PROG1400 - Assignment 2

Object Oriented Vocabulary

Date: 3rd of April 2023

Name: Nathan L'Abbe

Table of Contents

[Polymorphism 4](#_Toc131507598)

[Inheritance 4](#_Toc131507599)

[Encapsulation 4](#_Toc131507600)

[instance of a class 4](#_Toc131507601)

[instance method 4](#_Toc131507602)

[instance variable 4](#_Toc131507603)

[class method 4](#_Toc131507604)

[class variable 4](#_Toc131507605)

[immutable 4](#_Toc131507606)

[class diagram 4](#_Toc131507607)

[static 4](#_Toc131507608)

[Abstract Classes 5](#_Toc131507609)

[Derived Classes 5](#_Toc131507610)

[scope 5](#_Toc131507611)

[subclass 5](#_Toc131507612)

[parent class 5](#_Toc131507613)

[base class 5](#_Toc131507614)

[method overloading 5](#_Toc131507615)

[array 5](#_Toc131507616)

[collections 5](#_Toc131507617)

[final 5](#_Toc131507618)

[interface 6](#_Toc131507619)

[enum 6](#_Toc131507620)

[class 6](#_Toc131507621)

[getter 6](#_Toc131507622)

[setter 6](#_Toc131507623)

[objects 6](#_Toc131507624)

[super 6](#_Toc131507625)

[ArrayList 6](#_Toc131507626)

[UML 6](#_Toc131507627)

[this (In the JAVA Programing Language) 6](#_Toc131507628)

[constructor 6](#_Toc131507629)

[References 7](#_Toc131507630)

# Polymorphism

Text Book Definition: “The quality or state of existing in or assuming different forms” (Merriam-Webster, 2020)

When it comes to Java, this is typically referring to the concept of having multiple of the same function or method that take different amounts of variables. For an example, take the following code.

public class Polymorphism {  
 public static void main(String[] args) {  
 int x = 3;  
 int y = 4;  
 int z = 5;  
 System.*out*.println(*sum*(x,y));  
 System.*out*.println(*sum*(x,y,z));  
 }  
  
 public static int sum(int firstNumber, int secondNumber)  
 {  
 int sum = firstNumber + secondNumber;  
 return sum;  
 }  
 public static int sum(int firstNumber, int second Number, int thirdNumber)  
 {  
 int sum = firstNumber + secondNumber + thirdNumber;  
 return sum;  
 }  
}

There are two functions called sum inside of the class main, however the first instance of the function sum only takes two arguments, while the second function takes three arguments. This is also known as function *overloading*, which is static polymorphism when done inside the same class. Static polymorphism is done at compile time, while the other form of polymorphism known as dynamic polymorphism is done during run time.   
  
Dynamic polymorphism is done through function *overriding*, which is when a child class has the same function name, number of arguments and return type, however the instructions inside of the code block are different. So, the child object can share a function name with the parent object even if the code inside the function is completely different.

# Inheritance

Inheritance as a concept in object orientated languages is used to create a new class, based on existing methods and variables from another class. In the following example,

# Encapsulation

# Instance of a class

# Instance method

# Instance variable

# Class method

# Class variable

# Immutable

# Class diagram

# Static

Static is a keyword that means the variable or method it is attached to can be called by the class itself, which is useful for creating a variable that is known by all members of that class.

static int numberOfBicycles  
int ID

“Static Int numberOfBicycles means all Bicycles known it because the class Bicycles knows it.   
numberOfBicycles could be used when setting the ID to ensure that all Bicycles created have a unique idea. ”

# Abstract Classes

# Derived Classes

Derived Classes are classes which inherit from a Parent Class, which leads to Derived Classes also being known as child classes.

# Scope

# Subclass

# parent class

A Parent class is a class that has a child class, also known as a derived class, which inherits all functions and variables of itself. This means that the child class has every variable and can use every method of the parent class if nothing in the child class overrides the method of the parent class.

# Base class

# Method overloading

Method overloading is when a class has multiple functions with the same name, however they differ in one of the 3 following ways.

1. Number of Arguments
2. Data Type of Arguments
3. Order of Arguments

Number of Arguments is simple enough to explain, if there are two methods with the same name, the compiler will choose the method that matches the number of arguments.   
The next way is the data type of arguments. If there are two methods with the same number of arguments, but one takes 2 string arguments and the other takes 2 int arguments, and the method is called with 2 string arguments. Then the method called is the first one of the two.

The final way is Order of Arguments, which will require an example.

# Array

# Collections

# Final

final is the keyword used to make a variable the equivalent of a constant in other languages, meaning that the variable cannot be changed or modified after creati334on.

# Interface

# Enum

# Class

# Getter

Getters are methods created for a class that are made to retrieve a single variable stored in the object, and return it to where it was called for.

# Setter

Setters are methods created for a class that are created to take a variable from its arguments and store its value into a specified variable in itself. Typically, setters are named along the lines of “setAge” with the variable inside of the object being modified being named “age” as well.

# Objects

# Super

Super is a keyword used inside of child class’s to refer to variables or methods of the parent class. Often used to get the original version of a method of the parent class that has been override by the child class. It can also be used to call the parent class’s method of the same name with the intended variables, and when that is completed continue with code unique to the child class’s method with the same name.

# ArrayList

# UML

# This (In the JAVA Programing Language)

# Constructor

# References