**Part 1**

**1.1the steps that involved in the process of building an application:**

First you got to imagine your overall picture, you need to know what you want to do, don’t just start, and get into coding from the first step; applications aren’t easy to make. Here are some steps that you may take to make an application:

1. Study the program that you are going to code: that will help you to know what you really need to execute your ideas
2. Start to Imaging and design your application in your mind: simulating the overall pictures will help you to take your first step into your project journey.
3. Execute your plans and ideas into a pseudo code, and how you are going to design your code, so you know how your code will look like, doing a pseudo code will help you to choose the most suitable programming language that fit your ideas.
4. Generate your work from pseudo code to a flow chart, so you start to imagine how you are going to code your application.
5. Start to write you application code, be careful to execute all your ideas in right methods and paradigms, don’t forget being wise with your codes, don’t just code to finish, try to find the best and most effective way to code with and use your code standards for better quality.
6. Separate your code into levels, start to finish one by one until you get the overall result.
7. Debug your code and try it; to discover any possible errors: syntax errors, logical and run-time errors, by that you can avoid the gaps as much as you can.
8. Since you are going to design an application, how about to test it! Testing your application between two users- User interface test -will able you to see where you have problems and gaps. Don’t forget to test your app performance and usability. Also Check out the security and where do you have any glitches.
9. keep tracking the people who are going to use it, because their feedback, will be your key to fix any other problems, after you locally upload your application.

**1.2 what is the algorithm and the characteristics of a good algorithm**

**Algorithm:** is the specific process to transfer your what you are going to do into commands, to explain your problem from the input and express until the output to reach out the solution. We do use algorithms in our daily life, not just in the field of computing.

**How does a good algorithm look like (characteristics)?**

1. Defining an input and output in a proper way.
2. Unambiguity: your instructions and commands should be as clear as possible.
3. Finiteness: you can’t write an infinity algorithm; your command should be counted.
4. Feasible: make sure to write your commands as simple as possible and make it easy to read and understand.
5. Autonomous language: your command must be applicable in every coding language, which is mean that you need to write an independent algorithm which can be coded by any language.

**1.3- An algorithm that converts a binary number into decimal and converts the decimal into digital.**

1. My mission is to create a program to convert any binary number into it decimal form, then into a digital form.
2. Starting with a binary to a decimal form, first thing we must understand is the process of transforming from a binary to a decimal: the user will input a binary number and get a decimal one; so, we must create a binary and decimal variable. There are another two variables need to be considered, which is the remainder and the power of two.
3. Make two ways for the input: if the input is zero then the decimal number is zero too. But if the input isn’t zero, we need to get the remainder after we divide the binary by ten, we equalize the decimal variable with the decimal and the remainder\*power. Moreover, equalize the binary variable with the binary value divided by ten. After that multiply it by two raised to the zero power.
4. Keep repeating this until the binary values is zero, and the decimal number is executed.
5. Talking about the digital form, you will recognize that your number might be upside down, you need to reverse it. Start with defining the values and the remainder of them, we start by dividing one of the variables by ten and take the remainder out of it and equal it with the number values divided by ten and store it in the remainder, after that equalize the reverse number variable to the previous values and multiply it by ten and add the remainder to it, keep repeating this process until the decimal value is less than one.
6. To print the digital form, first thing we need to know is that how many digits inside the decimal number, make a loop to count that, starting from number variable equal to number one and less than the decimal number, then multiply the number by ten and store and put it inside the loop again, keep doing that until you the variable is bigger than the decimal number. You will get the number of digits after that.
7. Compare the digits to with the cases, where each digit should go into a one case, execute your cases by print statements that simulate the digital form.

**My algorithm in short words was like that:**

* **Create a method that convert binary into decimal:**

Start method

Divide the binary number by 10 and store the remainder into temporary variable.

Decimal num = decimal num + temp \* base(“pow”)

Divide the binary number by 10

Mutably the base(“pow”) by two

Return decimal

End method

* **Convert the decimal into digital form:**

Find a way to count the digits of the decimal number

Put the number of digits into array

Put that array into switch statement and start of the cases.

Put cases from 0 to 9.

Every case should print a digital number out of it.

**Describe the algorithm steps in pseudo code:**

**Start method**

**Declare some variables: decimal, power, temporary(remainder)**

**While binary= zero (true), break and end the loop**

**else**

**temporary = binary%10**

**decimal = decimal + temporary\*power**

**binary = last value of binary/10**

**power =last value of the power\*2**

**end the loop**

**print “Loop result”**

* **[Second mission]**

**Declare some variables Num, digit counter, reverse number, array for the digit counter**

**int digit**

**int cont = 0**

**loop to count the digit (i=1;i<=Number1; i=i\*10)**

**counter ++**

**a loop to reverse the digits ( i=1;i<=Number1;i=i\*10)**

**Counter - -**

**degt=Number%10**

**arr[Counter]=degt**

**Number=Number/10**

**arr[Counter]=degt**

**loop to go after the digits ( i=0;i<Counter2;i++)**

**case name of**

**a statement to go with condition (switch /while or others) \\note: (\n) refer to a new line**

**“0”: Output print (" \_\_\n| |\n|\_\_|\n\n")**

**“1”: Output print ("|\n|\n\n")**

**“2”: Output print (" \_\_\n \_\_|\n|\_\_\n\n")**

**“3”: Output print ("\_\_\n\_\_|\n\_\_|\n\n")**

**“4”: Output print ("|\_\_|\n |\n\n")**

**“5”: Output print (" \_\_\n|\_\_\n \_\_|\n\n")**

**“6”: Output print (" \_\_\n|\_\_\n|\_\_|\n\n")**

**“7”: Output print ("\_\_\n |\n |\n\n")**

**“8”: Output print (" \_\_\n|\_\_|\n|\_\_|\n\n")**

**“9”: Output print (" \_\_\n|\_\_|\n \_\_|\n\n")**

**cont++**

**end**

**1.4 The code will be submitted on the GitHub:**

**1.5Evaluation of the algorithm and the relationship between the written algorithm and the implemented code:**

Writing an algorithm before you start to code will give you a better experience when you decide to write any program, algorithms is like a map that you can follow to reach your destination, when you write your own algorithm to, you will be fully aware of the problems and mistakes that you could make, so you will avoid them, and you will save time and effort.

The relationship between the code and the algorithm is like the relationship between the couch and the player, the algorithm will couch you to write a better code. So, writing algorithm isn’t something essential, but it will able you to have a great flow and overall picture of how your work is going to be.

**Part 2**

**2.1Programming paradigms:** programming paradigm is your way to express your code, you define what paradigm is most suitable to be used: we have different types of main paradigms: imperative and declarative programing paradigm. Talking about imperative paradigms, we have procedural paradigm, Object oriented paradigm and event driven paradigm. Each paradigm has different characteristics, ways, and functionalities.

**Procedural paradigm characteristics:**

1. Goes from up to down -up to down approach- which is mean that it uses a technique to divide complex codes into smaller pieces and functions; this technique made the procedural programming flexible to apply.
2. Often used in basic languages like: C, COBL, BASIC, FORTRAN.
3. Contain two basics component: function and procedure.
4. Easy to learn, code and but not to debug.
5. Straight forward.
6. Can’t simulate real word event.

**Object oriented paradigm characteristics:**

1. Classes and objects: classes are the main thing in OOP, classes offer you a great place to put your objects in.
2. Encapsulation (getters and setters): when you want to play with your code, you should encapsulate it by making your variables and methods privately.
3. Abstraction, divide your data into public or private.
4. Inheritance: you can make your classes share the attributes and methods.
5. ability to simulate real-world events much effectively
6. It is easy to develop and maintain the code
7. Security and safe to apply.
8. Polymorphism, you can do the same action using different method and parameter by overload and do the opposite of that by override.

**Event driven paradigm characteristics:**

1. Event: any type of action that need to happen in the process of execution, such as: mouse right/left click, drag, scrolling, etc.…
2. There is a lot of types of interactions whether we are talking about a mouse, keyboard, touch screen or even an operation system action: like a message or a task.
3. Trigger: the triggers are as same as listeners; they keep listening and detecting if any event happens.
4. Event handler: this process is about control the event and action, it is start after the listener detect an action, and there will be a code to run in order to response, then the result came in the page appearance. Event handler is the guy who is responsible to handle those action and make sure to happen.
5. Event loop: event loop will get you back to start another event over and over.
6. Graphics UI: user interface done with event driven paradigm, the graphics user interface can be divided into classes, which prove the point of that Event driven can be used along with other paradigms.
7. Event listener: is a detector which is responsible to keep listening to any type of actions, and when he detects an action, the code will start moving on and the event handler will handle everything.

**The relationships between the paradigms:**

All three paradigms are depending on each other’s as I have mentioned before, in the event driven paradigms we do use a lot of object-oriented classes; and that is to make our code simpler and easier to handle, we use oops a lot in the graphic user interface. Let’s not forget that also the event driven paradigm depends on the procedural paradigm; before we invented OOP, we used to use the event handler as a subroutine with a procedural program. In conclusion the event driven is depending on both of procedural and object-oriented paradigms.

**2.2 Examples on the Programming Paradigms (GitHub)**

**2.3Compare and contrast the paradigms.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type:** | **characteristics** | **Usable** | **Security** | **Programming languages:** | **approach** |
| Procedural | 1. Divided into functions.  2.Deal with commands written in  3. module.  The focus is on ideas and data. | 1.Low productivity.  2.Doesn’t simulate real life problem.  2.Usable for beginners. | 1.Less secure.  Function shares a global data.  2.No access specifiers. | 1.C.  2.Pascal.  3.Basic.  4.Fortran.  5.Cobol. | Top-down approach. |
| OOP | 1.Divided into objects.  2.Deals with character User interface.  3. A Class can inherit attribute from another class. | 1.Simulate real life problem, and perfectly deal with them.  2. simple and easy to deal with.  3.high productivity. | 1.More secured; by data hiding use.  2.There are access specifiers. | 1.C++  2.Java  3.Oracle | Bottom-Up Approach. |
| Event Driven | 1.Deals with Graphical User Interface to develop your program.  2.Straight Connection between the user and computer.  3.Events are the ones who decides what happen next.  4.Have two types of events:  4.1One that start the event  4.2One to react to the event. | 1. Used to preform actions and handling the events.  2. can perform many actions.  3. the best way to execute a perfect Graphical User Interface. | 1.As secured as the OOP; Event-Driven uses the OOP.  2.there are access specifiers. | 1.Java  2.C++  3.Visual Basic  Note: most of OOP languages support Event-Driven Paradigm. | Can be handles by the user; by action and events. |

**2.4 evaluation of the code samples, in relation to their structure and the unique characteristics.**

I have written three code samples; each one represents a programming paradigm or two. In the first code sample, it was about a program that tell you whether your able to get your driving license Based on your age. In this sample I have used two functions, the first one to calculate your age, and the second is the main function that the user put his age in. Using only function made my code goes under procedural oriented program, it was a little bit hard to debug this code and deal with, especially because you deal with a top-down approach, that restrict you to program in a specific way. But it is still a good experience, because you can learn the basics of the programming language that you code with.

After that I have written the second sample, it was about a race between three drivers, and I need to get the first and second name of each driver, Age, Country, rank, and prize. So, I have started with creating a main class and created the variables names in it, then I have created another class and name it as “Driver”, I have defined the full name of each driver, the age, and the country, created an instructor for them, and used the method of getter and sitters. I have created another class and named it as “Title Winner”, I have made sure that he extends driver, so he inherits from him, I have created an default instructor and another instructor to initialize my objects and methods to set and get them. Finally, I have created my objects in main class, and fill the variables, and printed each driver: Rank, Prize, Full Name, Age and Country. The OOP helped me to organize my work and make it even more secured, I was able to apply the OOP in a good way, I have used to the methods and the classes inheritance in a proper way too, the classes and objects helped me to organize my work better that the procedural sample, it was also easier to debug the code and handle it.

Last Sample was about an event that happen after the user click on a button, I have made a program that simulate an online rosary” Subha” or “سبحة” so every click on the bottom is counted, I have started by doing the window sittings- size and visibility – then initialize the button and the text place, after that I have added a constructor and added window and action listeners. The event driven helped me to handle the events and show the results on the Graphical User interface, the overall experience was not bad, I wish that I was more experiences in this paradigm, I will make sure to keep studying and learning more on the event driven paradigms.

* Note: all the templets will be submitted on the GitHub.

**Part 3**

**3.1 Use the IDE to manage the development process of the above code implementations (GitHub)**

**3.2IDE evaluation comparing with no IDE:**

I have used Eclipse as my main IDE, I have had a great IDE experience, that was by saving a lot of time especially in the coding process; because of the ide shortcuts and resources that it offers. IDE offered me a better experience in my debugging process; especially when I have syntax and logical errors, eclipse was giving me the specific line and problem that I need to fix, it even offered me a good suggestion to some lines. When I was not using ide, I was facing a lot of issue in running the code, I have used to execute my code in the terminal, but there were some restrictions there, such as that it was giving me a lot of run-time and mystery error’s; but in ide it is a way much better; IDE was giving me the specific error, type, and place. Another thing I want to mention, that using an IDE offered me to have a lot of advantages than without it; that is because of project management that ide offered to me, I wasn’t able to separate my code into classes without ide, neither that I was able to have a fully event driven experience without it. Starting to use an IDE was a little bit weird, because I am not used to have a machine to help writing my code, I was used to implement everything by myself, even in the run process, I used to run my codes by command in terminal, that why using an IDE was a big step for me; because I have saved a lot effort and time to do what I like, which is coding.

In conclusion, I advise every programmer to start use any type of an IDE, it will give you better performance, results, and experience. IDE will even make you more professional and will save your time and effort.

**3.3My debugging processes:**

Debugging and testing took the most time in the process of developing the code. However, I have started my debugging process by checking the syntax error’s, I remember that I have faced some errors in variables and method names, some in the brackets and curly braces, and some in the semi colon. But the IDE was helping me to catch them, so that was a kind of an easy process to do. In term of logical errors, I have faced some of them, as an example: in the digital form I was getting the number reversed, so I have debugged my code line by line and I wasn’t able to detect the error, then I tried to use break points, but it wasn’t effective, so I have reprogrammed the code. Then I have checked whether I have any run-time errors, I were having one; because I was dividing on zero, I have used “try and catch” and print statements to catch the error.

**3.3The debugging facilities that available in the IDE:**

1. **Divide your code into half pieces:** this debugging process isn’t that much worthy to try, but sometimes It might be usable, especially when you have a huge code, and you can’t break it into break points so you can use it to get a closer image on your logical error.
2. **Print statements:** this could be usable in logical and run-time errors; you can use it to check the output of your variables.
3. **Breakpoints:** after you have detected your error area, you can use punch of breakpoints to see where exactly the error is, the code will keep moving until he finds an error in a breakpoint, he will pause.

Note: you can use f-5 and f-6 as shortcuts for step into and step over.

**3.4developing more secure and robust by the debugging process:**

debugging is your way to develop your application and secure it from different hacking attempts and glitches, let me start by example: when I was a little boy, I used to hack my games through an app called “lucky patcher”, after years, I have asked myself, how the app was doing that? it was all about that the app was finding the gaps and the bugs inside the game code, and exploit it to do whatever he want to do.

Many applications get hacked just by a little and unrecognized bug, most of those bugs being toward logical errors. Here is some suggests about doing better debug toward the app secure and robust:

1. Be sharp with your debugging process: being smart and sharp in the process of debugging is so important; you need to leave no way for any debug or error. You also need to be more indigent and aware from anybody who trying to hack your application.
2. Do a debugging routine: don’t stop debugging! Put a continuous plan to keep debugging your application every scheduled amount of time, doing this step will make you a step close to a “perfect application”, where your application is secured and robustness.
3. Debug the UI: debugging the user interface will give you the customers point of view, where you can find any gaps, and weaknesses of the application.

**3.5Outline my coding standard’s:**

Variables: Pascal

Package name: all small letters

Class name: Pascal

Method name: Pascal

No more than one statement per line.

maximum number of lines is 100

Name of creator.

Constructer: Pascal

Use comments from time to time.

**3.5Critically evaluate why coding standards was necessary in my work specifically and in team works generally:**

Coding standards had helped me to organize my work, when I started to code, I wasn’t fully aware of having specific standards, because I wasn’t dealing with that much of problems. But lately I had been obligated to use standards and I was a little bit not comfortable with them at the start, but then I used to them, and I have realized what I was missing by not using personal standards. Coding standards had helped to easily deal with my code, I recognize myself that I started to code in more quality and even quantity, I also have been having less syntax errors, less logical and run time errors, so my code had become more maintainable. Another thing that standards had done to my codes, is that they become more readable and easier to debug; that was because of the standard of naming.

Talking about standards in teamwork, to be honest I have never worked with a programmer’s team. But I think that I have some ideas on what I should do when I start to work with my teammates.

I will be using a lot of comment, especially on the complex lines, so they don’t get lost or miss anything. I will also put my name as comment at the last line of the program so my teammates be aware that I wrote the code; so, they play with my rules and standards. I recommend that every team should have a constant standard between them so they can avoid the conflict that come from standards. I will also not write more than 100 lines inside one class, that will help the team to avoid the complexity and keep their work area clean.

Reference list

Claudia (2020). *How to Write a Program: Coding, Testing & Debugging - Video & Lesson Transcript*. [online] Study.com. Available at: https://study.com/academy/lesson/how-to-write-a-program-coding-testing-debugging.html.

Expert Wannabe. (2010). *Advantages and Disadvantages of Using IDE*. [online] Available at: https://salfarisi25.wordpress.com/2010/12/22/advantage-and-disadvantage-of-using-ide/.

https://codeahoy.com/about (2016). *Effective Coding Standards*. [online] CodeAhoy. Available at: <https://codeahoy.com/2016/05/22/effective-coding-standards/>.

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