Encapsulation is defined as the encasing of attributes and methods.[2] This makes the notion of "information hiding" and data hiding possible. In OOP objects may communicate with each other but are not allowed to know how how other objects are implemented.[3] For example, in Java, a way encapsulation is performed is by making a variable private, but having the getter and setter methods of that variable be public thus allowing that variable to hide within the methods. The getter and setter methods here allow for the value of the private variable to be read and updated by other classes .[4] In short, encapsulation allows for a programmer to control the access to variable by other classes by hiding these attributes in methods and allowing these attributes to be updated and read if the methods encapsulating them are public. In a way, it makes the code more secure, because it allows the programmer freedom in determining who has access to certain attributes and who, if anyone, can change the data on the attributes,(an example being balance in a bank account. No person should have illegal direct access to that variable and it should be protected from being changed).

Polymorphism

Polymorphism is the reason we "can program in the general rather than in the specific as long as they(sub-classes) share the same super-class."[5] Due to polymorphism, a programmer may add to a system without having to go back and change the majority of the program. With polymorphism, a subclass object is an object of the super-class itself therefore making the super-class extensible, which is helpful for

1 Timothy Boronczyk, 2009, What's Wrong with OOP, accessed 21 November 2018, https://zaemis.blogspot.com/2009/06/whats-wrong-with-oop.html
2 Paul Deitel and Harvey Deitel, *Java How to Program (Early Objects)* (10th Edition Pearson, 2016)
3 Paul Deitel and Harvey Deitel, *Java How to Program (Early Objects)* (10th Edition Pearson, 2016)
4 Thorben Jansen, 2017, *OOP concept for Beginners: What is Encapsulation,* accessed 21 November, 2018, https://stackify.com/oop-concept-for-beginners-what-is-encapsulation/
5 Paul Deitel and Harvey Deitel, *Java How to Program (Early Objects)* (10th Edition Pearson, 2016)

implementing “layered software systems”. [6] It also allows for one interface to be used in multiple objects, regardless of their super-class, as long as those objects implement that interface. Another way to look at interfaces is as a tool that declares what methods need to be implemented, but allows the programmer the freedom to implement the methods as they may see fit.

Abstract classes may be used as well. In Java How to Program (Early Objects), an Employee example may be found in which the abstract super-class chooses to leave the method “earnings” abstract. When this abstract method is later implemented in concrete sub-classes, a programmer may specify how that method will be implemented-that is, “earnings” abstract method is tailored to the type of employee the “earnings” method is implemented for. Due to these features of polymorphism, a programmer can reduce the amount of work he or she may need to do for writing a program. A programmer may choose to add sub-classes to the hierarchy, they may override methods, overwrite existing methods, etc. all without having to rewrite or code something that has already been coded and merely needs to be adapted to fit the needs of a certain subclass.

Problems with OOP

As we have seen, the benefits of object oriented programming are clear. Inheritance, encapsulation and polymorphism make coding and programming easier, more secure, makes code recyclable and reusable, as well as extensible. As with most things, however, object oriented programming has its problems and setbacks. Some of the problems with OOP include the complexity of programs or systems, the speed, and the effort that goes into coding or designing an OOP program.

Complexity in OOP seems to be an issue with programmers. Some may argue that more lines of code are needed to accomplish the same thing that might be accomplished using less lines in a non-OOP language. Arguments have been made that transparency may be difficult to achieve in systems created with OOP due how large the systems become. At some point, due to inheritance and polymorphism, it becomes difficult to know how all the sub-classes and super-classes relate to one another. Some OOP languages also seem to allow multiple inheritance, which can lead to even more complexity in the system. [7] To drive the point of complexity home, Joe Armstrong is quoted as having said, “You wanted a banana but what you got was a gorilla holding a banana and the entire jungle.”

As far as speed is concerned, this is a setback that was more of an issue in the past than it is now, but as systems do get larger and more complex , it is an issue that should nonetheless be addressed. A computer has only so much memory and space to process information, and in the past the space and memory in a computer was less than we are afforded now.[8]

Planning is essential when building any system, and when it comes to a system where OOP is used, this holds even more true. Programmers need to plan meticulously before writing code. The hierarchy between classes need to be clearly drawn out especially because the systems built using OOP can be extremely large and complex. Due to the size of the systems, programmers also need to spend more time writing the actual programs for the system as well as spend more time maintaining said programs. All in all it can become quite tricky to plan, build and then maintain OOP systems.

Conclusion

Object oriented programming is useful for building reusable, secure and adaptable programs due to inheritance, encapsulation and polymorphism. As with most things, these very advantages offered by OOP languages can become a double edged sword and can lead to complexity, programs that take longer to compile and run, as well as longer to plan, write and maintain. Some of these problems are caused by sloppy programming but these continue to be issues all the same. In the end OOP serves a certain purpose and is an ideal method to use when solving certain problems, while other kinds of programming (such as functional programming) may be more useful and optimal in other circumstances.

---

6 Paul Deitel and Harvey Deitel, *Java How to Program (Early Objects)* (10th Edition Pearson, 2016)

7 Charles Scalfani, 2016, Goodbye, Object Oriented Programming, 21 November, 2018, https://medium.com/@cscalfani/goodbye-object-oriented-programming-a59cda4c0e53

8 Jeffrey L. Popyack, 2012, CS 164: Fall 2012- Introduction to Computer Science, 21 November 2018, https://www.cs.drexel.edu/~introcs/Fa12/notes/06.1_OOP/Disadvantages.html?CurrentSlide=2