

**SAINT VINCENT COLLEGE OF CABUYAO**

***Bachelor of Science in Information Technology***

**LABORATORY MANUAL**

**PF101 – Object-Oriented Programming**

**Laboratory Exercise No. 1**

***Introduction to Object-Oriented Programming***

***(with Review of Programming II)***

Submitted by:

|  |
| --- |
| **Magno, Lancelei C.** |

[Name]

|  |
| --- |
| **2A2** |

[Section]

Submitted to:

|  |
| --- |
| **Prof. Apollo Neil R. Duran** |

[Name of Instructor/Professor]

|  |
| --- |
|  |

**GRADE**

|  |
| --- |
| **AUG 2024** |

[Date]



***Laboratory Exercise No. 1***

**Introduction to Object-Oriented Programming**

**I. OBJECTIVES**

|  |
| --- |
| **At the end of the exercise, the students are expected to:**   * **Familiarize with the object-oriented programming approach** * **Construct a modular program applying standard control structure** * **Develop a solution to the given problems using Arrays and in a modular**   **program.** |

**II. EQUIPMENT/ MATERIALS**

|  |
| --- |
| **The following equipment or materials will be needed to perform the laboratory exercise:**   * **PC with Java Compiler and IDE (Eclipse, NetBeans, jGrasp, etc.)** * **Internet Connection for Online Java Compiler/Editor and Submission** * **USB for backup and file storage** |

**III. PROCEDURE/ DISCUSSION**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Laboratory Work No. 1 Research and select one existing Procedural-Oriented and one existing Object-oriented Programming Language (except Java). Explain and discuss the characteristics of each selected programming language considering the areas to be examined as presented in the given table below:   |  |  |  | | --- | --- | --- | | **No.** | **PROCEDURAL PROGRAMMING** | **C** | | 1. | In procedural programming, program is divided into small parts called functions. | Due to the nature of procedural programming being a top-down approach, the larger problem is broken down into smaller subproblems, and an alternate method is used to solve each subproblem. | | 2. | There is no access specifier in procedural programming. | There is no limit on accessing the data within the program. Anything can be manipulated. | | 3. | Adding new data and function is not easy. | The data and the function are separate meaning that manipulating those will be tedious. | | 4. | Procedural programming does not have any proper way for hiding data so it is less secure. | Code implemented is on plain sight making it less secure. | | 5. | In procedural programming, overloading is not possible. | The ability to name a function or method with the same name is not possible though it might be helpful is a way you may not confuse one function from another but managing it will be tedious. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **No.** | **OBJECT-ORIENTED PROGRAMMING** | **GDScript (Godot built-in prog lang)** | | 1. | In object-oriented programming, program is divided into small parts called objects. | Analysing, managing, and manipulation of data is much easier. | | 2. | Object-oriented programming has access specifiers like private, public, protected etc. | Accessing of data is limited to prohibit external code from becoming engaged with an object's internal operations. | | 3. | Adding new data and function is easy. | Since data and methods are combined into objects, adding of data and using it functions is much easier compared to Procedural. | | 4. | Object-oriented programming provides data hiding so it is more secure. | Programs with high value data such as personal information can be hidden to keep the security not only the data but also security of the system. | | 5. | Overloading is possible in object-oriented programming. | A single function call can carry out many tasks depending on the context when it is made to an overloaded function, which executes a customized implementation of the function relevant to the call situation. |  Laboratory Work No. 2 Construct a Java code that will provide a computation of both Harmonic and Geometric Series of n - 1.  In mathematics, the Harmonic Series has the sum of reciprocals from positive integers, for example:  harmonic series  On the other hand, the Geometric Series has the sum from a constant ratio between successive terms like:    Save your program as PF101LabExer1-2.java with the given Function Name for each task done using Recursion and format the result with two decimal places only.   1. **Function Name: *harmonicSum()***   **Sample Run:**  Enter a Positive Integer: 7  The Harmonic Sum is: 2.59   1. **Function Name: *geometricSum()***   **Sample Run:**  Enter a Positive Integer: 7  The Geometric Sum is: 1.99 |

|  |
| --- |
| Laboratory Work No. 3 Construct a Java program that will employ a user-inputted Array with the following tasks using a function and either While or For Structure:   1. **Function Name: *inputArray()***  * This will allow the user to input the elements of the array named *“myArray”* based on its length or size he/she entered.  1. **Function Name: *displayArray()***  * This will print the elements of the array done in the *inputArray()* function.  1. **Function Name: *sumArray()***  * This will display the sum or total of all the inputted elements of the array.  1. **Function Name: *averageArray()***  * This will output the average of the elements based on entered length or size of the array  1. **Function Name: o*ddArray()***  * This will display the elements that are odd numbers.  1. **Function Name: e*venArray()***  * This will display the elements that are even numbers.  1. **Function Name: *divbyfiveArray()***  * This will display the elements that are divisible by five.  1. **Function Name: *highestArray()***  * This will display the highest value element within the array.  1. **Function Name: *lowestArray()***  * This will display the lowest value integer within the array.   Then, save your program as PF101LabExer1-3.java.  **Sample Run:**  Enter Size of the Array: 5  Enter the 5 Elements:  Index[0] : 10  Index[1] : 9  Index[2] : 8  Index[3] : 7  Index[4] : 6  The Sum of All Elements is: 40  The Average of the Elements is: 8  The Odd Number/s: 9 7  The Even Number/s: 10 8 6  The Number/s Divisible by Five: 10  The Highest Element: 10  The Lowest Element: 6 |

**IV. DATA REPRESENTATION/ OUTPUT PICTURES**

|  |
| --- |
| **~LABWORK 1 – 2~**      **~LABWORK 1 – 3~** |

**V. RESULTS INTERPRETATION/ OBSERVATION**

|  |
| --- |
| There are significant differences between procedural programming and object-oriented programming. OOP is handy, to be sure, but it also relies on the individual's use case. For the task at hand, one could find procedural programming more appealing.  Within the Lab work, the tasks required by the program are arranged so that they are independent of one another. You can quickly identify the method that is handling the task if there is an issue with it. |

**VI. CONCLUSIONS**

|  |
| --- |
| This work's usage of OOP makes code management and debugging considerably simpler. From my experience, I discovered a bug while attempting to display the array, which is handled by my displayArray() method in the code. I can just look into the method directly and resolve the issue, whatever it may be. |

**VII. STUDENT OUTCOMES ADDRESSED**

|  |
| --- |
| ***(… to fill out by your instructor)*** |

**VIII. APPENDICES**

1. **RUBRICS AND SCORING**

|  |
| --- |
| ***(… kindly refer to rubrics and scoring provided)*** |