

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**FINAL ASSESSMENT FOR THE BSC (HONS) INFORMATION TECHNOLOGY; BSC (HONS) COMPUTER SCIENCE; YEAR 2**

**ACADEMIC SESSION 2021; SEMESTER 3**

**PRG2104: OBJECT ORIENTED PROGRAMMING**

**Project DEADLINE: Week 14**

**INSTRUCTIONS TO CANDIDATES**

# 

# This assignment will contribute 50% to your final grade.

* This is an individual assignment.

**IMPORTANT**

# The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

# Coursework submitted after the deadline will be awarded 0 marks

**Lecturer’s Remark** (Use additional sheet if required)

I.............................. (Name) ...................std. ID received the assignment and read the comments....................................... (Signature/date)

**Academic Honesty Acknowledgement**

“I ........Chew Chien Zhen / 19037746........(student name). verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment.”

….......Zhen/14/11/21......... (Student’s signature / Date)

Overview

The aim of this project is to achieve the learning outcomes of [d] of this subject as mentioned in the syllabus, your role is to analyse, apply, and design a software application using object oriented programming. You also need to demonstrate your work at the time of submission. This overall assignment mark will contribute 50 % of your final grade.

**Table of Content**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1.0 Abstract 3**

**2.0 Introduction 4**

**3.0 Use cases used 5**

**4.0 UML diagram of the system 11**

**5.0 How the program works 12**

**6.0 Problems initially faced and how they were solved 15**

**7.0 Self-reflection on the overall OOP project 16**

**1.0 Abstract**

It is a simple calculator where it is used to calculate simple math calculations. It is designed to be able to do the four basic math calculations in one single page to make it easier to use.

The goal of this project is to code a calculator using scalafx GUI library only. During the calculator’s development, SceneBuilder was often used in making the program’s interfaces. The RootLayout, Welcome page, and Main page is made through the SceneBuilder.

**2.0 Introduction**

In this report, we will be going into details about the simple calculator I designed using scalafx GUI only and how it functions. By using the scalafx programming language, this simple GUI calculator app is designed to perform basic calculations like Addition, Subtraction, Multiplication and Division. This report covers all the factors and the difficulties as well as flaws of the application that was designed.

The scalafx designed base calculator was designed to be easy to use and is intended to require minimal page changing but also handling the calculation task all in one go. The report will also cover what improvements can be done to make the calculator better for usage with more time. The relationships between classes or objects will also be discussed later on.

Finally, there will also be a discussion section of how the calculator works and screenshots of the calculator’s interfaces will be included to give a better and clearer view of what the calculator looks like and how it functions as a simple calculator.

Graphical user interface

Description automatically generated

**3.0 Use cases used in this GUI calculator design**

1. The first use case is the check in button on the Welcome page. By clicking the big “Start” button, it will direct the user to the calculation page where the user can start doing their calculations

Welcome page:

Graphical user interface, text, application

Description automatically generated

Calculation page:

Graphical user interface, website

Description automatically generated

1. The second use case is the “=” button that are individual (one “=” button for every calculation mode) for all four Addition, Subtraction, Multiplication and Division. They are all separate operations.

Addition:

Graphical user interface

Description automatically generated

Subtraction:

Graphical user interface

Description automatically generated

Multiplication:

Graphical user interface, website

Description automatically generated

Division:

Graphical user interface

Description automatically generated

1. The third use case is being able to close the file under the File tab in the Menubar of the program. This is possible with the help of program called SceneBuilder. Inside the SceneBuilder we have access to the Menubar. Under the Menubar, we can find the Menu File.

Graphical user interface, text, application

Description automatically generated

Under the Menu File we can access the Menuitem Close which we can configure by making it call the handleClose in the On Action column which can be found under “Code : MenuItem” which closes the program without the need to click the “X” on the top right.

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

On an important note, it is a must to make sure the Mnemonic Parsing checkbox is checked. This can be found under the “Properties : MenuItem” on the very bottom of the tab. Make sure to save the file by hitting “Ctrl + S”



Top left under File, click on Close.

Graphical user interface, application

Description automatically generated

Once it closes, command prompt will display as follow. (successful close execution)

Text

Description automatically generated

1. The fourth use case is closing the program only with your keyboard. This is possible also because of the SceneBuilder’s help.

Firstly, is to make certain keys on your keyboard to open certain tabs on the Menubar. This is possible by simply putting an underscore ( \_ ) right before the alphabet.

In this case, we want to access the the File tab to reach the Close option to close the program with our keyboard. Therefore, we need to add an underscore before the alphabet F and make the F key on our keyboard as the hotkey for the File tab. Same goes for the Close option under the File menu. In this case we will use O as the Close option’s hotkey.

Leftside menu in the SceneBuilder:

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, table

Description automatically generated

So now the hotkeys are set and ready to be executed through the keyboard.

What it looks like when executed on the program. ( Alt + [hotkey] )

Press the Alt key to highlight the tab



Then follow up with the F hotkey to open the dropdown menu.

Graphical user interface

Description automatically generated with low confidence

And then hit the O hotkey on your keyboard to close the program completely.

The underscore ( \_ ) underneath the alphabet is an indication of what hotkey is linked to it.

**4.0 UML diagram of the system**

Diagram

Description automatically generated

*Figure 1*

Relationship between classes/objects in the program:

**5.0 How the program works**

Execution through command prompt:

The whole calculator program is a file that must be executed through command prompt.

Start by calling the folder with the command “cd (folder name)” and follow up with the command “sbt”

A picture containing text, orange

Description automatically generated

After that just run the program by executing the “run” command.

Text

Description automatically generated

The execution will popup the program’s Welcome page.

Graphical user interface, text, application

Description automatically generated

Click on start next to go to the main Calculator screen.

Graphical user interface

Description automatically generated

Start calculating from thereafter. A simple calculator for simple calculations.

Additional info, there is a hotkey to close the program without using a mouse.

( Alt -> F -> O ) hotkey on keyboard to close the program.

Graphical user interface, text, application

Description automatically generated

The button (Recording Template) on the very bottom leads the user to an empty template for recording purposes as a temporary interface

Graphical user interface, table

Description automatically generated

The Back button on the bottom brings the user back to the calculating page(addition, subtraction, multiplication, division).

**6.0 Problems initially faced and how they were solved:**

Errors when trying to insert values:

* Initially when designing the main calculation pages, a lot of errors were showing up in the command prompt when attempting to insert a value in either left or right side of the column.
* These errors were solved by just adding more private val operators under the class “Buttons” in the Buttons.scala code.

“Not a member of MainApp” error

* The Startpage class was showing on the command prompt indicating it was not a member of the MainApp therefore MainApp.showMain() could not be executed
* This issue was solved by importing the ch.makery.address.MainApp into the Starpage.scala code.

**7.0 Self-Reflection on the overall OOP project**

In this solo project, I have made use of the scalafx api library and Practical recordings to piece together the code for the Simple calculator. The process of importing from the scalafx library is important as you put in your code. A lot of results ended up with the error of either the class or the object not being a member of the main scala. After looking through certain Practical recordings I was able to solve all of the errors and managed to make the program function as intended.

The problems faced during the project’s middle phase was there was a lot of errors when trying to execute the calculator through the command prompt. Initially when designing the main calculation pages, a lot of errors were showing up in the command prompt when attempting to insert a value in either left or right side of the column. These errors were solved by just adding more private val operators under the class “Buttons” in the Buttons.scala code. Another problem is the Startpage class was showing on the command prompt indicating it was not a member of the MainApp therefore MainApp.showMain() could not be executed. This issue was solved by importing the ch.makery.address.MainApp into the Starpage.scala code.

Moving on to the project’s strength and weaknesses. The strength is that this calculator is easy to use and is very straight forward. There is not much needed to get to the main page which is the calculator part and it is not confusing to use. The weaknesses of this project is that the calculator does not support decimals therefore it is only capable of displaying whole numbers.

To conclude this project, the calculator can be significantly better given more time as more function could be implemented to make the calculator more capable to display calculations. As of now, the calculator is only capable of doing simple calculations (like addition, subtraction, multiplication and division) and it is only capable of displaying whole numbers as an answer.

**assignment SPECIFICATION**

|  |  |
| --- | --- |
| **Learning Outcome Being Assessed** | 1. Write a program that use: object-oriented programming concepts - inheritance, polymorphism, GUI programming - event-driven programming, layout managers. |
| **Submission Deadline**  **Microsoft Team Submission** | **Monday, (Week 14) by 4.00p.m.**  Late submission will be awarded 0 mark.  Create a submission folder named “Project\_yourID”. Put your project folder and documentation report into this submission folder. Zip it and submit this zipped file into the Microsoft Team.  In Microsoft Team, the zip file can be inside the assignment page and in the Onenote assignment page should contain the reports for lecturer to provides feedback. |
| **Outline of Problem** | This assignment stipulates the design of a system by the identification of the classes required and the relationships among them.  Students are required to demonstrate the ability to apply their knowledge on inheritance and polymorphism in their implementation of the solution. Students also need to make use of scalaFX GUI library to create an GUI Application. Other GUI libraries are not acceptable. |
| **Detail Question** | Propose a standalone GUI system such as the following  1. Calculator  2. Scientific Calculator  3. Library Management System  4. Todo List  5. Any Personal Games  6. Queue System  7. Customer Management System  8. Chess  9. more…  The proposed system should have least four use cases. For examples: A Library Management System should have at least four following use case:  • Check In  • Check Out  • Search Book  • View Book    This system should utilize object oriented programming concept in designing and developing.    You should proposed additional classes or trait to achieve elegance design.  You are required to design your own user interface for ease of use.  If you refer to any sources from youtube, github or other code repository in creating the application, please do citation. Any works that are not created by you will not be considered in marking.  If you refer to sources from your senior, you will be caught for plagiarism. System will check and compare your code from previous and current submission library. |
| **What you should hand in** | The following items are to be handed in:   * A cover page (use the template provided). * A documentation report includes the UML diagrams that describe the classes/objects identified from problem domain, and their relationships among these classes/objects used in the program. – in softcopy * A description that show the program is working for all the 4 features. * An A4 page to be written by **you on the** personal reflection that includes:   + A description of how you applied the above object oriented concepts in your assignment.   + The problems encountered during this assignment and how you solved these problems.   + An evaluation of the strengths and weaknesses of your submitted work. * The project/solution files, including the source code for the Item class, all pre-compiled classes, test driver program and application program. – in **softcopy**   NOTE: Submitting the assessment means you have agreed that your work is original and comply with the rules and regulations (refer to Academic Impropriety) |
| **Paper Size / Format** | |  |  | | --- | --- | | Paper size | A4 (Use only one side of the paper) |   For the personal reflection write-up,   |  |  | | --- | --- | | Paragraph format | 1.5-line spacing | | Font size | 12 points | |
| **Academic Impropriety** | Sunway University takes a strong stand on plagiarism. Any students found to have copied work, colluded or presented work that is not their own will be punished under the terms stated in the rules and regulations booklet. Students are permitted to use 3rd party components, however all such code must be well described and credit awarded to the respective owner. Students must also ensure that the majority of source code is their own, and that the core algorithms are their own work. The use of copyright materials is forbidden.  \*subject to change anytime without prior notification  **The work that you submit must conform to those regulations.** |
| **Assessment:**  **Report** | Contributes 50% to the overall final assessment mark.  Refer to ASSESSMENT CRITERIA FOR Project table for further elaboration of marking distribution. |

**ASSESSMENT CRITERIA FOR Project**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mark /**  **General Impression** | **Area / Assessment Criteria** | | | | | | | |
| **Class Definition and Design** | | | **Application Program** | | **Style** | **Use of Third-Party Library** | **Documentation** |
| **Fulfillment of Requirements**  **(x2)** | **Relationships  among classes** | **UML Class  Diagram** | **Fulfilment of requirements**  **(x2)** | **GUI Implementation** | **Naming  Convention** | **Mastery** | **Documentation (Report)** |
| 5  Excellent | 1. Correct and complete  * Classes – both basic classes and the “collection” class * Identification of data fields, visibility modifiers and types. * Constructors  1. Additional features / operations provided. 2. Originality and Uniqueness | Correct application of all the following concepts:   * inheritance * polymorphism * abstract class * generic programming. | Perfectly correct diagram.   * All notations are correctly used. * Class members are complete, and * The diagram is consistent with the class design. | The following are provided   * Menu navigation * Execution of all the required operations * Input validations * Completely correct, efficient and elegant use of programming constructs. * Methods are extensively used to achieve complete modular programming. | Illustrated an excellent mastery in Event Driven Programming and correct use of layout classes.  GUI Components and layout component are design and use correctly. | Full adherence to naming convention with appropriate, meaningful and correctly spelt identifier names. | An excellent correct use of third-party libraries to solve problem.  No errors in utilizing the third-party libraries.  Demonstrated in depth understanding of third-party library model. | * Complete and well written documentation. * All Required section is included. * Table of content is formatted properly.   Very few typo or spelling mistake. |
| 4  Very Good | Correct and complete   * Classes – both basic classes and the “collection” class * Identification of data fields, visibility modifiers and types. * Constructors * Quite original and uniqueness | Correct application of the following concepts:   * inheritance * polymorphism * abstract class | * Complete with only one very minor error in notations used. * Diagram is consistent with class design. | The following are provided   * Menu navigation * Execution of all the required operations * Correct and efficient use of programming constructs. * Methods are used to achieve a high degree of modular programming. | Illustrated a good mastery in Event Driven Programming and correct use of layout classes.  GUI Components and layout component are design and use correctly with minor error. | Adherence to naming convention with meaningful, appropriate and correctly spelt identifier names. | Make a good use of some third-party libraries to solve problem.  No errors in utilizing the third-party libraries.  Demonstrated in good understanding of third-party library model. | * Complete and well written with only one very minor error in documentation. * Not all required section is included only miss one section. * Table of content is formatted reasonably.   Few typo or spelling mistake. |
| 3  Average | Correct and complete   * Basic classes * Identification of data fields, visibility modifiers and types. * Constructors * Partial originality and uniqueness | Correct application of the following concepts:   * inheritance * polymorphism | Quite complete with not more than two minor errors in notations used. Diagram is consistent with class design. | The application program demonstrates the correct execution of all the required operations  One or two minor errors. | Illustrated an average mastery in Event Driven Programming and correct use of layout classes.  GUI Components and layout component are design and use correctly with some error. | General adherence to the naming convention with one or two minor errors. | Make use of some third-party libraries to solve problem.  Minor errors in utilizing the third-party libraries.  Demonstrated in average understanding of third-party library model. | * Quite complete and good written with only few minor error in documentation. * Not all required section is included only miss few section. * Table of content is formatted.   Few typo or spelling mistake. |
| 2  Poor | Some errors or 1 incomplete basic class.  No originality with some uniqueness | Correct application of inheritance | Some errors in notation or diagram’s consistency with class design. | Incomplete application program.  Some errors. | Illustrated a poor mastery in Event Driven Programming and correct use of layout classes.  GUI Components and layout component are design and use correctly with few error. | Limited adherence to naming convention. | Make use of a third-party libraries to solve problem.  Some errors in utilizing the third-party libraries.  Demonstrated in poor understanding of third-party library model. | * Incomplete in documentation with few major error. * Not all required section is included with missing lot of section. * Table of content is partially formatted.   Lot of typo or spelling mistake. |
| 1  Very Poor | Very major errors or more than 1 incomplete basic class. | Illogical inheritance hierarchy | Major errors in notation or diagram’s consistency with class design. | Grossly incomplete application program.  Very major errors. | Illustrated a very poor mastery in Event Driven Programming and incorrect use of GUI components. | Serious lack of adherence to the naming convention. | Make use of a third-party libraries.  Major errors in utilizing the third-party libraries. | * Grossly incomplete in documentation with major error. * Major required section is not included. * Table of content is not formatted.   Lot of typo or spelling mistake. |