**ASSIGNMENT 1 FRONT SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | **JavaProgramming** | | |
| **Submission date** |  | **Date Received 1st submission** |  |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | Nguyen Huy Hoang | **Student ID** | GCH200739 |
| **Class** | GCH0908 | **Assessor name** | Dinh Duc Manh |
| **Student declaration**  I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice. | | | |
|  |  | **Student’s signature** | Hoang` |

**Grading grid**

|  |
| --- |
| Grade (0-10) |
|  |

|  |  |  |
| --- | --- | --- |
| **❒ Summative Feedback: ❒ Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **IV Signature:** | | |

Table of Contents

[A. INTRODUCTION 4](#_Toc97576614)

[B. REQUIREMENT 4](#_Toc97576615)

[I. Non-functionality Requirements: 4](#_Toc97576616)

[II. Functionality Requirements: 4](#_Toc97576617)

[C. UI DESIGN 5](#_Toc97576618)

[I. Login wireframe 5](#_Toc97576619)

[II. Register wireframe 6](#_Toc97576620)

[III. Main Interface wireframe : 7](#_Toc97576621)

[D. IMPLEMENTATION 8](#_Toc97576622)

[I. Explain Program Structure 8](#_Toc97576623)

[II. Explain Classes 16](#_Toc97576624)

[1. Class Account 16](#_Toc97576625)

[2. Person class: 17](#_Toc97576626)

[3. Student Class 18](#_Toc97576627)

[III. Explain Important Algorithm. 19](#_Toc97576628)

[IV. Explain how to handle errors 19](#_Toc97576629)

[E. TEST 21](#_Toc97576630)

[I. Test Plan 21](#_Toc97576631)

[II. Test Log 21](#_Toc97576632)

[F. RESULT 24](#_Toc97576633)

[G. CONCLUSION 28](#_Toc97576634)

1. **INTRODUCTION**

On the occasion of the summer, a lecturer in greenwich want to self-open an extracurricular class to be able to train soft skills and share some enterprise experiences for some aspiring students. Because of quite small amount of students participating as well as storing list of participating students on paper is quite obsolete and waste of time, he ask me to make for him a simple app to manage list of these students in this extracurricular class. Based on his requirement, I need to make a simple application via java swing which at least has four primary functions: create - read - update - delete.

1. **REQUIREMENT**
2. **Non-functionality Requirements:**

* Simple colouring and easy-to-use graphical design.
* Clear and readable label for each button as well as text field.
* Efficiency in managing a small amout of information.
* Able to run on Window 11 Operating System.
* English is used as main language.
* Data input: Full name, gender, major, Student ID, address, email.

1. **Functionality Requirements:**

* Adding Student information into list.
* Delete Student information in list.
* Update – edit student information in list.
* Searching for student information in any field.
* Login – logout.
* Create more account for other manager.
* Store all information to file text or sql.

1. **UI DESIGN**

List of Wireframes:

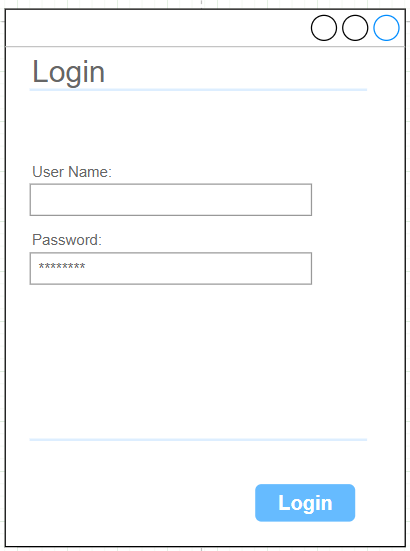
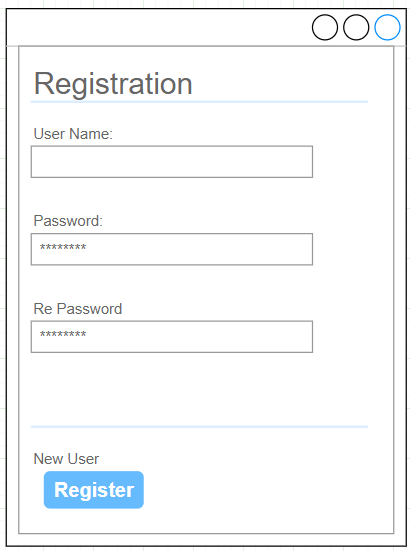
1. **Login wireframe**

Figure : Login Frame

Explain: This frame is the first interface appear each time user want to use this app. Because account which can login to this app also have provided for main user, this form has no button to register the new account.

1. **Register wireframe**

Explaination: Although Login frame has no button to switch to registration frame, This frame also has been designed and it can be open via main interface of application in which just user(lecturer) can create more account for the others.

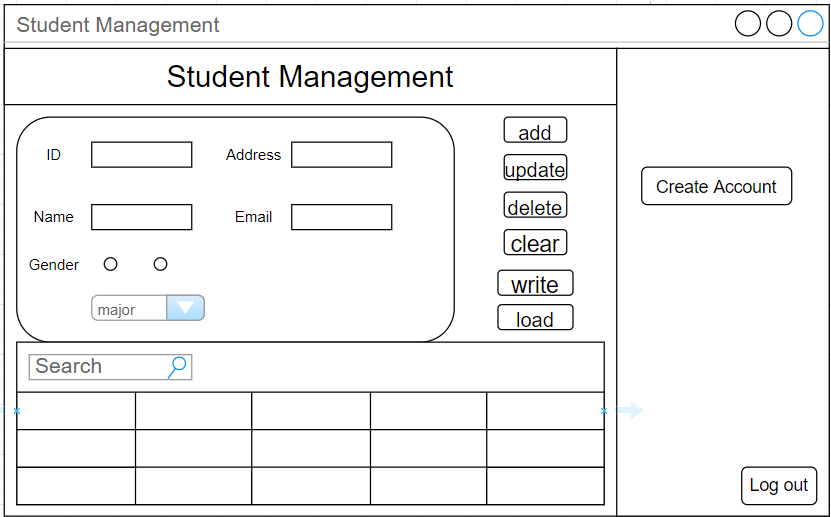
1. **Main Interface wireframe :**

Figure : Main interface Wireframe

Explain: This is main interface wireframe of application. There are some text field to type the information of student. Buttons which to manipulate with functions of this app. Below the input form will be the table that show all information of student which either load from file text or input from user. Additional, here is the search field which help user easier to find the information of student based on key word. In the right side of application, button create account will have function to help user can register more account for others and log out button clicked will dispose this form as well as sign out of system.

1. **IMPLEMENTATION**
2. **Explain Program Structure**
3. **Source Package Struture:**

|  |  |
| --- | --- |
|  | * Images Packages: Store image of app * LoginFrame Packages: Containing Java files which perform login-register function of application (Class Account, LoginFrm Jframe) * Swing Packages: Containing only ManageFrm Jframe file which is the main interface of this application. * fullClass Packages: Containing Java files which has been programmed in form of OOP paradigm(Interface Manage, Person class, Student class, StudentList class). Thereby, they will be used to initialize object that help in designing app process. |

1. **Source Code:**
   1. **LoginFrm.java File:**

|  |  |  |
| --- | --- | --- |
|  | * Import Java.io.”…”Exception to handle Exception thrown in Write and Read data process from File Text in each stream. * Import Java.io.File(OutPut-InPut)Stream -two stream objects that facilitate file access to help writing and reading data (Byte-based) to file text. * Import Java.io.Object(OutPut-InPut)Stream To read an entire object from or write an entire object to a file via serialization. Serialized object is represented as a sequence of bytes that includes the object’s data and its type information * Import Java.util.ArrayList and Java.ul.List to handle Object of class Account with Type List and arraylist. * Import Javax.swing.JoptionPane to show notify or confirm pane to Frame. | |
|  | | * loadData() and writeDataToFile() function perform the ability to read and load data (Account information) from text file. * Both of them also use try-catch surrouding around code to catch exception when handling data from text file. * In loadData() function, there is while loop which just execute when ois( object of ObjectInputStream) is set with available() method and the value return is greater than 0, code inside while loop will be executed. Particularly, accounts arraylist will set by casting of ois data. * In WriteDataToFile() function, it operates quite similar as loadData(). It just write all data in accounts array list to file by writeObject method. |
| **Navigator of Loginfrm GUI** | | * TxtField and function buttonn have been located above other panel because they need to be shown on screen that help user can see and click. * jSeparator has been used to make textField more eye-catching. * Jlabel1 is the label which has been set the background image of LoginFrm. |

* 1. **ManageFrm.java**

|  |  |
| --- | --- |
|  | * Import LoginFrame.LoginFrm and LoginFrame.registerFrm to handle the login and register to app. * Import fulClass.(Person-StudentList-Student-Manage) to import class and interface which will help handle and store the information of student in temporary memory. * Import Java.io.”…”Exception to handle Exception thrown in Write and Read data process from File Text in each stream. * Import Java.io.File(OutPut-InPut)Stream -two stream objects that facilitate file access to help writing and reading data (Byte-based) to file text. * Import Java.io.Object(OutPut-InPut)Stream To read an entire object from or write an entire object to a file via serialization. Serialized object is represented as a sequence of bytes that includes the object’s data and its type information * Import Java.util.ArrayList and Java.ul.List to handle Object of class Account with Type List and arraylist. * Import Java.until.Calendar to call and display the time based on UTC and locale of my computer. * Import Javax.swing.table.DefaultTableModel to handle and manipulate with the table in frame which display all information of student. * Import Javax.Swing.RowFilter-TableRowSorter to make the searching function of app. |

|  |  |
| --- | --- |
|  | * Both of their function will be invoked when user click on LoadFile or WriteFile button, this will invoke these event. These functions perform the ability to read and load data (Student information) from text file. * Both of them also use try-catch surrouding around code to catch exception when handling data from text file. * In btnLoadFileActionPerformed() function, there is while loop which just execute when ois( object of ObjectInputStream) is set with available() method and the value return is different to 0, code inside while loop will be executed. Particularly, I initialized the object of Student class and assigned it with casting of ois data and then i add it to ds List(list). SetList is the method in interface Manage which set this list to list student. Showtable to show the information just added to table * In btnWriteFileActionPerformed() function, it operates quite similar as btnLoadFileActionPerformed(). It just write all data in student array list to file by writeObject method. |
|  | * This function will be invoked when user click on Add btn on GUI. It will executed and check whether this new information of student has already existed. If not, it will add this student into Student list. |

|  |  |
| --- | --- |
|  | * checkFrom() function perform not only the check the validation of form, I also get all value from text field and automatically assign them to available variables. * Additionally, this function also initial new object of Student Class and transmiss full value from text field into parameters of constructor in student class. |
|  | * When user click Delete Btn on GUI, the event of this button will be invoked. btnDeleteActionPerfomed() function perform the delete function of this app. It will get the selected row index in table. The “if-else” statement will be executed and with condition editedIndex must different to -1, the code inside will be executed. Delete is the method of interface Manage which perform remove this object out of list Student and show table again without of this student Information just deleted. * When user click Update Btn on GUI, the event of this button will be invoked. BtnUpdateActionPerformed function perform edit information of student in table. It also get selected row and display it onto text fields. If this student information after check in list and it is not null, Outlist is method of interface Manage which return the list student permit user set the new information just updated to this choosen student information row and. After that, showTable() function will show this new student information after updating in table on main interface of application. |
| **Navigator of ManageFrm form** | * This GUI form has been separated into three primary panels, each panel contain fixed function: * jPanel1 contains 6 buttons and they perform 6 main functions of this application include : Add, Update, Delete, Clear, WriteFile, LoadFile. They located above other panels because they need to be shown on screen that help user can see and click. Additionally, this also contain txtSearch which have set the event for searching function, it will search immediately as soon as user type characters into txtSearch. * Jpanel4 contain the jlabel7 which has been set for the name of application and color. * jPanel3 contains some other decoration and create and logout buttons. |

* 1. **JUnit TEST for ManageFrm:**

To help ManageFrm can implement properly, StudentList Class play a quite important role because it has some method that will support so much in adding student information into object list process. So, below is the Junit test of StudentList

|  |  |
| --- | --- |
|  | * Three of important methods in class StudentList need to test with Junit Test are adding, OutList and SetList Function. * In testAdding, i has tried testing the ability to add an object of class Person(class has been inherited by class Student). * In testOutList, this has been more complicated than testAdding, because it need to have a value inside that need to added before outlist has been tested. assertEquals has compared the output of ArrayList which executed OutList method. * In testSetlist, it quite similar to testAdding. |
|  |  |

**Result from JUNIT TEST:**

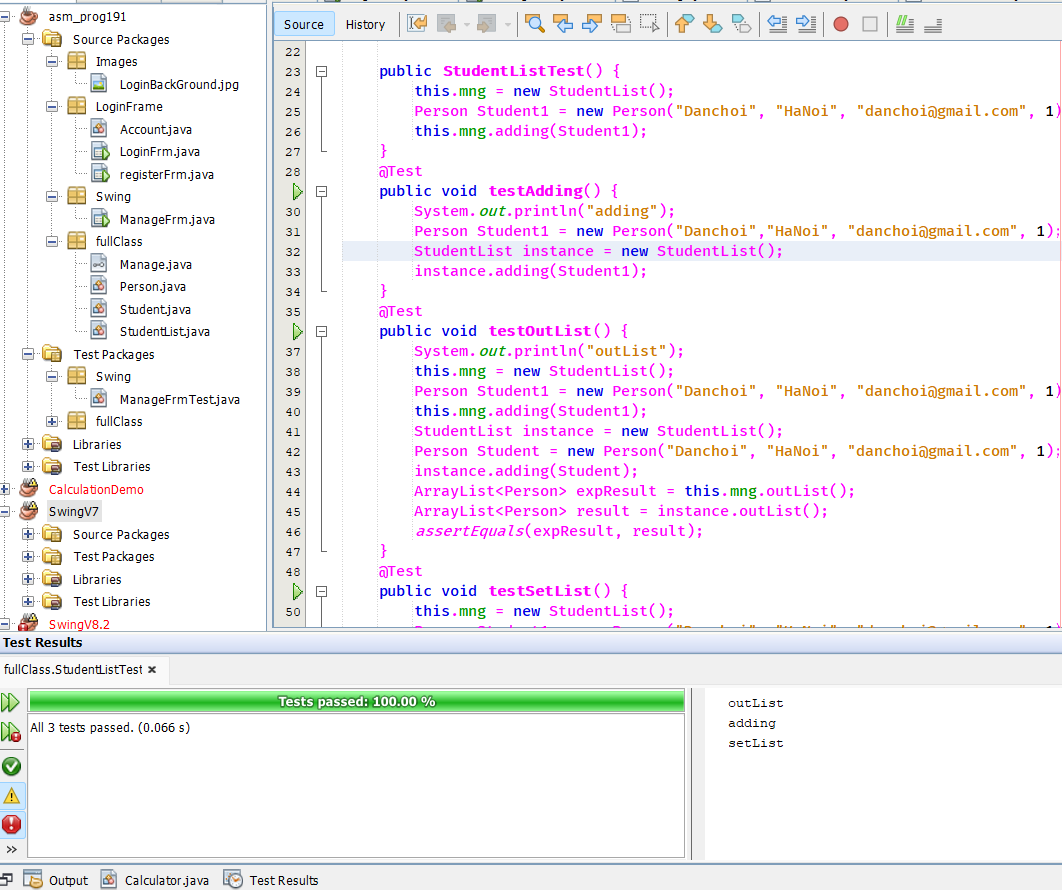
****After some times fixing and debug, the test result are passed:

Figure : Result of JUNIT test

1. **Explain Classes**
2. **Class Account**

|  |  |
| --- | --- |
|  |  |

Table : Source code of Class Accout.

This class located in LoginFrame package, with “Public” access modifier permit to initialize and call the object of this class inside and outside package. Account class implement interface Serializable for purpose to read and write data to file. Because when exchanging data between application and text file, the data is represented as bytes, not as objects. So, when "account" class implements Serializable interface, objects of this class be able to be converted to byte streams and they can be stored this byte stream in particular memory and in here is in a text file.

There are two properties of account class, which are “username” and “password” and both are set with “private” access modifier to ensure Encapsulation of object-oriented programming.

Constructer Account has applied constructer overloading with one constructer has enough parameter and one constructer has nothing. This help initialize the new object of this class can with or without parameter transmission.

Some methods getter and setter which automatically created to perform actions of this class. Additionally, two methods have been overrided: Equal and Hashcode(which are implicitly defined in every Java class) to support working with collection. Override the equals() method so that it doesn't consider only object identity, but also the value of the two relevant properties, in here is examine the exists or accuration of account. Once I override equal() method, this is essential to override hashCode(). It returns an integer representing the current instance of the class. Calculate this value consistent with the definition of equality for the class.

1. **Person class:**

|  |  |
| --- | --- |
|  |  |

Person class located in fullClass package and created with “Public” access modifier.

This class is represent for basic charateristics of a person which include enough information of a student should be input in list student.

Here are 4 properties which are ‘name’, ‘address’, ‘email’ and ‘gender’, all of them also initialized with “private” access modifier. Constructer also get all properties of a person. Additionally, some getter and setter methods have been automatically inserted to perform other actions of class Person.

1. **Student Class**

|  |  |
| --- | --- |
|  |  |

Student class is the most important class, it located in fullClass Package and created with “Public” access modifier. Student class extends Person class to inherit characteristics and basic information which have been declared as properties in class person(which will be the essentials information of a student). Additionally, This class also implements Serializable interface to ensure read and write data of students list to file smoothly.

Two properties of this class are ID and major and both declared with “private” access modifier. All setter and getter methods as well as equal and hashcode method also automatically created and they play roles as in Account class.

1. **Explain Important Algorithm.**

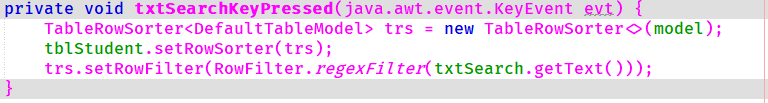
This Application has been designed quite basic and simple; hence, it seem to have no complicated algorithm. However, there also are some important algorithm and most of them are in Searching function.

Table : Source code of Search function

Although this source code is quite brief but it also has basic Comparison algorigthm and Filter algorithm which are implement on the table that show full information of student list. Particularly, when I initialize “TableRowSorter”, and then I use setRowSorter method for tblStudent(the table display student’s informartion) TableRowSorter will invoke Comparators for doing comparisons.

Because this search can search in any field(column in table), Comparator compares and will has been specified for the column that match with string which transmissed in from txtSearch(Text field for searching) and RowFilter is used to filter out the row that corresponds to this and setRowFilter will display this row in table.

1. **Explain how to handle errors**

Because most of events in this application also are working and interact with file, there are some unexpected events that occurred during the execution of a program and specifically they are exceptions (such as ‘ArrayIndexOutOfBoundsException’, ‘FillNotFoundException’, ‘NullPointer Exception’, ‘EOFException’, …) which thrown many times when I designed this application.

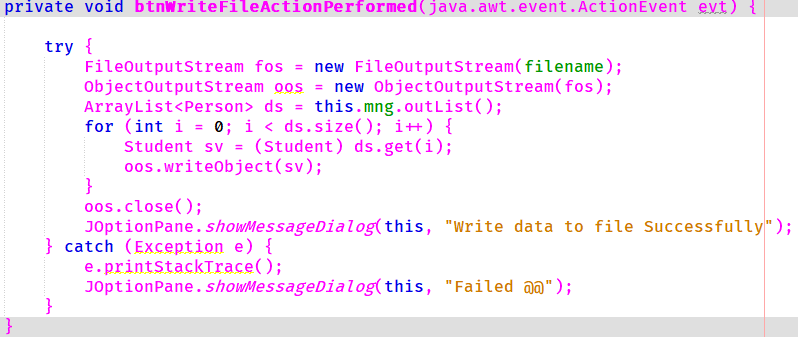
To handle these errors, the best way that I can perform is using try-catch surrounded each code that interact with text file.

Figure : Source code using try-catch to handle errors

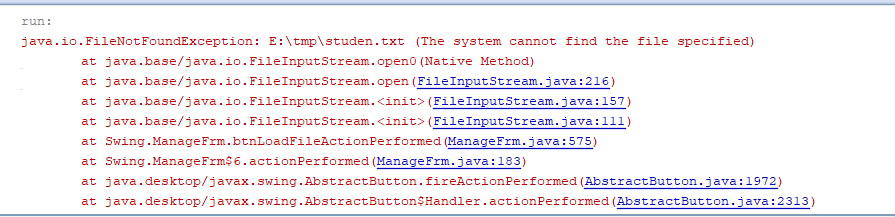
Specifically, when an error/exception (maybe the file name is wrong ) inside the try block is thrown, the catch block will be executed to handle the exception. This will help me avoid breaking the operating process and catching any errors of this application. Additionally, for easy determine the errors of code, I use printStrackTrace() method in each catch block to print exactly what exception or what error that the program are meeting.

Figure : Example of printStackTrace to help handle errors.

1. **TEST**
2. **Test Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | What is being Tested | How | Test data used | Expected results |
| 1 | Validation of Input | Enter typical values, boundary value, values that should be rejected | Information of products | Good data accepted, bad data rejected |
| 2 | Verify the Signin/Signup process | Input some Username which has been used, Test system check account | Function involve to Signin/Signup | Reject the improper handle to signin/signup process |
| 3 | Functionality | Try using function in application | Function in app | Function operating properly and meet requirement. |

1. **Test Log**

Test Login/Register by interacting with the available account in file text: Username : danchoi; Password: 123

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Test Case | Decription | Actual Result | Expected Resuld | Overall |
|  | Test Login with wrong user name | Enter wrong username : “danchoi123” instead of “danchoi” | Error | Error | Pass |
|  | Test Login with wrong Password | Enter wrong password : “000003” instead of “123” | Error | Error | Pass |
|  | Test Login with empty username | Not enter username | Error | Error | Pass |
|  | Test Login with empty password | Test Login with empty password | Error | Error | Pass |
|  | Test Login with Exact Username and Exact Password | Login with username:”Danchoi”  &  Password:”123” | Login Successfully | Success | Pass |
|  | Create a new accout with the username has been used | User name “Danchoi” which has been used | Error | Error | Pass |
|  | Test Create a new accout with wrong common username structure | User name contain special characters such as ‘@’, ‘#’, “%” | Accepted by System | Error | False |
|  | Test Create a new accout with wrong common password structure | The length of password less than 8 | Accepted by System | Error | False |
|  | Test Create a new accout with proper username & password | Create a new accout with username “danchoi123” & password “123456” | Signup successfully | Success | Pass |
|  | View all Information of students list in main interface | Check all Information of students list in main interface | Information of students list in main interface are Viewable in Homepage | Success | Pass |
|  | Test checking assurance of system | Try to get in the main interface before login | Error | Error | Pass |
|  | Test checking assurance of system | Make the text field be blank and click add button | Error | Error | Pass |
|  | Test Functional ability | Open and close Application | Open and close smoothly | Able to open and close | Pass |
|  | Test Functional ability 2 | Add new Student information | Adding successfully | Able to add new student information | Pass |
|  | Test Functional ability 3 | Delete existed Student information in table | Deleting successfully | Able to delete existed student information | Pass |
|  | Test Functional ability 4 | Update or edit information | Accepted by system | Able to edit existed student information | Pass |
|  | Test Functional ability 5 | Clear form | Success | Success | Pass |
|  | Test Functional ability 6 | Log out | Able to Log out | Able to log out | Pass |
|  | Test Functional ability 7 | Create new account | Easily Create new account | Success | Pass |
|  | Test Functional ability 8 | Searching any student information with key word | Easily search student | Able to search student | Pass |
|  | Test Functional ability 9 | Load File Text with wrong path or wrong file name | Notify Error | Error | Pass |
|  | Test Functional ability 10 | Write File Text with wrong path or wrong file name | Notify Error | Error | Pass |
|  | Test Functional ability 11 | Load and write file with exact path and file name | Success | Success | Pass |
|  | Test Functional ability 12 | Searching student with wrong key word | Accepted by system | Notify Not Found | False |

1. **RESULT**

Screenshots of running program:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |
| --- |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
|  |  |

1. **CONCLUSION**

Learning and making a simple application above by Java Swing make me can get some more knowledge in Understand basic programming skills and OOP paradigm as well as gain experience with Java development environments, learn how to program with graphical user interfaces. Not only that I partly understand how to working with files in java applications and detect and handle uncomplicated errors.