

**COLLEGE OF COMPUTING AND ENGINEERING** **(CCE)**



**CCS108 – Object-Oriented Programming**

**Laboratory Exercise No. 7**

***Encapsulation***

Submitted by:

|  |
| --- |
| Yocor, Angelo T |

[Name]

|  |
| --- |
| **2CS-B** |

[Section]

Submitted to:

|  |
| --- |
| **Prof. Terrence A. Lim** |

[Name of Instructor/Professor]

|  |
| --- |
|  |

**GRADE**

|  |
| --- |
| 10/30/22 |

[Date]



***Laboratory Exercise No. 7***

**Encapsulation**



**I. OBJECTIVES**

|  |
| --- |
| **At the end of the exercise, the students are expected to:**   * **Distinguish the difference between the application of the given**   **access modifiers.**   * **Create the class fields and methods applying the different**   **access modifiers**   * **Apply the encapsulation through accessor (getters) and mutator**   **(setters) method.** |

**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 Construct a class named Student to initiate a student profile. The said class must have:  • A string data field named as **studFullName** that designates the student name.  • An integer data field named as **studAge** that designates the student age.  • A double data field named as **studGWA** that designatess the student GWA.  Use the given sample code with partial implementation: |

|  |
| --- |
| public class Student {// define instance or local variables studFullName, studAge, studGWA/\* modifier datatype variable name\*//\* modifier datatype variable name\*//\* modifier datatype variable name\*/public static void main(String[] args) {//create an objectStudent stud1 = new Student();//set name for stud1stud1.studFullName = "Juan Dela Cruz";//set age for stud1stud1.studAge = 24;//set GWA for stud1stud1.studGWA = -3.75;//print the student profileSystem.out.println("Student Full Name:\t" + stud1.studFullName + "\nStudent Age:\t" +stud1.studAge + "\nStudent GWA:\t" + stud1.studGWA);}}Laboratory Work No. 2 In the previous laboratory work the user has set GWA to -3.5 which is illegible value for a student grade. To avoid this problem, there must be a checking of the new GPA value to ensure that would an eligible or acceptable. You should use private attributes or information hiding, setters, and getters method.  Now make all your variables private instead of public and add the following setters and getters:  • A method with a name **setFullName(String studentName)** to store the student profile.  • A method with a name **setAge(int studentAge)** to store the student age.  • A method with a name **setGPA(double studentGWA)** to store the student GWA.  • A method with a name **getFullName()** that returns the name of student or profile.  • A method with a name **getAge()** that returns the student age.  • A method with a name **getGPA()** that returns the student GPA. |

|  |
| --- |
| Use the class applied in the Laboratory Work No. 1 and then integrate the methods you implemented here to set new values instead of using the variables directly. Sample Run:Student Full Name: Juan Dela CruzStudent Age: 24Student GWA: -3.75Laboratory Work No. 3 As you may notice in the Laboratory Work No. 1, you were able to force the user of class Student to apply setters with new values to some variables and the getters in case he/she needs to read them.  In this part, you are going to check the new value inside the setters before assigning the new value for the instance variable. For example, when the new value of the GPA is -3.5 you also check first if the new value of the GPA is > -1. If so, we assign it to the instance variable otherwise we print an error message.  With this event, -3.5 is not > -1 so you will not continue with the assignment. Add the if statement inside the setAge() and setGPA() methods to check if the new value is an eligible or acceptable value or not. If yes, assign it to your instance variable, if not, print an error message and terminate the program.  Lastly, save the entire application with a name or folder CCS108LE7 with package as pnc.laboratory.exercise7. |

**IV. DATA REPRESENTATION / OUTPUT PICTURES**

|  |
| --- |
| Laboratory Work No. 1     **Codiva link**: <https://www.codiva.io/p/ba850058-abd8-421d-985b-2c89017fb686> Laboratory Work No. 2 |
| **Covida link**: <https://www.codiva.io/p/cab4842f-444d-4fc0-93cc-49428c87ee9f> Laboratory Work No. 3 |
|  |
| The program wil be exit if the user enters a <-1 or <0 in a GWA and age  Codiva link: <https://www.codiva.io/p/18646570-3dcd-4fd4-9f20-5b8cd2857f67> |

**V. RESULTS INTERPRETATION/OBSERVATION**

|  |
| --- |
| In this program code, encapsulation is greatly applied in every data that is present. Data hiding is implemented because it is required to use the set method to access the data. Whatever the user has inputted the system, the code data will not be tampered if the set method is not used. It is implemented in every data getting and in every data setting. At the end of the code all of the recorded data will be presented with ease.in the laboraty exercise 3 is we modified the setters in the method,we add a checker or if else to check if the inputs of the users are valid. If valid we set it else we end the program. |

**VI. CONCLUSIONS**

|  |
| --- |
| I learned in this laboratory exercise is that we can incapsulate as many as we want in classfields, Encapsulation is the Java term for combining code (methods) and data (variables) into a single entity. A class's variables are shielded from view by other classes and are only accessible through the methods of the class in which they are found thanks to encapsulation.in the exercise 1 is we don’t use the encapsulation and getters and setters.in the 2nd exercise is we set the variables in to private then we add a setters and getters by variables.so that the user will not directly access are variables and he or she use the setters and getters,in the 3rd exercise is we added a checker that will check if the users input is valid like for example is in the GWA the program only accept a >-1,if the user enter a <-1 is the program will automatically exit,and if the agge is <0 is also the program exited,else if the users input are valid the input of users will be added in the variables that hold the GWA and age. |

**VII. STUDENT OUTCOMES ADDRESSED**

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

**VIII. APPENDICES**

1. **RUBRICS AND SCORING**

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