Programming Fundamentals

Covers OOP, POP, EDP and algorithms.

Written by Shenesh Perera of IDM, for LO2 in the programming unit 1 assignment.

2018

Shenesh Perera

IDM

9/24/2018

Contents

[Introduction 2](#_Toc527046515)

[Object Oriented Programming 3](#_Toc527046516)

[Breakdown 3](#_Toc527046517)

[Encapsulation 4](#_Toc527046518)

[Inheritance 5](#_Toc527046519)

[Polymorphism & Abstraction 5](#_Toc527046520)

[Procedural & Event-driven Programming 6](#_Toc527046521)

[Procedural Programming vs Object Oriented Programming 6](#_Toc527046522)

[Examples of the differences in the naming in POP vs OOP 7](#_Toc527046523)

[Event-Driven Programming vs Object-Oriented Programming 7](#_Toc527046524)

[The Integrated Development Environment 8](#_Toc527046525)

[Features of an IDE 8](#_Toc527046526)

[Sorting Algorithms 9](#_Toc527046527)

[Merge Sort 9](#_Toc527046528)

[Insertion Sort 9](#_Toc527046529)

[Bubble Sort 10](#_Toc527046530)

[Quick sort 10](#_Toc527046531)

[Counting Sort 10](#_Toc527046532)

[Selection Sort 10](#_Toc527046533)

[Implementation of Selection Sort 10](#_Toc527046534)

[Binary Search Implementation 12](#_Toc527046535)

**Blank Page**

# Introduction



Figure 1.0, Medium.freecodecamp.com, July 26/2018, Marcin Moskala

Programming has been a historical landmark, which has grown in many magnitudes and directions. It has been always on the edge of evolution by the end of every year. Way back in the times, when programming was just a tuning and toggling of a panel of switches, today’s modern method of programming might have been a fantasy.

Since then conventions such as structured and object-oriented programming were established, concepts and design paradigms were introduced, methodology to organize data known as data structures & algorithms were standardized and the usage of tools like interpreters, linters, compilers, debuggers and the giants like Integrated Development Environments soothed the hassle of developing large codebases.

This report will strategically breakdown and focus on development conventions and types of programming then attempt to explain IDEs, proceeded by the describing a multiple sorting algorithms and conclude with the implementation of *Selection Sort* and *Binary Search*.

# Object Oriented Programming

## Breakdown

Object-oriented programming (OOP) is a model that revolves around objects rather than a particular set of data or a predefined logic. Programming has always been viewed in the past as a way to simply give realistic substance to logic in such a way for a particular set of or single input(s), an expected output will be reached.

The need for OOP arose due to a set of developers finding the restriction of programming just being tied to logic, and instead required the ability on “how” to define the data that will *then* be manipulated by logic.

Ever since this particular model was implemented, the way programming was done, changed.

The first step when it comes to OOP is what’s factually known as “*data modeling*”. In this particular step, the programmer should identify what his data objects are and what types of relationships each object has with the rest of the objects.

After data modeling has been completed, the sum of objects are “generalized” as a class of objects, which will attempt to basically give a clear understanding or a labeling of some sort that will identify what type of data each object contains, which gives a direct idea on what type of logical or otherwise steps can be taken in order to manipulate the class.

This concept called classes will allow a programmer to create *custom* data types when needed to satisfy the purpose of the programmer that is not already incorporated within the language itself.

Classes also ensures reusability not only from the file it is defined but also in countless other OOP projects which makes distribution of OOP codebases very easy.

Custom or technically known as an abstract data type allows a programmer to employ data structures effectively to organize and manipulate data.

These “logical steps” are referred to as “methods” in OOP. Methods are employed by classes in order to grant the programmer to effectively and logically pursue any necessary requirement. You could say that methods grant the programmer an interface to interact with the objects in the class.

Classes play a huge role in OOP; one big factor that puts OOP apart from the rest is “inheritance”. OOP allows subclasses to be created from a class. Subclasses can be thought of as children to a parent class that can “inherit” the parent’s attributes.

Inheritance allows a programmer to, through creating subclasses from a parent class analyzes the objects of a class, cuts the unnecessary wasting of time on re-inventing the wheel and efficiency during the process of coding.

The OOP model also provides security to its objects through what’s known as “encapsulation”. Encapsulation is concerned with giving access to objects only when there is a necessity in giving access which in other words means that the likelihood of the objects in a class being corrupted can be reduced effectively.

While corruption is reduced, this is a major boost to the security of the application written using OOP concepts, as it reduces the vulnerability of unauthorized access to data of a class.

The Object Oriented Programming model has been employed in countless programming languages, Java, Python, C, C++, C#, Smalltalk, Delphi, Eiffel etc. The first OOP language however was known as Simula at 1965, by **Ole-Johan Dahl** & **Kristen Nygaard,** however is no longer industrially used.

Features of OOP

To capture the characteristics and standards of OOP, I will use the programming language, Java, to explain.

## Encapsulation

Java provides a programmer encapsulation features through access modifiers. Namely there are 4, public, private, protected and default. Each access modifier allows only a particular level of access to be made.

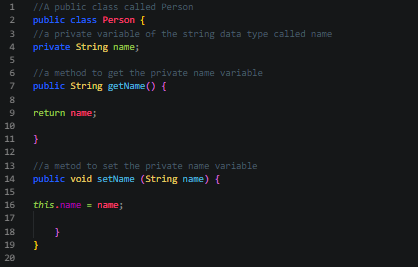
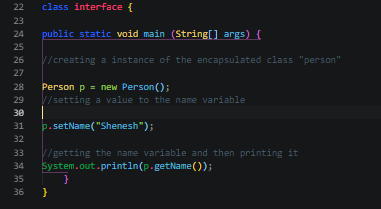


Figure 1.1, Shenesh Perera, September 19th/2018

Figure 1.1 demonstrates how access modifiers have been used, the getName() method and the setName() method are publicly accessible but the String variable “name” can only be accessed by these methods from within the class only. This will keep the string name from being accessed unnecessarily or be lead to corruption.

Figure 1.2, Shenesh Perera, September 19th/2018

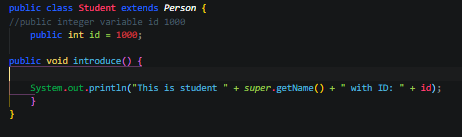


Here the Person class is initialized inside the main class of Java which is also public inside the interface class and the method setName() is called to set the name object of the class Person to “Shenesh” then the name object is printed to the console with the help of the getName() method.

## Inheritance

Java uses the extend keyword to create a subclass from a parent class. When a subclass is created, all the attributes of the parent class are inherited by the child.

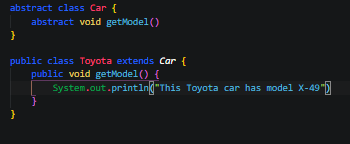
Figure 1.3, Shenesh Perera, September 19th/2018



In Figure 1.3, the method introduce in the subclass called Student of the parent class Person prints a particular string that requires the name of the person. This name is obtained using getName() but in order to call getName() from the Person class, the super keyword is used. When introduce() is called the sentence “This is student Shenesh with ID: 1000” will be printed to the console.

## Polymorphism & Abstraction

Figure 1.4, Shenesh Perera, September 19th/2018



Abstraction is the process by which a developer hides everything other than the required data in order to ensure efficiency. Here an abstract class called Car is created with the abstract method called getModel(), then a subclass Toyota is created from it, which again has a getModel() method. This particular getModel() method overrides the getModel() method of the abstract class Car through the functionality of polymorphism. Polymorphism is the method by which multiple subclasses contain different implementations of the same method to cater towards the needs of the subclasses.

# Procedural & Event-driven Programming

Procedural and Event-driven programming paradigms are not directly implemented in programming languages as much as OOP. In the modern day, these 2 paradigms have become sub-features of programming languages.

Procedural programming is focused on what’s known as a procedure call and this paradigm is directly influenced by structured programming.

Procedural Programming is also known as Procedure Oriented Programming (POP) due to this reason.

Event-driven programming is focused on events to determine the flow of the program, with the help of event handlers such as pressing keys on the keyboard etc.

They have a lot of commonalities with the OOP paradigm and the features they provide already are provided by OOP.

Therefore the features of each paradigm without being stated all over again will be compared to OOP and then be identified as to what makes these 2 paradigms different from OOP.

# Procedural Programming vs Object Oriented Programming

The ideology behind procedural programming is to break down an achievable task into a bunch of variables, subroutines and data structures but in OOP it is as we now know is to use data handling and identify data as objects and expose methods that allow a programmer to interact with the data.

Procedural programming uses “procedures” instead of “methods” on data structures while OOP has the this functionality and more in order to allow a programmer to operate an instance of a class that allows the existence of its own data structure. This is one crucial difference between the 2.

As previously mentioned the 2 have a lot of similarities, although the naming conventions of procedural programming is different, the functionality and the semantics behind them are the same.

|  |  |
| --- | --- |
| Examples of the differences in the naming in POP vs OOP | |
| Procedure | Method |
| Modules | Classes |
| Record | Object |

Table 1.0, Shenesh Perera, Sept 20th 2018

As such if a programmer that is already used to either paradigms wishes to learn the other paradigm, then he will not have a difficulty fighting the friction after the simple nomenclature differences and some other minor changes to behavior as previously mentioned are understood.

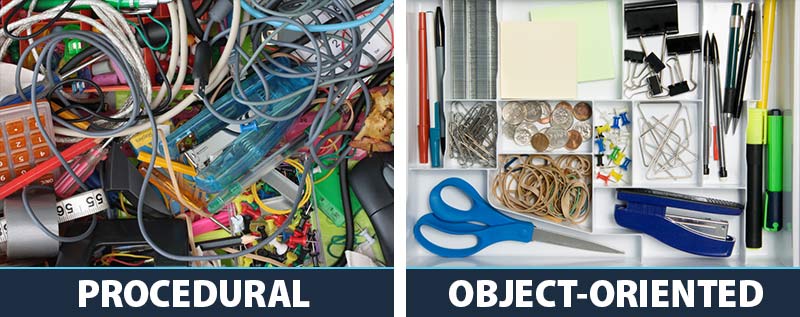


Figure 1.5, zionandzion.com, Will Yowell, September 1st 2015

# Event-Driven Programming vs Object-Oriented Programming

As much as POP is orthogonal with OOP, EDP is also orthogonal. EDP paradigm is almost always implemented with a OOP in a programming language.

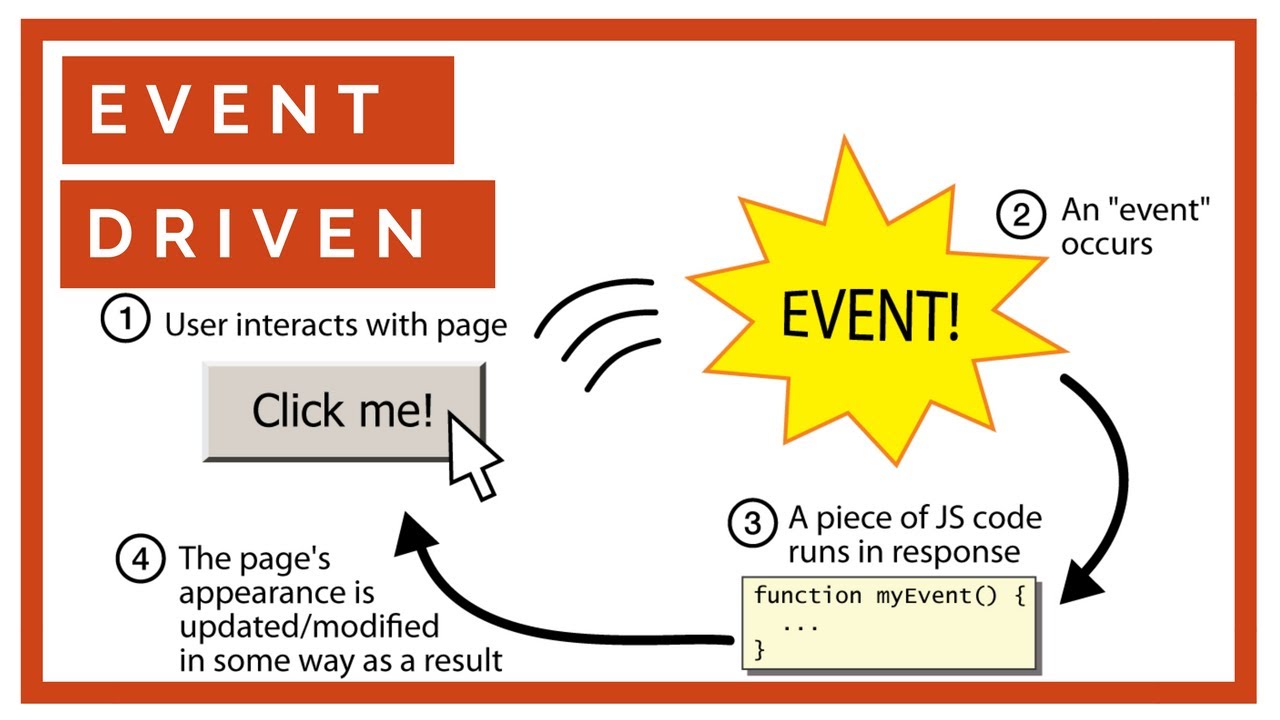
The control flow of an OOP language will be inverted when EDP paradigms are implemented. EDP establishes what’s known as an event loop, which is used to publish and subscribe to events using event handlers.

In OOP, control scope moves from one object to another when a method is called but in an EDP paradigm, control moves from one object to another through an event notification.

EDP and OOP are however mutually exclusive, there really is no meaningful ground when the 2 paradigms are compared in terms of differences, simply because the 2 address different problems.

Most languages like Java heavily based EDP using OOP theory. For example, in Java the event listener, source and everything including the event itself is an object.

Figure 1.6, youtube.com, Tech Primers, Jul 25, 2017



EDP paradigms allow direct user interaction with the program using event handlers, in figure 1.6 for example when the button is clicked, the onClick event handler is triggered so the code in the handler is run. As a result of this code execution, the appearance of the page is modified.

# The Integrated Development Environment

An Integrated Development Environment (IDE) is an application software that provides basic tools and more required to write and test code and eventually create software.

The tools like compilers, libraries, interpreters, testing platforms and frameworks, text editors etc. exist standalone whenever a programmer needs it but an IDE has all of this under one roof.

Without an IDE, a developer must think, select, deploy, integrate then maintain all these tools separately. These tools are “integrated” as one hence the naming. The goal behind this integrated toolset is to simplify the process of coding and to ensure efficiency for a developer.

## Features of an IDE

An IDE usually consists of a code editor, compiler or an interpreter and a debugger granted access through a single graphical user interface. The developer writes and edits the code in the code editor.

The compiler translates this code also known as the source code into a readable form that is executable by a computer system.

The debugger tests the entire source code to point and fix issues or bugs.

An IDE can also provide much more advanced features like object and data modeling, unit testing, source code libraries, automation tools, a terminal and/or a package manager.

The GUI of an IDE is divided into a lot of menus and toolbars, the menu bars of the IDE usually include navigation to features like error diagnostics, formatting, IntelliSense, linting and beautifying code.

More modern IDEs are capable of directly interacting with version control platforms like GitHub, GitLab and BitBucket.

An IDE can provide support for development with standard design models like the Model-View-Controller(MVC).

Most modern IDEs allow the usage of extensions that are developed by fellow developers of the community that uses the IDE in order to provide more and more features or extend features that already exist in the IDE.

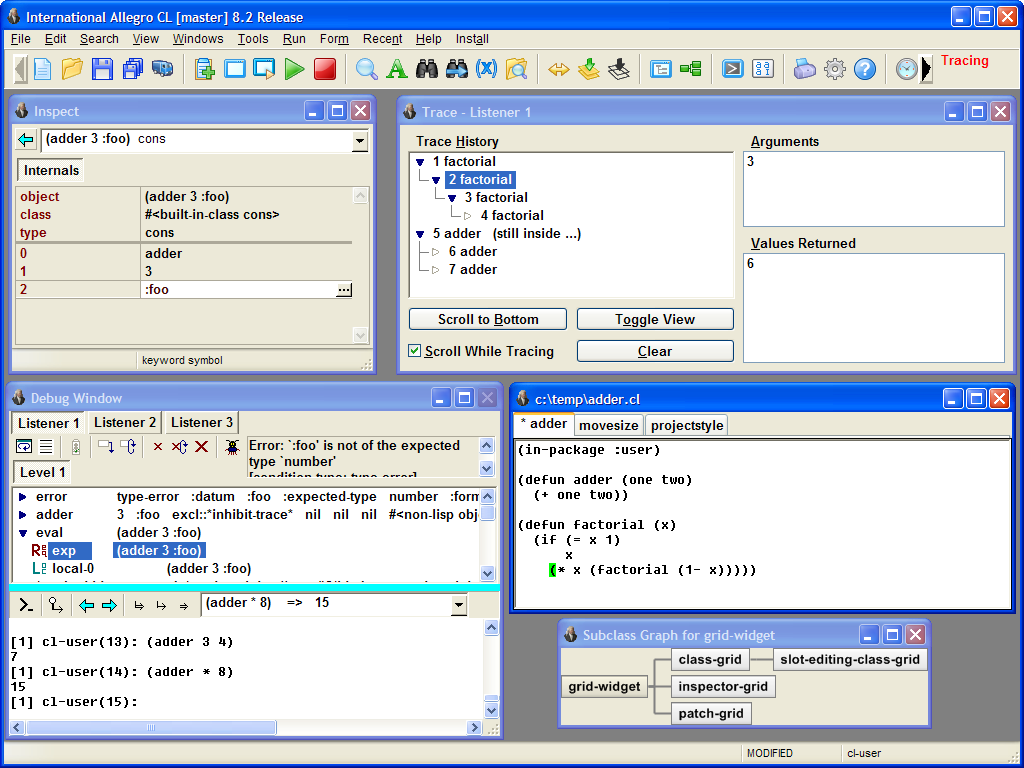


Figure 1.7, franz.com, FranzInc

# Sorting Algorithms

A sorting algorithm is consists of a set of instruction that takes an array of data as input then performs operations native to the sorting algorithm on the array upon completion provides a sorted array of the data. Most algorithms such as the following are comparison-based.

## Merge Sort

Merge sort is based on comparison, focusing primarily on merging 2 pre-sorted arrays so that the final array is sorted the way we wish it to be.

## Insertion Sort

Insertion sort attempts to build a fully sorted array, one element at a time. It goes through the array that is given and removes 1 element per round, then matches the place where the element should belong in the array so that it would be sorted, then places the element in that position.

## Bubble Sort

Bubble sort compares each adjacent pair of elements in the array then swaps them if they are not in order until the entire array is completely sorted.

## Quick sort

Quicksort picks an element from the array that’s given and calls it the pivot element, then splits them into 2 sub-arrays such that all elements follow conditional order with respect to the cardinality of all the elements in the array.

## Counting Sort

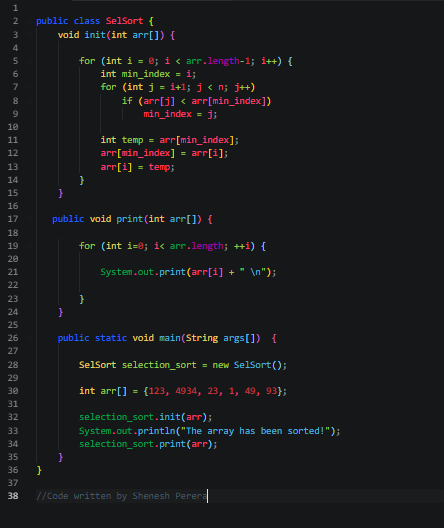
Counting sort is affiliated with integer data that assumes that each element in the array has a key value ranging from 0 to X for some integer X. For every element in the array, this sorting algorithm determines the number of elements that are less than the current element, then with this information counting sort attempts to correct the position of each element such that the resulting array is sorted.

## Selection Sort

Selection sort, takes an array then sorts it by repeatedly searching for the minimum element while maintaining ascending order from the unsorted part then putting it at the very beginning of the array. The algorithm employs the usage of 2 subarrays, one which consists of the sorted part and the other with the rest that is unsorted.

In each round of the sorting algorithm, the minimum value from the unsorted subarray is moved to the sorted subarray such that the resulting array will be sorted in ascending order.

# Implementation of Selection Sort



The implementation has been done using Java.

# Binary Search Implementation

The implementation has been done using Java.

