



**Programming**  
**30201101**  
**D/615/1618**

**Section (1)**

**Final Assignment**

**Submitted to**  
Eng. Dania Alsaid

**Submitted on**  
June 19<sup>th</sup>, 2022

**Submitted by**  
Marwan Tareq Shafiq Al Farah

**Student ID**  
21110011

**Spring 2021 – 2022**

## *Part 1*

### 1.1. **The process of building an application:** (Lehman and Belady, 1985; TechyV, 2020)

- 1- **Identify and Define the Problem:** the first stage that I did is to thoroughly evaluate the problem to determine a viable solution. There may be numerous solutions to a single problem, so I always try to figure out what solution best fits my program's requirements.
- 2- **Design and Plan for a Solution:** the following step is to figure out how I am going to turn that solution into an executable program after I have identified the components that I will need to solve my problem and stated what shape my solution will take. In this step I usually like to draw a flow chart and write a pseudo code.

#### **The process of building an application from writing to execution:**

- 3- **Code the Program:** it includes 3 steps:
  - a) **Coding:** it is the step that involves translating the design into the syntax of the programming language that is being used for the design to become a functioning program.
  - b) **Compiling:** it is the process of converting the source code from a high-level language such as Java to a low-level language such as assembly language and then machine language, therefore, it converts the source code to 0s and 1s which makes the computer able to understand it.
  - c) **Debugging:** after the source code gets compiled, it will most likely return to you with a list of errors that need to be solved for the program to function properly. Debugging is the process in which you look at the code, identify any syntax, logical, or runtime errors, fix them, and then try to recompile the code. These steps aren't just done after all the coding was completed, rather they can be done while in the process of coding the program.
- 4- **Test and Check the Solution:** in this final step, I like to make sure that the code does what it is supposed to do without any logical or runtime errors occurring. I usually try to enter a variety of different inputs and think as clumsy as possible to make sure that the program functions perfectly, regardless of what the input was, for the program to be as efficient as possible.

If any errors get detected, the issue should be resolved by modifying and debugging the code. At this point, rigorous retesting is required to ensure that this fix does not break anything else in the code, and that you don't have any other hidden errors.
- 5- **Final Output:** after all of the testing and debugging is done, I become certain that my program meets the needs and requirements of it, and that it solves the problem perfectly.

- 1.2. **Algorithm (in coding/programming):** a set of rules or instructions that is done by taking the inputs, performing a series of operations on them, and then returning the wanted output. (Indicative, 2020)

**The Characteristics of a good algorithm:** (Tutorlalspoint, 2021; Upadhyay, 2022)

- 1- **Unambiguous:** any good algorithm should be unambiguous and clear for other coders to follow.
- 2- **Inputs:** any good algorithm that requires inputs, must have well-defined inputs
- 3- **Outputs:** any good algorithm will return one or more well-defined outputs that match the desired outputs.
- 4- **Finiteness:** any good algorithm should have a finite number of instructions, therefore, it cannot be defined as a good algorithm if it ends up in an infinite loop.
- 5- **Feasibility:** any good algorithm has to be simple and generic and does not need some futuristic technologies, therefore, it can be implemented and replicated easily with the available resources.
- 6- **Independence:** any good algorithm has to be independent of the programming language that it originated in, therefore, it can be replicated and implemented using any other programming language.

### 1.3. Explanation of the Algorithm:

First of all, this algorithm starts by declaring and initializing the variable `binaryNumber` of type `int`, then it calculates the value of the `decimalNumber` by calling the method `binaryToDecimal` that takes the `binaryNumber` as a parameter.

In the `binaryToDecimal` method, it starts by declaring and initializing the variables `decimalNumber` and `numOfBinaryDigits` of type `int` so that they both equal 0. After that, a while loop begins which has the condition that while the `binaryNumber` doesn't equal 0 it keeps looping, and in the while loop it adds to the `decimalNumber`, `binaryNumber mod 10` multiplied by base 2 raised to the power of `numOfBinaryDigits`. In the while loop, it also checks if all the digits in the binary number either equal 1 or 0, and if one digit doesn't equal 1 or 0, the method returns -1 to the main method. If all goes well, after the while loop ends the method returns the `decimalNumber` to the main method.

The main method checks if the `decimalNumber` equals -1 it prints "Invalid Binary Number!" and the code ends. If all goes well, the `binaryNumber` and the `decimalNumber` will be printed on the console. Then, the `printDigitalNumber` method will be called that takes the `decimalNumber` as a parameter.

In the `printDigitalNumber` method, it starts by declaring and initializing `num` which is a 2-dimensional array of type `int` in which each row represent a number from 0 to 9 and each column represents 1 line in the digital number. After that, the `numOfDigits` will be calculated using the `calculateNumOfDigits` method. After `numOfDigits` is calculated, a for loop that will iterate 3 times (variable `i` from 0 to 3) begins that will print each horizontal line of the `decimalNumber`. It contains inside it a nested for loop that will iterate `numOfDigit` times (variable `j` from 0 to `numOfDigits`), that will print each digit in each

horizontal line. Then, the variable digit will be calculated by dividing the decimalNumber by base 10 raised to the power of numOfDigits minus 1 minus j all mod 10. Then, the code checks if variable i equals 0 (printing the first horizontal line), and if it is true it checks if num[digit][0] equals 1 then it will print “\_”, and if num[digit][0] doesn’t equal 1 then it will print “ “. If variable i doesn’t equal 0, then it checks if it equals 1 (printing the second horizontal line), if it is true then it will enter a for loop that will iterate 3 times (variable k from 1 to 4) to print each line (either | or \_ ) in each digit in the second horizontal line, that checks inside it if num[digit][k] equals 1, then it checks if k equals 2 then it will print “\_”, and if k doesn’t equal 2 then it will print “|”. If num[digit][k] equals 0 then it will print “ “. If variable i doesn’t equal 0 or 1, then it checks if it equals 2 (printing the third horizontal line), if it is true then it will enter a for loop that will iterate 3 times (variable k from 4 to 7) to print each line (either | or \_ ) in each digit in the third horizontal line, that checks inside it if num[digit][k] equals 1, then it checks if k equals 5 then it will print “\_”, and if k doesn’t equal 5 then it will print “|”. If num[digit][k] equals 0 then it will print “ “. After that, it prints a new line, and when the outer for loop ends the code returns to the main function to end.

Whenever the pow method is called it takes the parameters “base” and “power” of type int, and it declares and initializes the variable result by making it equal to 1. Then, the code enters a for loop that will iterate power times (variable i from 1 to power), in which it will multiply the variable result by the variable base. After the for loop ends, the pow method returns variable result.

Whenever the calculateNumOfDigits is called it takes the parameter “decimalNumber” of type int, and it declares and initializes the variable numOfDigits by making it equal to 0. Then, the code checks if decimalNumber equals 0, and if it is true it will make the numOfDigits equal to 1. If it was false it will enter a while loop that will keep iterating while the decimalNumber doesn’t equal 0, in which it will keep dividing the decimalNumber by 10 and increase the numOfDigits by 1. When the while loop ends, the method returns numOfDigits.

### **Pseudo Code:**

```

start
    input binaryNumber
    calculate decimalNumber by using the binaryToDecimal method by taking
    binaryNumber as the parameter
    if decimalNumber equals -1
        print "Invalid Binary Number!"
        return
    print binaryNumber
    print decimalNumber
    print digital number using the printDigitalNumber method by taking decimalNumber
    as the parameter
end

pow(base, power)

```

```

    result = 1
    for i in 0 to power
        multiply result by base
    return result

binaryToDecimal(binaryNumber)
    decimalNumber = 0
    numOfBinaryDigits = 0
    while binaryNumber doesn't equal 0
        adds to decimalNumber, binaryNumber mod 10 multiplied by base 2 raised to the
        power of numOfBinaryDigits
        if binaryNumber mod 10 doesn't equal 1 and doesn't equal 0
            return -1
        divide binaryNumber by 10
        increase numOfBinaryDigits by 1
    return decimalNumber

calculateNumOfDigits(decimalNumber)
    numOfDigits = 0
    if decimalNumber equals 0
        numOfDigits = 1
    else
        while decimalNumber doesn't equal 0
            divides decimalNumber by 10
            increase numOfDigits by 1
    return numOfDigits

printDigitalNumber(decimalNumber)
    num[][] = {{1, 1, 0, 1, 1, 1, 1},
               {0, 0, 0, 1, 0, 0, 1},
               {1, 0, 1, 1, 1, 1, 0},
               {1, 0, 1, 1, 0, 1, 1},
               {0, 1, 1, 1, 0, 0, 1},
               {1, 1, 1, 0, 0, 1, 1},
               {1, 1, 1, 0, 1, 1, 1},
               {1, 0, 0, 1, 0, 0, 1},
               {1, 1, 1, 1, 1, 1, 1},
               {1, 1, 1, 1, 0, 0, 1}}
    calculates the number of digits by using the method calculateNumOfDigits that takes
    decimalNumber as a parameter
    for i in 0 to 3
        for i in 0 to numberofDigits
            calculate the digit by dividing the decimalNumber by base 10 raised to the
            power of numOfDigits minus 1 minus j all mod 10
            if i equals 0
                if num[digit][0] equals 1

```

```

        print " _ "
    else
        print "  "
    else if i equals 1
        for k in 1 to 4
            if num[digit][k] equals 1
                if k equals 2
                    print " _ "
                else
                    print "|"
            else
                print "  "
        else if i equals 2
            for k in 4 to 7
                if num[digit][k] equals 1
                    if k equals 5
                        print " _ "
                    else
                        print "|"
                else
                    print "  "
            print new line

```

- 1.4. The source code for the algorithm is located under the package “algorithm” with the name of the class being “Algorithm”. The largest binary number that this code can handle is 111111111, and its equivalent in decimal format is 1023, because the binaryNumber variable is defined as an integer because the use of the charAt() method is forbidden. Therefore, after consulting with the instructor she told us that we can turn in 2 source codes: one without the use of charAt, and one with the use of charAt in which binaryNumber is defined as a String which will allow the value of the decimalNumber to reach millions. The second source code is located under the package “algorithm” with the name of the class being “AlgorithmWithCharAt”.

```

Binary Number: 1010101010
Decimal Number: 682
Digital Number:

```

```

_ _ _
|_|_|
|_|_|

```

Firstly, the code is unambiguous and very easy to follow and understand because the code is fully commented, and because it uses functions and methods which divide the code into smaller more manageable pieces making the code easier to understand.

```
static public int pow(int base, int power) //pow method
{
    int result = 1; //declare and initialize result

    //iterate to calculate the result
    for (int i = 0; i < power; i++)
    {
        result *= base; //calculate the result
    }
    return result; //return the result
} //end of pow method
```

Unambiguous

The code also has well-defined inputs (binaryNumber), as well as well-defined outputs (for the decimal number to be calculated and printed, and for the digital number to be printed on the console) that match the expected result as you can see below because the equivalent of the binary number 1010101010 is the decimal number 682.

```
//Binary Number
int binaryNumber = 1010101010;
```

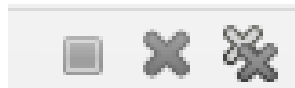
## Well-Defined Inputs

```
Binary Number: 1010101010
Decimal Number: 682
Digital Number:
```

692

## Well-Defined-Outputs

The code is also finite since the code ends immediately after printing the digital number, and therefore, the algorithm didn't enter an infinite loop.



## Finiteness

The code is also very feasible and simple since it doesn't need any futuristic technology to implement, and because it can be replicated easily.

Finally, the algorithm is also independent of Java since it can be replicated using almost any programming language in the world.

The screenshot shows a C++ IDE with a dark theme. The top toolbar includes icons for Run, Debug, Build, Share, Save, and Security. A language dropdown menu is set to 'C'. The main editor displays the following code:

```
#include <stdio.h>
int powe(int base, int power);
int binaryToDecimal(int binaryNumber);
int calculateNumOfDigits(int decimalNumber);
void printDigitalNumber(int decimalNumber);

int main() { //main method

    //Binary Number
    int binaryNumber = 1010101010; //declare and initialize binaryNumber

    //Decimal Number
    int decimalNumber = binaryToDecimal(binaryNumber); //declare and initialize (by going into the method) de

    //Check if the binary number is valid or not
    if(decimalNumber == -1)
```

Below the code editor, there are three input fields labeled "Input":

- Binary Number: 1010101010
- Decimal Number: 682
- Digital Number:

The output area at the bottom shows a grid of characters representing the digital number, followed by the message "...Program finished with exit code 0 Press ENTER to exit console."

Independence  
Same algorithm implemented in C

As you can see in the example below, the implemented code matches the written algorithm in pseudo-code almost perfectly, as almost every line in the pseudo-code is met by an equivalent of that line in the Java language in the source code.

```

start
  input binaryNumber
  calculate decimalNumber by using the binaryToDecimal method by taking
  binaryNumber as the parameter
  if decimalNumber equals -1
    print "Invalid Binary Number!"
    return
  print binaryNumber
  print decimalNumber
  print digital number using the printDigitalNumber method by taking decimalNumber
  as the parameter
end

```

Pseudo Code Example

```

public static void main (String[] args) { //main method

  //Binary Number
  int binaryNumber = 1010101010; //declare and initialize binaryNumber

  //Decimal Number
  int decimalNumber = binaryToDecimal(binaryNumber); //declare and initialize (by going into the method) decimalNumber

  //Check if the binary number is valid or not
  if(decimalNumber == -1)
  {
    System.out.println("Invalid Binary Number!");
    return;
  }

  //print binaryNumber and decimalNumber
  System.out.println("Binary Number: " + binaryNumber);
  System.out.println("Decimal Number: " + decimalNumber);

  //Digital Number
  System.out.println("Digital Number: ");
  printDigitalNumber(decimalNumber); //print digital number
} //end of main method

```

Pseudo Code's equivalent in the implemented source code in Java

## **References:**

Indicative (2020) *What Is A Programming Algorithm? Data Defined, Indicative*. Available at: <https://www.indicative.com/resource/programming-algorithm/>

Lehman, M. M. and Belady, L. A. (1985) *The Programming Process, Program evolution: processes of software change*. Available at: <https://www.cs.bham.ac.uk/~rxb/java/intro/2programming.html>

TechyV (2020) *STAGES / STEPS INVOLVED IN PROGRAMMING - Techyv.com, TechyV*. Available at: <https://www.techyv.com/article/stages-steps-involved-programming/>

Tutorialspoint (2021) *Data Structures - Algorithms Basics, Tutorialspoint*. Available at: [https://www.tutorialspoint.com/data\\_structures\\_algorithms/algorithms\\_basics.htm](https://www.tutorialspoint.com/data_structures_algorithms/algorithms_basics.htm)

Upadhyay, S. (2022) *What Is An Algorithm? Characteristics, Types and How to write it, Simplilearn*. Available at: <https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-an-algorithm>



## *Part 2*

- 2.1. **Programming Paradigm:** A programming paradigm is a type, style, or method of programming and solving problems through the use of programming languages. The difficulty of employing a paradigm varies depending on the language itself. Multiple paradigms can coexist in the same software. (Indicative, 2021)

A programming paradigm does not refer to a specific programming language, but rather to how you program. There are many programming languages, but they all require some kind of strategy when the languages are implemented and that approach is called a paradigm. (Thanoshan, 2019)

- 1- Procedural Programming Paradigm:** it is most likely the first programming paradigm that any coder learns. It is a paradigm that uses a linear top-down approach to lay down a set of instructions that informs the computer on how to execute a task step by step. Procedural Programming consists of a large program that is split down into smaller, more manageable components called procedures or functions. (Bhatia, 2021)

**Characteristics of Procedural Programming Paradigm:**

- a) **It follows a top-down approach.** (Jamshidi, 2021)
  - b) **The program is broken down into functions:** they are blocks of code that each execute a specified purpose. (Jamshidi, 2021)
  - c) **Predefined Functions:** they are functions/methods that are built in by default with the language that is being used, or can be accessed by libraries such as `println()`, `Math.pow()`, `charAt()`, etc... (Bhatia, 2021)
  - d) **Local Variables:** it is a variable that only exists in the scope that it is being used in. (Bhatia, 2021)
  - e) **Global Variables:** it is a variable that is declared outside of any scope of a function/method, and can be accessed anywhere in the code. (Bhatia, 2021)
  - f) **Parameter Passing:** it allow for parameters to be passed to functions/methods so that they can be used in them. (Bhatia, 2021)
- 2- Object-Oriented Programming Paradigm (OOP):** it is a programming paradigm that uses classes and objects. OOP is used to break down a large code into smaller reusable codes (classes) which could be used to build objects or to access static methods without the need to create an object. (Doherty, 2020; Lewis, 2020) An object is a collection of data, processes that are used for the manipulation of the data, and functions that provide data-related information. (Lerdorf, 2017)
- Characteristics of Object-Oriented Programming Paradigm:**
- a) **Encapsulation:** this works by making specific sensitive data private and not accessible outside the class, therefore, it is hidden from users. This works by making the access to your variable/attributes private, and creating getters and setters for them. (Lerdorf, 2017; Herrity, 2021; w3schools, 2021)
  - b) **Polymorphism:** it is the ability for methods with identical names and parameter types to behave differently depending on the receiver. There are 2 types of polymorphism: static polymorphism such as overloading, and dynamic polymorphism such as overriding. (Janssen, 2017; Lerdorf, 2017; Herrity, 2021)

- c) **Inheritance:** it is the ability to define a class of objects that inherits from a parent class. The child class has the ability to own new data elements and methods, but the parent class's data elements and methods are also available to objects in the child class without having to rewrite their declarations. (Lerdorf, 2017)
  - d) **Abstraction:** the basic goal of abstraction is to keep consumers away from any unneeded information. It is the process of picking specific pieces of data from a bigger pool in order to present the user only the relevant details of an item for them. It also allows the code be reused in other places. (Hartman, 2022)
- 3- Event-Driven Programming Paradigm:** it is a paradigm that determines the flow of the code based on various events such as a mouse being clicked, or a key typed on the keyboard. This is done with the help of listeners that detects and listen for events such as Action Listener, Mouse Movement Listener, Mouse Listener, Keyboard listener, etc... The Graphical User Interface (GUI) is one of the most well-known applications of this paradigm. (Tucker and Noonan, 2004)
- Characteristics of Event-Driven Programming Paradigm:** (Kolesnikova, 2022)
- a) **Service-Oriented:** it is used to design programs that are made to provide a service for the user without slowing down the computer and usually these programs work in the background of the operating system.
  - b) **Time-Driven:** some codes use time as their trigger, therefore, it does an process every fixed period of time like every 5 minutes, every hour, every day, every month, etc..., or when a particular time comes.
  - c) **Trigger Functions:** they are the functions that get done when a specific trigger occurs.
  - d) **Events:** events such as a mouse being click, a key on the keyboard being typed, etc..., are the events that trigger a function to happen.

### **The relationship between the programming paradigms:**

All programming paradigms work simultaneously together to help each other reach the goal of solving the problem at hand in the most efficient way possible by having each paradigm provide various unique features that are provided by that paradigm.

For example, one of the major drawbacks of procedural programming is that the program units do not accurately reflect real-world entities, making them not reusable and it is very unusual for programmers to start from scratch on each new project, therefore, by using OOP, programmers can make reusable codes by dividing their code into classes and making use of objects, attributes, methods, etc.... Also, OOP handles the issue of being able to access global data from anywhere in the code in procedural programming, to being able to access this data securely with the use of encapsulation in OOP.

Another example is that procedural programming follows a top-down approach which works very well with any code that doesn't need any type of user intervention, but when user intervention is required, although some operations are able to be handled by the procedural programming, but the event-driven programming paradigm can enable the user to do it much more easily.

When the event-driven paradigm is using the GUI (Graphical User Interface), OOP can make it much easier to handle, as the programmer can divide the JButtons, JTextFields, JFrames all

into classes which will make the code much easier to trace and understand and makes the programmer able to reuse these classes.

## **2.2. Write code examples for the Procedural, Object-Oriented and Event-Driven Programming Paradigms:**

These code examples are located in separate packages called “eventDriven”, “oop”, and “procedural”. “eventDriven” contains several classes that talk about the Object-Oriented Programming Paradigm and the Event-Driven Programming Paradigm which are “Main”, “MainFrame”, “People”, “Student”, “Teacher”, “Employee”, and “Administration”. These classes were built on the OOP and Event-Driven Paradigms, and they also contain some aspects of the Procedural Paradigm as well that will be discussed later on in the report. In the “procedural” package there is another class called “Procedural” that covers the Procedural Paradigm in a more clear and detailed way. And in the “oop” package it contains 4 classes “Main”, “Animal”, “Mammals”, and “Cat” in which they dive deeper into objects.

## **2.3. Compare and contrast between Procedural, Object-Oriented and Event-Driven Programming Paradigms used in the source code:**

First and foremost, each programming paradigm has its own unique characteristics that enables the programmer to code using various tool and techniques.

For example, the class “MainFrame” didn’t exist on its own in the beginning as it was a part of the “Main” class. After examining the code thoroughly, I figured out that the Procedural Paradigm was not the way to go with the “MainFrame” because the Procedural Paradigm doesn’t offer code reusability, and I needed to reuse the “MainFrame” in several parts of my code, then I remembered that OOP does support code reusability, therefore, I created a separate “MainFrame” class. If I had continued with the “MainFrame” code being in the “Main” class, then I would have had to repeat the same piece of code in several parts of the code to be able to achieve the same result.

Also, when I wanted to begin creating the school system that I have done, I thought of coding it using the Procedural Paradigm, as the user could input the data on the console by using ifs, switches, whiles, and for loops to accomplish that, then I decided to convert to the Event-Driven Paradigm, as it supports the GUI which is more appealing visually for the user than the console, and also because when dealing with events, although it is possible to accomplish using the Procedural Paradigm, but the same task could be done much easier using the Event-Driven Paradigm, as it supports triggers, event handlers, listeners, and many others.

OOP has also helped the Event-Driven Paradigm in my code by dividing the various JFrames and all of its components on various methods that can be accessed statically, in different classes to be able to use the wanted JFrame anywhere in my code without the need to rewrite it.

There are more comparisons between the 3 paradigms, that I have used in my source code, in the table below:

Comparison	Procedural	OOP	Event-Driven
<b>Access Modifiers</b>	It doesn't use access modifiers	It uses access modifiers such as: Public, Private, Static, etc...	It doesn't use access modifiers
<b>Flow of Code</b>	It follows a top-down approach	It follows a bottom-up approach	It is controlled by events
<b>GUI, JFrames, JButtons, JTextFields, etc...</b>	Not supported	Not supported	Supported
<b>Encapsulation, Inheritance, Polymorphism, Objects, Classes, etc...</b>	Not supported	Supported	Not Supported
<b>Size of the code</b>	I broke down the code to functions and methods in the same class	I broke down the a large code to smaller codes by putting them in classes	The code can't be broken down into anything as it either comes with OOP or with Procedural
<b>Code Reusability</b>	Not reusable	Reusable	Not reusable
<b>Code Expansion</b>	It is harder to expand and add functions and variables	It is easier to expand by adding classes or objects	It is harder to expand because Event-Driven consists of many parts such as JFrames, JButtons, Listeners, etc... that require a lot of work
<b>Data Hiding</b>	Not supported	Supported through the use of encapsulation	Not supported

#### 2.4. Critical Evaluation of the code samples in relation to their structure and characteristics:

I think that the code samples are structured in a very well manner as there is very minimal unnecessary duplication of any lines of code and that the code is easy to trace and debug because the code is fully commented. Also, each class consists of fully encapsulated attributes, getters and setter, constructors, and various methods.

**These code samples meet the characteristics of their used paradigms:**

### **1- Procedural Programming Paradigm:**

As mentioned before, the Procedural Paradigm follows a top-down approach and so does the code of the Guess The Number Game, and some parts of the codes of the school system. The program can be broken down into functions that may include parameter passing which is another characteristic of the procedural paradigm.

```
public static int guessTheNumber()
```

Without Parameter  
With Return Value

```
public static void setYear(String year) {
```

With Parameter  
Without Return Value

```
static void createAccount()
```

Without Parameter  
Without Return Value

```
static int calculateNumber(int num)
```

With Parameter  
With Return Value

The codes also uses a number of Predefined Functions such as equals(), println(), etc...

```
if(Administration.getUsers()[0][x].equals(user))
```

Predefined Functions (equals())

The code contains Local Variables

```
public static int guessTheNumber()
{
    System.out.println("\nGuess the Number has started");
    int secretNumber = 0;
    int numOfAttempts = 0;
    int option1 = 0;
    int option2 = 0;
    int large = 10000;
    int small = 1;
    int mid;
```

Local Variables

The code contains Global Variables

```
1 package paradigms;
2 import java.util.Scanner;
3 public class procedural {
4
5     static int numOfAttempts = 0;
6
7     public static void main(String[] args) {
8         int option = 0;
9     }
```

Example of Global Variable  
numOfAttempts

## 2- Object-Oriented Programming Paradigm:

The code includes all 3 characteristics of OOP which are: Encapsulation, Polymorphism and Inheritance.

**Encapsulation:** all attributes in this code are fully encapsulated by the use of getters and setters:

```
//attributes
private static String name;
private static String gender;
private static String nationalID;
private static String dateOfBirth;
private static String placeOfBirth;
```

Setting attributes to private

```
public static String getName() {
    return name;
}
public static void setName(String name) {
    record[0][nameCount] = name;
    nameCount++;
    People.name = name;
}
public static String getGender() {
    return gender;
}
public static void setGender(String gender) {
    record[1][genderCount] = gender;
    genderCount++;
    People.gender = gender;
}
public static String getNationalID() {
    return nationalID;
}
public static void setNationalID(String nationalID) {
    record[2][nationalIDCount] = nationalID;
    nationalIDCount++;
    People.nationalID = nationalID;
}
public static String getDateOfBirth() {
    return dateOfBirth;
}
```

Getters and Setters

**Polymorphism:** the codes includes both overloading and overriding examples.

```
static int calculateNumber()
{
    int num = 0;
    while(record[6][num] != null)
    {
        if(MainFrame.user == record[6][num])
        {
            break;
        }
        num++;
    }
    return num;
}

static int calculateNumber(int num)
{
    while(record[6][num] != null)
    {
        if(MainFrame.user == record[6][num])
        {
            break;
        }
        num++;
    }
    return num;
}
```

Overloading

```
Employee.java X MainFrame.java X People.java X Administrati...
57 //adminPage method
58 static void adminPage()
59 {
60     calculateNumber(0);
61     setNumberCopy(getNumber());
62     final JFrame frame4 = new JFrame();
63     frame4.setVisible(true);
64     frame4.setLayout(null);
65     frame4.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
66     frame4.setSize(750, 750);
67
68     //Personal Profile
69     JButton btn1 = new JButton("Personal Profile");
70     frame4.add(btn1);
71
Teacher.java X
234 //adminPage method
235 static void adminPage()
236 {
237     calculateNumber(0);
238     setNumberCopy(getNumber());
239     final JFrame frame4 = new JFrame();
240     frame4.setVisible(true);
241     frame4.setLayout(null);
242     frame4.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
243     frame4.setSize(750, 750);
244
245     //Personal Profile
246     JButton btn1 = new JButton("Personal Profile");
247     frame4.add(btn1);
```

Parent Class

Child Class

Overriding

**Inheritance:** inheritance is used in these classes, as class “Administration” inherits from class “Teacher”, and class “Teacher” inherits from class “Employee”, and both classes “Employee” and “Student” inherit from the class “People”.

```
public class Employee extends People{
public class Teacher extends Employee{
public class Administration extends Teacher {
public class Student extends People{
```

### 3- Event-Driven Paradigm:

The two main characteristics of the event-driven paradigm that I covered in my codes are trigger functions and events.

```
textField.addMouseListener(new MouseListener() {  
    public void mouseReleased(MouseEvent e) {  
    }  
    public void mousePressed(MouseEvent e) {  
    }  
    public void mouseExited(MouseEvent e) {  
    }  
    public void mouseEntered(MouseEvent e) {  
    }  
    public void mouseClicked(MouseEvent e) {  
        textField.setEnabled(true);  
        textField.setText(null);  
    }  
});
```

```
textField.addKeyListener(new KeyListener() {  
    public void keyTyped(KeyEvent e) {  
    }  
    public void keyReleased(KeyEvent e) {  
    }  
    public void keyPressed(KeyEvent e) {  
        if(e.getKeyCode() == 10)  
        {  
            setNumber(0);  
            while(getRecord()[0][getNumber()] != null)  
            {  
                if(textField.getText().equals(getRecord()[0][getNumber()]))  
                {  
                    personalProfile();  
                    frame1.hide();  
                    break;  
                }  
                setNumber(getNumber() + 1);  
            }  
        }  
    }  
});
```

```
btn1.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        frame1.hide();  
        personalProfile();  
    }  
});
```

Examples of listeners that listen for events to trigger a function

Also, all of these codes follow all of the coding standards that I will be mentioned below, and the codes went through all of the debugging process, and it also has undergone rigorous testing by myself, many friends and family members.

## **References:**

- Bhatia, S. (2021) *What is Procedural Programming? [Definition] - Key Features*, hackr.io. Available at: <https://hackr.io/blog/procedural-programming>
- Doherty, E. (2020) *What is object-oriented programming? OOP explained in depth*, educative.io. Available at: <https://www.educative.io/blog/object-oriented-programming>
- Hartman, J. (2022) *What is Abstraction in OOPs? Java Abstract Class & Method*, Guru99. Available at: <https://www.guru99.com/java-data-abstraction.html>
- Herrity, K. (2021) *What Is Object-Oriented Programming? The Four Basic Concepts of OOP | Indeed.com*, Indeed.com. Available at: <https://www.indeed.com/career-advice/career-development/what-is-object-oriented-programming>
- Indicative (2021) *What Is A Programming Paradigm? Data Defined*, Indicative. Available at: <https://www.indicative.com/resource/programming-paradigm/>
- Jamshidi, N. (2021) *characteristic of procedural programming - Perfect description*, solsarin. Available at: <https://solsarin.com/characteristic-of-procedural-programming/>
- Janssen, T. (2017) *OOP Concepts for Beginners: What is Polymorphism – Stackify*, Stackify. Available at: <https://stackify.com/oop-concept-polymorphism/>
- Kolesnikova, T. (2022) *Event-Driven Programming - Applications & Features*, StudyBay. Available at: <https://studybay.com/blog/event-driven-development-features/>
- Lerdorf, R. (2017) *Object-Oriented Terminology*, University of Minnesota Duluth. Available at: <https://www.d.umn.edu/~gshute/softeng/object-oriented.html>
- Lewis, S. (2020) *What is object-oriented programming (OOP)?*, TechTarget - Search App Architecture2. Available at: <https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP>
- Thanoshan, M. . (2019) *What exactly is a programming paradigm?*, freeCodeCamp. Available at: <https://www.freecodecamp.org/news/what-exactly-is-a-programming-paradigm/>
- Tucker, A. B. and Noonan, R. E. (2004) *Event-driven programming*, *Computer Science Handbook, Second Edition*. doi: 10.1201/b15723-8.
- w3schools (2021) *Java Encapsulation and Getters and Setters*, w3schools. Available at: [https://www.w3schools.com/java/java\\_encapsulation.asp](https://www.w3schools.com/java/java_encapsulation.asp)



## *Part 3*

### **3.1. The use of Eclipse IDE in the project.**

### **3.2. Evaluation of the use of an IDE for the development of applications contrasted with not using an IDE:**

**IDE (Integrated Development Environment):** it is software for building computer programs that allow programmers to combine the various parts of building a computer program into one location which is the IDE, such as debugging, altering source code, creating executables, and many others. (Lynda, 2018; Codecademy, 2021)

#### **1- Using an IDE for the development of applications: (Codecademy, 2021)**

- a) IDEs provide syntax highlighting, in which it colors each part of the code for it to be easily read by the programmer, in which each color has its meaning.
- b) By pressing the “Ctrl + Space” buttons it enables the autocomplete on the IDE which saves a lot of time and prevents a lot of the mistakes that might occur if the programmer wrote the code himself instead of auto-completing it.
- c) Some programming languages (like Java) need to have their code compiled into an executable file before running the code (compiled from .java to .class), therefore, IDEs also build your executables by default when you press the run button.
- d) IDEs provide programmers with a toolbox of debugging tools for them to use such as breakpoints, debugger, and many other tools that will be discussed later in the report.
- e) IDEs notify the programmer whenever a syntax error occurs while writing the code for them to be able to fix it before compiling and running the code.

#### **2- Not using an IDE for the development of applications (using a regular software for building computer programs):**

- a) They don't normally offer syntax highlighting, therefore, all of the text will be in one color, such as black, purple, etc...
- b) They don't offer an autocomplete option, which forces the programmers to write the whole code themselves, therefore, the chances of human errors in the code will increase drastically, and the number of syntax errors will also increase.
- c) They don't build your executable file by default when you run the code, therefore, the programmer needs to compile the code themselves before running the code which will cost the programmer more time.
- d) They don't usually provide breakpoint or debuggers, therefore, the programmer has to debug their code using traditional methods of debugging such as using the print function, divide and conquer method, etc...
- e) They don't normally notify the programmer of any errors before compiling or even before running the code.

### 3.3. Debugging process:

**Debugging:** it is the process of preventing, detecting, and removing any logical, syntax, or runtime errors (bugs) that may occur in your code, that could cause the code to suddenly crash, not compile at all, or behave in an unexpected manner. (Bennett et al, 2021)

#### **The Debugging Process:**

- a) **Define the problem:** in this step, I tried to do my best to identify the root cause of the error to be able to fix it. Paying attention to error messages if the error was a syntax or runtime error helped identify the problem.
- b) **Make use of Existing Debugging Tools and Techniques:** tools such as println, techniques such as divide and conquer, and the use of breakpoints were really helpful in identifying the source of your bug.
- c) **Write down everything:** I liked to write down everything that I have done to try to identify or remove the bug so that I don't waste my time repeating things that I have already tried out.
- d) **Localize the problem:** trying to localize the problem to a certain scope or a certain method can make things a lot faster with the process of debugging.
- e) **Try to replicate the problem:** when debugging, I used to try to go over each step that I did to try to replicate the problem that I caused, firstly to try to fix the existing problem at hand, and to learn from my mistakes for the future.
- f) **Turn to other programmers for help:** luckily I didn't have to turn to other programmers for help yet as I didn't encounter any bug that I wasn't able to identify and resolve on my own, but I'm sure that someday I will need to ask for the help of other programmers.
- g) **Remove your Bug:** after following all of these steps I usually will have been able to identify my bug, and then I have to find a way to remove it in a way that doesn't affect the rest of my code.
- h) **Test your code:** after removing the bug I always do rigorous testing on my code to check that everything is functioning the way that it is supposed to.

#### **Debugging facilities available in IDE:**

IDE provides a lot of debugging facilities that make that process much easier and faster.

- a) **It provides the programmer with a list of errors and warnings:** IDE has a feature that helps programmers detect and fix syntax errors by showing them exactly where the error or the warning is and proposing to the programmer a list of solutions to fix it.
- b) **Debugger:** it enables the coder to set a breakpoint within their code to stop the execution of the code until the programmer has examined the line of code that has the breakpoint and examined the variables, and then the coder chooses either to step into, step over, step return, resume, etc...

- c) **Print:** using the printing functions is a very powerful tool in debugging that can help you identify the exact location of your bug by continuing to print your variable throughout different parts of your code until the bug is detected. The print debugging method isn't provided by the IDE rather it can be used on any software development program as it comes with the language itself.
- d) **Divide and Concur:** this is done by making use of comments and print statements, by making the lines that you suspect are the problem comments, and using print statements at the same time. Much like the print method, the Divide and Concur method isn't provided by the IDE rather it can be used on any software development program as it comes with the language itself.
- e) **Try and Catch:** it is a method that exists in many programming languages and it doesn't come with the IDE. What it does is that it tests the code that is in the scope of the "try", and checks for the exception that is specified in the "catch". If it is detected the exception will be printed in the console.

### 3.4. How can the Debugging Process be used to Help Develop More Secure, Robust Applications:



As previously stated, debugging is an essential part of the coding process since it eliminates any syntax, logical, or runtime errors in your code, making it foolproof and ready for any input from the user, even if the input is unexpected.

Debugging and debuggers allow users to gain insight into the logic of the application, therefore, if someone who has enough programming experience wants to hack an application, they can easily employ debuggers to accomplish their objectives, especially if the code of the application has any loopholes or errors that haven't been detected by the programmer or the debugger, that the hacker can use to access the application. (Cybersecurity ASEE, 2022)

When a person with malicious intents, the correct tools, and sufficient knowledge of coding decides to tamper with an app, things can quickly spiral out of control as they could use reverse engineering to deconstruct a piece of software and then thoroughly analyze its logic to completely comprehend its behavior, which will help the hacker to tamper with the code, to breach the data or do individual attacks. (Cybersecurity ASEE, 2022)

All of this may be avoided by thoroughly debugging the code, rigorously testing it, and ensuring that no vulnerabilities exist that could threaten the application's security. Finally, if the debugging procedure is implemented correctly, your program will be incredibly reliable, secure, and robust, with very little risk of security breaches.

### 3.5. Coding standards and their importance:

- 1- **Using proper naming conventions:** it refers to the name that the coder gives to classes, packages, methods, variables, etc... Using the camelCase or the PascalCase are examples of the naming conventions that I used while writing my codes.
  - a) **Classes:** they follow the PascalCase naming convention, e.g. MainFrame.java  
class.  MainFrame.java X
  - b) **Methods:** they follow the camelCase naming convention, e.g. adminPage ()  
method. `static void adminPage()`
  - c) **Variables:** they follow the camelCase naming convention, e.g. numOfDigits  
variable. `int numOfDigits = 0;`
  - d) **Packages:** they follow the camelCase naming convention, e.g. algorithm package.  
>  algorithm

- 2- **Fully encapsulate class attributes:** to set all class attributes' access modifiers to "private" and to fully encapsulate them with the use of getters and setters.

```
private String year;

public String getYear() {
    return year;
}

public void setYear(String year) {
    People.setDepartment("Student");
    People.record[9][People.nameCount - 1] = year;
    this.year = year;
}
```

- 3- **For the code to be properly commented:** comments are used for the code to be easier to read, especially when teams of programmers work on a code, because the code might be read by a variety of people with diverse levels of java experience.
- 4- **Initialize all variables and objects along with their declaration:** the best practice in java is to immediately initialize any variable and objects as soon as it is declared either by using constructors or by giving the variable a value, to avoid any unnecessary runtime errors.  
`Administration admin1 = new Administration("21110011", "marwan321");`
- 5- **To make use of Indentations:** Firstly, it makes your code much easier to understand and trace by anyone who will view your code. Secondly, it makes your code look much more professional, concise, and beautiful.

### Importance of Coding Standards Individually:

- 1- **Correction of Bugs:** following the coding standards helps by making the process of identifying and correcting bugs and errors in the code much easier.

- 2- **Makes searching for pieces of the code easier:** by the use of comments in your code, I was able to use “Ctrl + F” to find one line of code in a class that contains hundreds of lines of codes in a matter of seconds.
- 3- **To be able to identify if a name is the name of a class, package, method, or variable by just checking if it follows the camelCase or the PascalCase.**
- 4- **Makes your work look more professional and concise.**

#### **Importance of Coding Standards as a Team:** (Multidots, 2020)

- 1- **Enhanced efficiency:** Programming standards aid programming teams in detecting problems early on, if not totally preventing them, which will boost productivity of the team during the process of software development which will also reduce the risk of the project failing.
- 2- **Minimal Complexity:** when a code is complex, it is more likely to have bugs in it, therefore, coding standards assist in the construction of less complex software programs, which reduces errors.
- 3- **Easy to Maintain:** if programming standards are followed, the code will be concise and easy to maintain and understand which will allow other coders to improve on the code much easier.

#### **References:**

Bennett et al (2021) *What is Debugging? Definition of Debugging, Debugging Meaning, The Economic Times*. Available at: <https://economictimes.indiatimes.com/definition/debugging>

Codecademy (2021) *What Is an IDE? | Codecademy, Codecademy*. Available at: <https://www.codecademy.com/articles/what-is-an-ide>

Cybersecurity ASEE (2022) *What is debugging and how do hackers use debuggers to compromise your mobile application?*, Cybersecurity ASEE. Available at: <https://cybersecurity.asee.co/blog/what-is-debugging/>

Lynda (2018) *What is an IDE?*, LinkedIn Corporation. Available at: <https://www.lynda.com/Development-Tools-tutorials/What-IDE/486760/505357-4.html>

Multidots (2020) *Importance of Code Quality and Coding Standard in Software Development, Multidots*. Available at: <https://www.multidots.com/importance-of-code-quality-and-coding-standard-in-software-development/>