Modern Object Oriented Programming

To say that Object Oriented Programming is a modern day paradigm may be a misnomer, at least the "modern," part. Although 'Object-Oriented,' programming and design is still the most traditional software development approach for engineering advanced software systems. It has not caught on with modern development until the 1980's. However, the idea of objects and the classes from which objects are derived from may be traced to the ancient Greek schools of Plato and Aristotle.

In The Republic, Plato describes 'Forms.' Mathematical/Geometrical structures that exist in an ethereal state. We can liken these structures to blueprints. Blueprints for real world objects. For example, say an individual is building a cart. The individual then decides that the vehicle would be more efficient to move with a wheel. This wheel does not yet exist; it exists in the person's mind. As a concept, if you will. Plato would argue that the 'Form' of the wheel, i.e. the circle of which the wheel is derived from does in fact exist. Once again the idea exists. Let's liken the idea as the class in the context of Object Oriented Programming. Plato's theory of forms is out of scope for this paper. I hope you get a general idea as to how it pertains to our discussion on Object Oriented Design. Now let's say that the circle "class," is in the mind of the individual. To create a functional material object in the real world, i.e., something that is practical. The designer must create the object from raw material, let's say using wood of which he/she harvested from a tree. This wooden wheel will now be a physical object just as the tree was once. We can say one of the properties of the wheel is that it is wooden, that it is has a color of light brown, and that it is rigid. The individual could have also used a metal rim and rubber for constructing the wheel. If the latter occurred, then the wheel will possess certain properties of which the wooden wheel would not. These properties would contribute to the functionality of the cart, i.e., in the performance, effectiveness and efficiency of the said cart.

Let's see how we can use the metaphors above regarding Object Oriented programming. It is very natural for human beings to think in objects. According to studies conducted by Jean Piaget, little children begin to distinguish reality as consisting of objects. [1] This is a phase known as "Concrete Operational Stage." The ancient Greek philosopher, Aristotle of Stagira postulates the "class." He uses the concept of the class for organizing the many specimens of organisms he collected. This organization structure in the terms of class lends itself to one of the tenants of Object Oriented Programming, i.e. Inheritance. Aristotle, took note that certain animals and plant life have structures that are similar in form and function, e.g. the hoofed animals he classified in the class belonging to the hoofed animal vs. the animals of which had paws.

In modern Object Oriented Programming, it is apt to use inheritance. This allows for less code use and duplication of code. By defining a class with properties(form), an object may have methods(function) that are shared across the entire class From this class aka the superclass other classes may inherit these properties and functions. That is once an object is created from such a class; it would inherit for example the form of a hoofed foot with the function of galloping.

Another tenant of object oriented programming is Polymorphism from the Greek words, many and shape. Aristotle may have documented the first classification of life on Earth, (there are no surviving records from philosophers or natural scientists before him) he did not distinguish that life forms may change over time depending on changes in the environment. In modern times Biologists have noticed that certain aquatic animals such as whales, dolphins, and manatees use their "arms" as fins. So much so that their arms ARE invariably fins, however, the underlying structure i.e. the bones are very much closer to the arm of land-dwelling animals than they are to the bone structure of fish fins. A human being could also use their arms as a propulsion through water. This an excellent example of how an arm has polymorphic properties.

In Object Oriented Programming a method may also be polymorphic in nature. Take for example the + symbol. This operative may be used to add numerical values and also to concatenate two or more string values.

In the study of Biology an organism is encapsulated from the environment, this allows the organism to carry on homeostasis. We only have to look at the cell as the basic building block of all life on Earth. The cell as an object that is encapsulated. Encapsulation is another tenant of Object Oriented Programming. The environment in which a cell lives can be very harsh, therefore the cell has a cell membrane or cell wall to protect itself from elements in the environment. In multicellular organisms, cells are also differentiated in form and function. This differentiation allows their utility to the organism. The cell has a function that is peculiar to the role of which organ it belongs to in a multicellular organism, e.g. the human. For this to occur the cell is encapsulated with a membrane contains specific protein markers and protein gates of which "float" on the phospholipid bilayer. Not only does this allow the cell to be recognized by other cells, and the immune system, these protein gates allows the cells to be selectively permeable from elements within the extracellular space. We can liken this to 'data hiding' which is the primary purpose of encapsulation as it pertains to Object oriented programming. Just as a molecular signal from one organ may act on specific individual cells, e.g. the Thyroid stimulating hormone, methods of an object should not alter the behavior of other objects of which they are not privy to. In modern object-oriented programming, we use the access modifier to accomplish this. We can use the private keyword for encapsulating data accordingly.

Object Oriented Programming is one of the most popular paradigms of software engineering. Functional programming is another paradigm that is becoming popular. Overall the use of these models is in assisting human beings to solve real world problems. Software just as ideas are abstract. It also requires not only problem-solving skills but abstract thinking. It has become popular, almost a necessity in post-industrial societies for incoming individuals to the labor force to know/understand programming. It has become clear that individual members of post-industrial societies be proficient in their technical acumen concerning computer literacy. So much so that there is a push for programming to be taught in all secondary schools as a requirement. Today's leaders in the tech world such as Mark Zuckerberg and Bill Gates are the main proponents of these initiatives.

So far in this essay, I tried to make analogies of object-oriented programming to other subjects such as Philosophy and Biology. For me this important because I understand that many students are coming

from non-programming backgrounds in their undergraduate studies. Biology is taught at every secondary school level. However, programming is not. Does the biological science require some specialty of learning? Does programming need a particular type of learning and thinking, that perhaps is not suitable for all students? Then why is Biology appropriate for all students? Biology just as all sciences requires critical thinking and problem-solving, also understanding of abstract ideas like classes for the organization of life. However, Biology is not taught in a vacuum. Today's high school students learn mathematics along with other disciplines. I strongly believe that depending on their interests. The student will excel in the subject that piques these interest(s). I think for students to excel in programming they must have an emotional stake in the subject. Perhaps the best way to grow their interest is through pedology. Indeed, I believe that every individual has a talent and use to society. Programming is a unique subject, and I think something that is the accumulation and crystallization of all human pursuits. Programming is the combination of the arts and sciences. It borrows something from pure sciences such as mathematics and also borrows from the practical industries of the arts. If a teacher can create an interest to a student who's primary interests are in automobile mechanics, the teacher can show how the student how he may be able to design a better engine by programming a microchip that can enhance the motor's performance. Learning is best done through association. And today our society, civilization is a whole is software based. I agree with Bill Gates and Mark Zuckerberg's coding for everybody initiative. Programming has to be fun; it has to be emotionally rewarding to the brain. I have experienced this reward myself. It is not any different than having a eureka moment in physics or finishing a piece of music on a musical instrument. I also believe that exposing younger members of society to programming it will also help them think logically. Programming like mathematics is a form of thinking. Today I've noticed young kids programming by using simulation software such as Minecraft. Is programming for everyone, yes it is.