PROGRAMMING FUNDAMENTALS

RIHAM AHAMED ABDUL RAHEEM
HND COMPUTING IDM

Contents

INTRODUCTION

Programming has been a historical landmark, which has grown in many magnitudes and directions. it's been always on the sting of evolution by the top of each year. Way back within the times, when programming was just a tuning and toggling of a panel of switches, today’s modern method of programming may need been a fantasy.

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

This report will strategically breakdown and specialise in development conventions and kinds of programming then plan to explain IDEs, proceeded by the describing a multiple sorting algorithms and conclude with the implementation of Selection Sort and Binary Search.

Figure 1: Introduction

OBJECT ORIENTED PROGRAMMING

Breakdown

Object-oriented programming (OOP) may be a model that revolves around objects instead of a specific set of knowledge or a predefined logic. Programming has always been viewed within the past as how to easily give realistic substance to logic in such how for a specific set of or single input(s), an expected output are going to be reached.

The need for OOP arose thanks to a group of developers finding the restriction of programming just being tied to logic, and instead required the power on "how" to define the info which will then be manipulated by logic.

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

The first step when it involves OOP is what's factually referred to as "data modeling". during this particular step, the programmer should identify what his data objects are and what sorts of relationships each object has with the remainder of the objects.

After data modeling has been completed, the sum of objects are "generalized" as a category of objects, which can plan to basically provides a clear understanding or a labeling of some sort which will identify what sort of data each object contains, which provides an immediate idea on what sort of logical or otherwise steps are often taken so as to control the category .

This concept called classes will allow a programmer to make custom data types when needed to satisfy the aim of the programmer that's not already incorporated within the language itself.

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

Custom or technically referred to as an abstract data type allows a programmer to use data structures effectively to arrange and manipulate data.

These "logical steps" are mentioned as "methods" in OOP. Methods are employed by classes so as to grant the programmer to effectively and logically pursue any necessary requirement. you'll say that methods grant the programmer an interface to interact with the objects within the class.

Classes play an enormous role in OOP; one big factor that puts OOP aside from the remainder is "inheritance". OOP allows subclasses to be created from a category . Subclasses are often thought of as children to a parent class which will "inherit" the parent's attributes.

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

The OOP model also provides security to its objects through what's referred to as "encapsulation". Encapsulation cares with giving access to things only there's a necessity in giving access which in other words means the likelihood of the objects during a class being corrupted are often reduced effectively.

While corruption is reduced, this is often a serious boost to the safety of the appliance written using OOP concepts, because it reduces the vulnerability of unauthorized access to data of a category .

The Object Oriented Programming model has been employed in countless programming languages, Java, Python, C, C++, C#, Smalltalk, Delphi, Eiffel etc. the primary OOP language however was referred to as Simula at 1965, by Ole-Johan Dahl & Kristen Nygaard, however is not any longer industrially used. (Lewis, 2020)

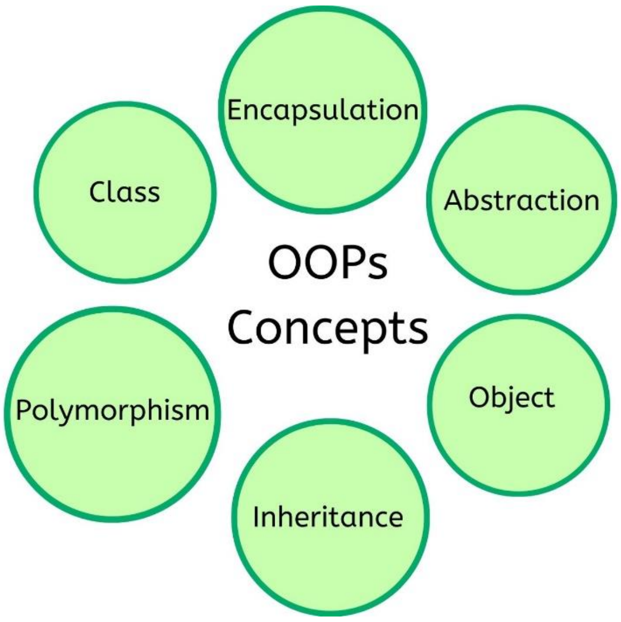


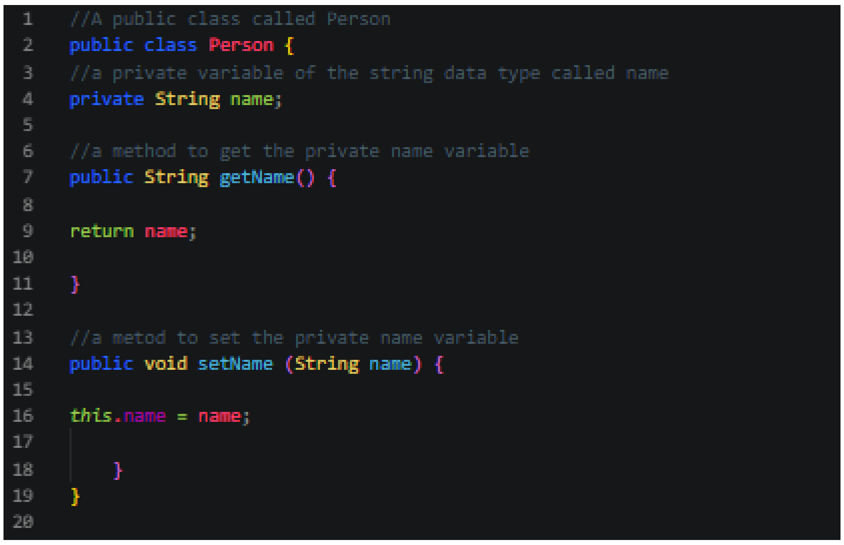
Figure 2: OOP Concepts

Concepts of OOP

To capture the characteristics and standards of OOP, I will be able to use the programing language, Java, to elucidate.

Encapsulation

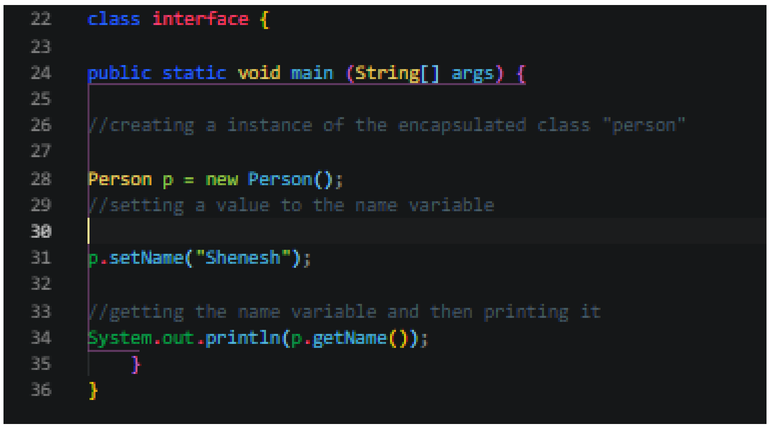
Java provides a programmer encapsulation features through access modifiers. Namely there are 4, public, private, protected and default. Each access modifier allows only a specific level of access to be made. (Tutorialspoint, 2021.)



```java
//A public class called Person
public class Person {
//a private variable of the string data type called name
private String name;

//a method to get the private name variable
public String getName() {

return name;

}

//a metod to set the private name variable
public void setName (String name) {

this.name = name;

    }
}

```

Figure 3: Java encapsulation

Figure 3, demonstrates how access modifiers are used, the getName() method and therefore the setName() method are publicly accessible but the String variable "name" can only be accessed by these methods from within the category only. this may keep the string name from being accessed unnecessarily or be cause corruption.



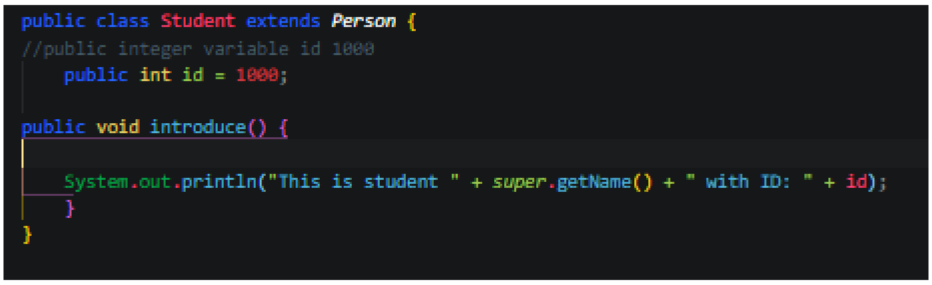
```java
class interface {

public static void main (String[] args) {

//creating a instance of the encapsulated class "person"

Person p = new Person();
//setting a value to the name variable

p.setName("Shenesh");

//getting the name variable and then printing it
System.out.println(p.getName());
    }
}
```

Figure 4: Java Class

figure 4, Here the Person class is initialized inside the most class of Java which is additionally public inside the interface class and therefore the method setName() is named to line the name object of the category Person to “Shenesh” then the name object is printed to the console with the assistance of the getName() method.

Inheritance

Java uses the extend keyword to make a subclass from a parent class. When a subclass is made, all the attributes of the parent class are inherited by the kid. (Tutorialspoint, 2021)



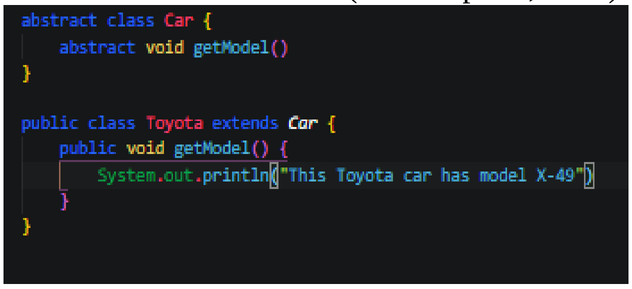
```
public class Student extends Person {
//public integer variable id 1000
    public int id = 1000;

public void introduce() {

    System.out.println("This is student " + super.getName() + " with ID: " + id);
    }
}
```

Figure 5: Inheritance

In Figure 5, the tactic introduce within the subclass called Student of the parent class Person prints a specific string that needs the name of the person. This name is obtained using getName() but so as to call getName() from the Person class, the super keyword is employed . When introduce() is named the sentence “This is student Shenesh with ID: 1000” are going to be printed to the console.

Polymorphism & Abstraction

Abstraction is that the process by which a developer hides everything aside from the specified data so as to make sure efficiency. Here an abstract class called Car is made with the abstract method called getModel(), then a subclass Toyota is made from it, which again features a getModel() method. This particular getModel() method overrides the getModel() method of the abstract class Car through the functionality of polymorphism. Polymorphism is that the method by which multiple subclasses contain different implementations of an equivalent method to cater towards the requirements of the subclasses. (tutorialspoint, 2021)



```
abstract class Car {
    abstract void getModel()
}

public class Toyota extends Car {
    public void getModel() {
        System.out.println("This Toyota car has model X-49")
    }
}
```

Figure 6: Polymorphism

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. (Tutorials, 2021)

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 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.

Table 1: PP vs OOP

PP	OOP
Procedure	Method
Modules	Classes
Record	Object

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.

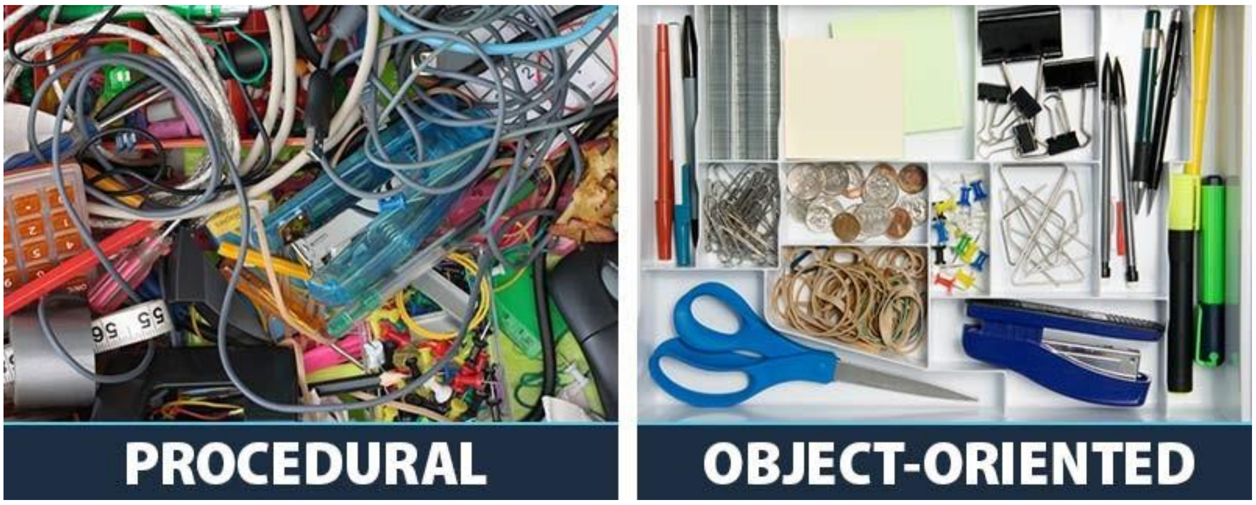


Figure 7: Procedural vs OOP

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.

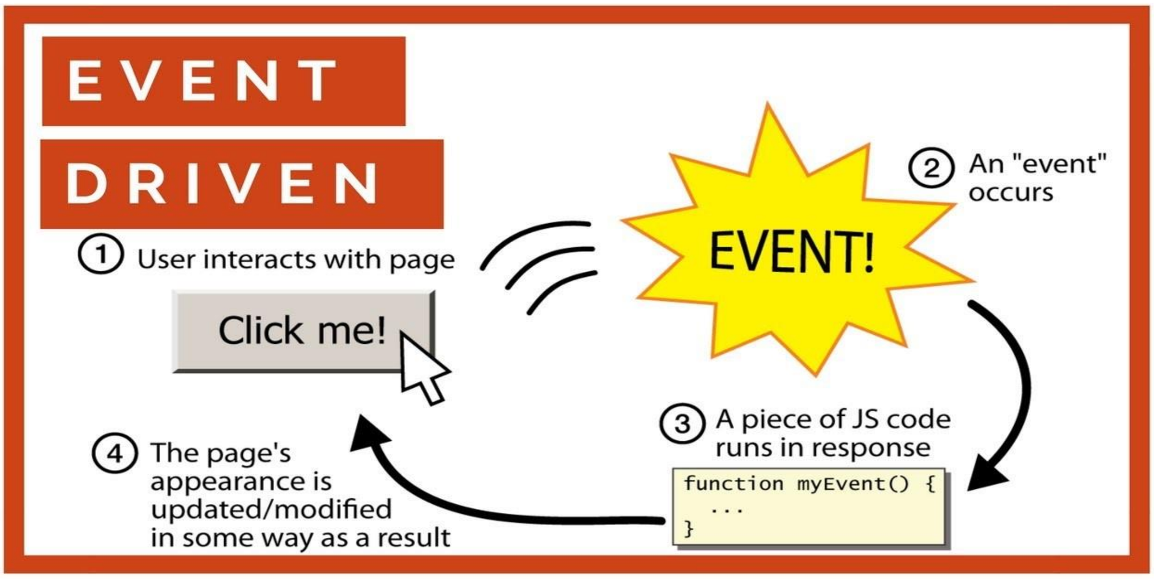

EVENT
DRIVEN
① User interacts with page
Click me!
EVENT!
② An "event" occurs
③ A piece of JS code runs in response
function myEvent() {
...
}
④ The page's appearance is updated/modified in some way as a result

Figure 8: Event Driven

EDP paradigms allow direct user interaction with the program using event handlers, in figure 8, 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.

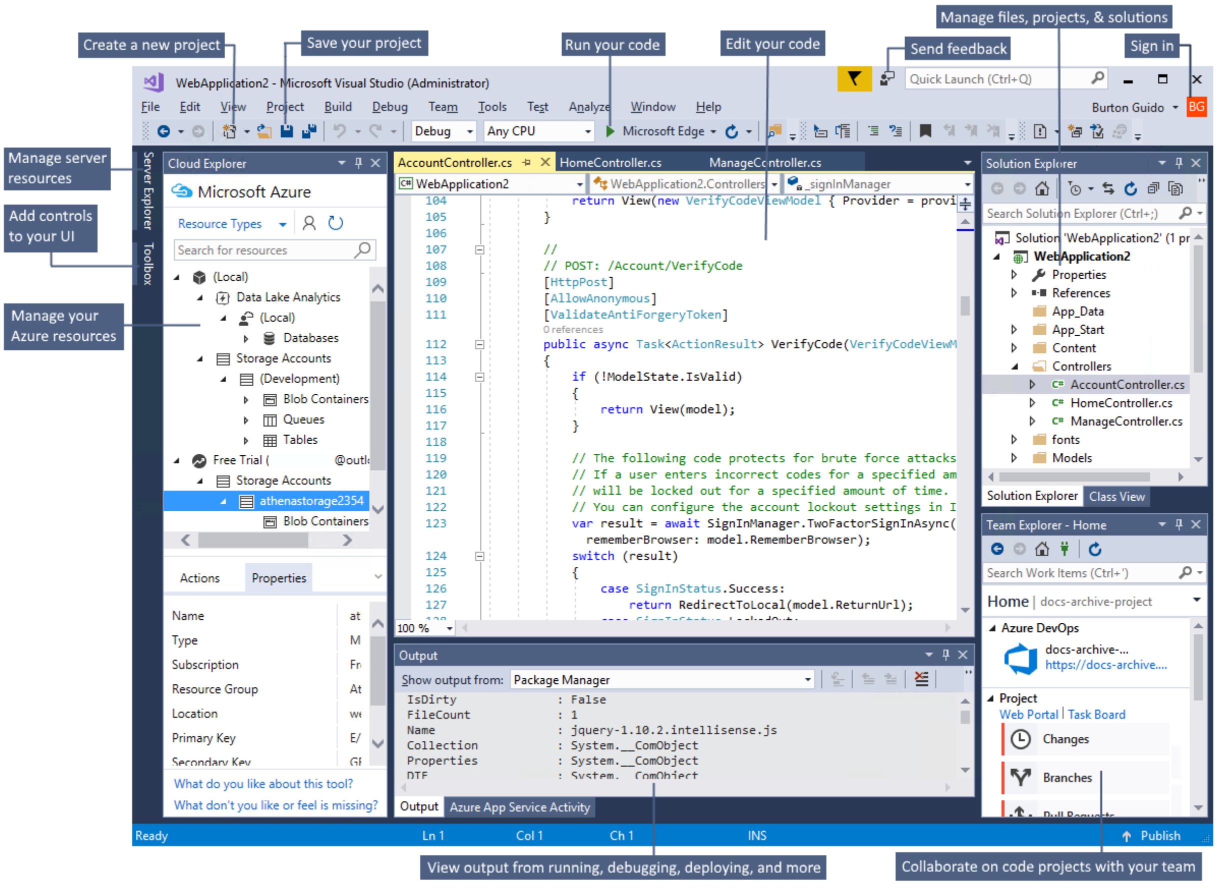
An IDE can also provide much more advanced features like object and data modeling, unit testing, source code libraries, automation tools, a terminal and/or a package manager.

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.



```
            return View(new VerifyCodeViewModel { Provider = provi
        }

        //
        // POST: /Account/VerifyCode
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> VerifyCode(VerifyCodeViewM
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }

            // The following code protects for brute force attacks
            // If a user enters incorrect codes for a specified am
            // will be locked out for a specified amount of time.
            // You can configure the account lockout settings in I
            var result = await SignInManager.TwoFactorSignInAsync(
              rememberBrowser: model.RememberBrowser);
            switch (result)
            {
                case SignInStatus.Success:
                    return RedirectToLocal(model.ReturnUrl);
```

Show output from: Package Manager

```
IsDirty                : False
FileCount              : 1
Name                   : jquery-1.10.2.intellisense.js
Collection             : System.__ComObject
Properties             : System.__ComObject
DTE                    : System.__ComObject
```

Figure 9: IDE

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.

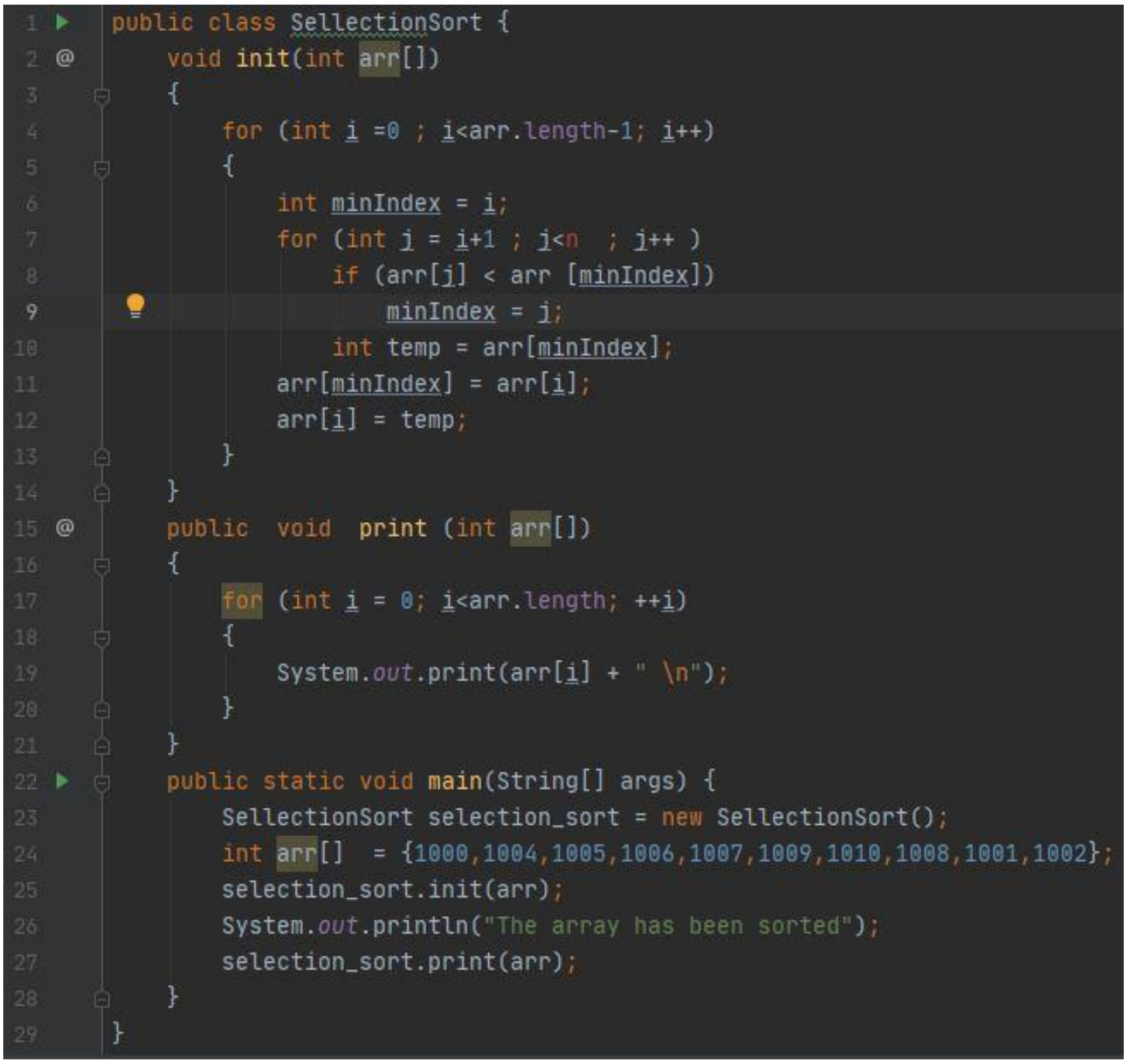
BINARY SEARCH

Find an ordered matrix by dividing the search interval in half repeatedly. Start with an interval that spans the entire sequence. If the search key value is less than the item in the middle of the range, decrease the range to the lower half. Otherwise, reduce to the upper half. Check repeatedly until the value is found or the range is empty.

IMPLEMENTATION

Selection Sort

The implementation has been done using Java.

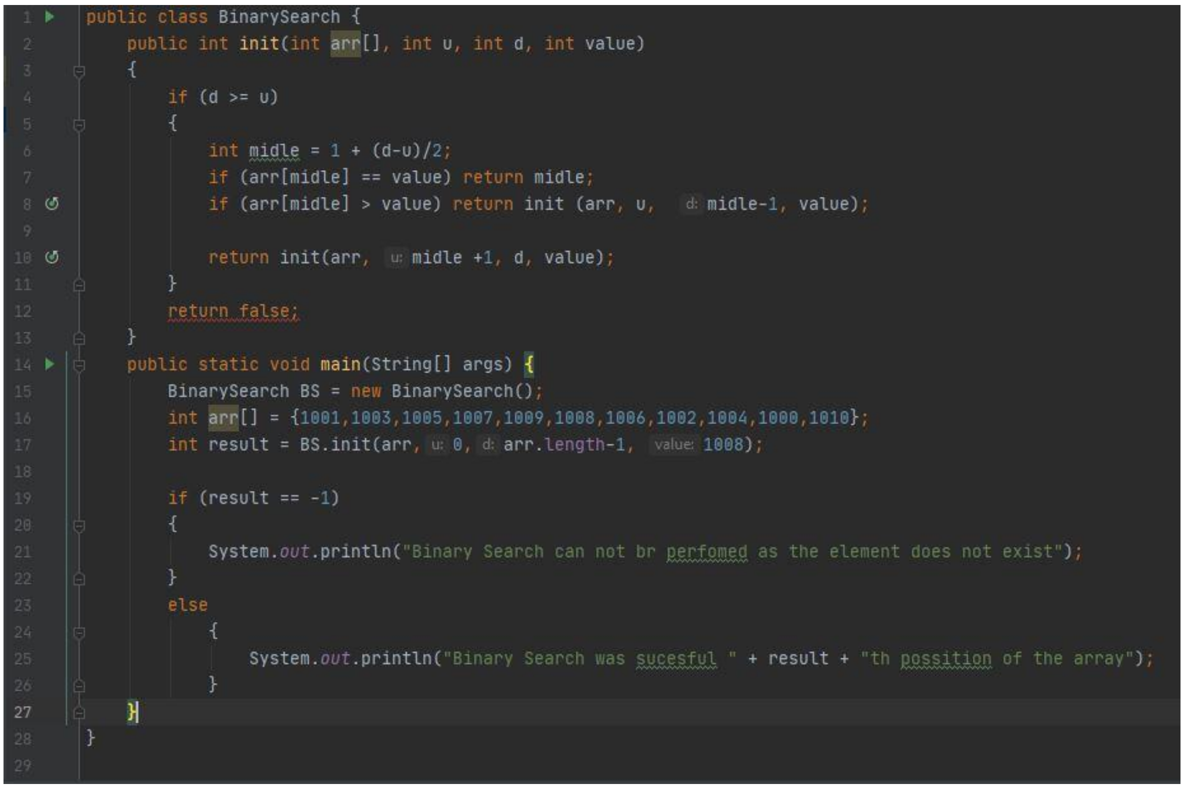


```java
public class SellectionSort {
    void init(int arr[])
    {
        for (int i =0 ; i<arr.length-1; i++)
        {
            int minIndex = i;
            for (int j = i+1 ; j<n  ; j++ )
                if (arr[j] < arr [minIndex])
                    minIndex = j;
                int temp = arr[minIndex];
            arr[minIndex] = arr[i];
            arr[i] = temp;
        }
    }
    public  void  print (int arr[])
    {
        for (int i = 0; i<arr.length; ++i)
        {
            System.out.print(arr[i] + " \n");
        }
    }
    public static void main(String[] args) {
        SellectionSort selection_sort = new SellectionSort();
        int arr[]  = {1000,1004,1005,1006,1007,1009,1010,1008,1001,1002};
        selection_sort.init(arr);
        System.out.println("The array has been sorted");
        selection_sort.print(arr);
    }
}
```

Figure 10:Implementation of Selection Sort

Binary Search

The implementation has been done using Java.



```java
public class BinarySearch {
    public int init(int arr[], int u, int d, int value)
    {
        if (d >= u)
        {
            int midle = 1 + (d-u)/2;
            if (arr[midle] == value) return midle;
            if (arr[midle] > value) return init (arr, u,  d: midle-1, value);

            return init(arr,  u: midle +1, d, value);
        }
        return false;
    }
    public static void main(String[] args) {
        BinarySearch BS = new BinarySearch();
        int arr[] = {1001,1003,1005,1007,1009,1008,1006,1002,1004,1000,1010};
        int result = BS.init(arr,  u: 0,  d: arr.length-1,  value: 1008);

        if (result == -1)
        {
            System.out.println("Binary Search can not br perfomed as the element does not exist");
        }
        else
            {
                System.out.println("Binary Search was sucesful " + result + "th possition of the array");
            }
    }
}
```

Figure 11: Binary Search Implementation

References

Lewis, S., 2020. *TechTarget.* [Online]
Available at: https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP
[Accessed 15 January 2021].

Tutorials, M. I. –. P. B., 2021. *What are Procedural, Event Driven and Object Orientated Programming.* [Online]
Available at: https://www.mooict.com/what-are-procedural-event-driven-and-object-orientated-programming/
[Accessed 15 January 2021].

Tutorialspoint, 2021. . *Encapsulation.* [Online]
Available at: https://www.tutorialspoint.com/java/java_encapsulation.htm
[Accessed 15 January 2021].

Tutorialspoint, 2021. *Inheritance.* [Online]
Available at: https://www.tutorialspoint.com/java/java_inheritance.htm
[Accessed 15 January 2021].

tutorialspoint, 2021. *Polymorphism.* [Online]
Available at: https://www.tutorialspoint.com/java/java_polymorphism.htm
[Accessed 15 January 2021].