**Introduction**

This assignment is designed for students to practice technical communications with non-technical audience. Students are required to choose a relatively complex term within one’s profession and convey the meaning and usage of it through a parenthetical definition, a sentence definition and an expanded definition. At least one expansion strategy should be visual representation.

For this assignment, I have chosen the **term**polymorphism, one of the four important features of object oriented programming languages. Target **audience** can be divided to two clusters. For general readers, the following definitions intend to provide a general idea of how polymorphism can be applied in different programming contexts. For programming beginners, the introduction of the concept will deepen understanding of concepts in their abstract forms by linking them to real life scenarios.

**Parenthetical definition**

The most common use of polymorphism (the ability of an object to take on many forms) occurs when a parent class reference is used to refer to a child class object.

**Sentence definition**

In object-oriented programming, polymorphism refers to a programming language’s ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes. It is considered to be a requirement of any true object-oriented programming language.

**Expanded definition**

*History of polymorphism*

The term polymorphic has Greek roots and means roughly many forms. Poly = many, morphos = form. Morphos is related to the Greek god Morphus who could appear to sleeping individuals in any form he wished and hence was truly polymorphic.

*How does it work*

Since polymorphism in our context is a computer science terminology where things will revolve around other basic terms like “program” and “method”, we decided to briefly mention their meanings here. To execute a program is basically when you give your computer a series of instructions in order for it to complete certain task, for instance: check your emails, open a browser etc. A method is one of the smaller components of a program, a step towards the final completion of a task, for instance, in order to check my emails, a program needs to be able to connect [method 1] to a mail server, like Gmail, then list [method 2] all the mails it has, and view[method 3] certain mail I choose to read.

* a. static polymorphism

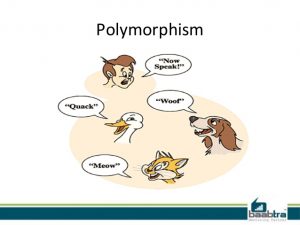
Referring to the first image above, we have a program about how our digestive systems work after consuming burgers. One of the obvious steps are to eat. We can choose to just have the burger or to make it a combo, but they are merely varied ways of enjoying one’s lunch, a polymorphic state of the eat method. Now we are able to observe how our body responds differently to various add-ons to the burger triggered by the separate eat method. This is called static polymorphism because customers are provided with menus before eating, knowing all the possible choices they can have with the burger.

[](http://engl301-arts.sites.olt.ubc.ca/files/2017/01/static-poly.jpg)

(static example)

* b. dynamic polymorphism

Referring to the second image above, we have a program for zoo managers to give a speaking test for all the mammals playing on the meadow. Instead of walking onto every single animal one by one and inviting them to speak, which can be quite time consuming and inefficient, the managers decided to give a shout to all on the spot. Assume all mammals have the speak method, but each makes a different sound. The polymorphic state of mammal ensures that all types of mammal, eg: dogs, cats, ducks, on the spot will conduct their actions of speak, while types of non-mammals will stay quiet. It is a dynamic process because animals come and go freely, and the managers have no idea how many are wandering on the meadow at the moment of testing. They also have no control over which mammal will make the first sound.

[](http://engl301-arts.sites.olt.ubc.ca/files/2017/01/dynamic-poly.jpg)

(dynamic example)

Works cited

http://www.webopedia.com/TERM/P/polymorphism.html

Ke, Ai, Jie Hou, Jian-xing Wang, and Bruce Eckel. *Thinking in Java*. Tai bei shi: Qi feng zi xun, 2002. Print.

Lippman, Stanley B., and Josée Lajoie. *C primer*. Reading, MA: Addison-Wesley, 1998. Print.