**Final Project**

**Object Oriented Programming**

Project Name: “Library Management System”

Student Name: Nathan Darien Tanner

Student ID: 2602225656

Class: L2BC

# **Table of Contents**

**Cover Page**

**Contents**

**A. Description**

I. Introduction

II. The function of the program

**B. Design**

I. Parts of the program and requirements

II. Function of each part of the program

**C. Implementation**

I. UML Diagram

II. Extensibility

III. Explanation of the methods made and used

**D. Lessons learned**

I. Learning about the concepts of OOP

II. Error handling and debugging

**E. Evaluation**

I. Does the program work properly

II. Future improvements that can be done

III. Reflection

**F. Evidence of a working program**

# **A. Description**

## **I. Introduction**

After it was announced there was a final project for Object Oriented Programming, I started browsing the internet for simple/intermediate Java projects that uses object-oriented programming principles. I specifically looked for a project that suits my interests and level of understanding in the Java language, and I ended up finding a project that was the perfect fit for me. The project was a Library Management System used to store information on which books are borrowed as well as who borrowed them.

<https://www.youtube.com/watch?v=LrIz-4lxURo>

I used the tutorial above to help me with the basic structure of the code, and then added a few more things to it so that the users can have a more immersive experience. I used IntelliJ IDE for this project and I will be posting all the code along with this report on my repository which can be accessed through my GitHub account here:

https://github.com/NathanDarienTanner/LibraryManagementSystem

## **II. The function of this program**

The function that this program serves is that of a Library management system with the purpose of serving as a catalogue of sorts for the users. In order to access the library management system, users must first enter their username and password, if they have forgotten the username/password or both, there will also be a backup login system in the form of a secret code. After the user has entered into the management system, the users will then have several features which they can make use of such as student registration as well as book issuing.

# **B. Design**

## **I. Parts of the program and requirements**

Requirements:

To run the program, you need to download Java and the SDK on your device. And then you can run it on an IDE or any application that can properly run a Java file.

## **II. Function of each part of the program**

The program starts out with the login page which will require users to enter their username and password in order to access the management system, if they have forgotten their username/password or both, they can alternatively use the secret code option in order to obtain their username and password.

Now that the user has access to the management system, the user can now make use of the features implemented into the library management system (LMS). The first implemented feature that we will be discussing is the registration feature which will mainly be used to keep track of who is going to be borrowing the books. This feature will ask the user to input the data and information of the person they are registering, the information that is needed include, the person’s ID, name, parent’s name, semester, and institute. Once all of the information needed has been filled in, the user can then add the data of the person they registered into the system. Another feature that has been implemented into the management system is the library department feature which allows the user to distribute books with ease to every department. This feature was made with the purpose to distribute books to the various institutes available when lecturers/teachers ask to borrow them. It comes with its own catalogue of books so that the user can just select the book which the institute wants to borrow and simply issue it to them.

# **C. Implementation**

## **I. UML Diagram**



## **II. Extensibility**

1. Comment lines are used to further explain the functions of each function and method and the purpose of each of them to help other programmers.
2. Variables use meaningful identifiers to help other coders to understand easier.
3. Methods are used to make the code easier to understand.
4. UML Diagram to make it easier for others to see how the relationship of the classes in this project.

## **III. Explanation of all the methods made and used**

### *DataEntry.java*

* initComponents(): This method is automatically generated by the NetBeans IDE's Swing GUI builder. It initializes all the components (such as labels, text fields, buttons, etc.) that are defined in the graphical user interface (GUI) form. The method sets up the initial appearance and layout of the form.
* jButton1ActionPerformed(): This method is the event handler for the "Submit" button's action event. It is called when the button is clicked. It retrieves the data entered by the user from various text fields and displays a message dialog box with the entered data.
* jLabel7MouseClicked(): This method is the event handler for the mouse click event on the "GO BACK" label. It is called when the label is clicked. It creates an instance of the MenuPage class, makes it visible, and closes the current DataEntry frame.
* main(): This method is the entry point of the application. It is automatically generated by the NetBeans IDE. It sets the look and feel of the GUI, creates an instance of the DataEntry class, and makes it visible.

These methods are specific to the Swing library and are used to handle user interactions, initialize GUI components, and start the application.

*EngDep.java*

* initComponents(): This method is automatically generated by the NetBeans IDE's GUI Builder. It initializes and sets up all the components (labels, buttons, text fields, etc.) on the form. The method defines the layout and positioning of the components.
* subActionPerformed(ActionEvent evt): This method is an event handler for the "OK" button. It is executed when the button is clicked. The method retrieves the selected item from the eng JComboBox (a dropdown menu) and sets the text of the sub button to the selected item. Then it displays a dialog box using JOptionPane.showMessageDialog() to show a message indicating that the book has been issued.
* jLabel3MouseClicked(MouseEvent evt): This method is an event handler for the "GO BACK" label. It is executed when the label is clicked. The method creates an instance of the LibDepartment class, sets it visible, and disposes the current frame (closing it). This allows the user to go back to the previous frame.
* main(String args[]): This method is the entry point of the application. It is responsible for creating an instance of the EngDep class and setting it visible. It also handles the look and feel settings for the GUI.

These methods are automatically generated by the NetBeans IDE and provide the necessary functionality for the GUI components to interact with the user and perform actions based on user input.

*LibDepartment.java*

* initComponents(): This method is responsible for initializing and setting up the graphical components of the user interface. It is automatically generated by the NetBeans GUI Builder and contains the code for creating and positioning the labels, icons, and other UI elements.
* jLabel3MouseClicked(): This method is an event handler that is triggered when the user clicks on jLabel3, which represents the image/icon for the English Department. It creates an instance of the EngDep class, sets it as visible, and disposes the current frame.
* jLabel4MouseClicked(): This method is an event handler that is triggered when the user clicks on jLabel4, which represents the image/icon for the Physics Department. It creates an instance of the phydep class, sets it as visible, and disposes the current frame.
* jLabel5MouseClicked(): This method is an event handler that is triggered when the user clicks on jLabel5, which represents the image/icon for the Chemistry Department. It creates an instance of the chemdep class, sets it as visible, and disposes the current frame.
* jLabel6MouseClicked(): This method is an event handler that is triggered when the user clicks on jLabel6, which represents the image/icon for the Newspaper Department. It creates an instance of the NewspaperDep class, sets it as visible, and disposes the current frame.
* jLabel7MouseClicked(): This method is an event handler that is triggered when the user clicks on jLabel7, which represents the "GO BACK" label. It creates an instance of the MenuPage class, sets it as visible, and disposes the current frame.
* main(): This is the entry point of the program. It sets the look and feel of the UI, creates an instance of the LibDepartment class, and sets it as visible.

### These methods are used for event handling, user interface setup, and navigation between different frames (windows) in the application.

### *LoginPage.java*

### initComponents(): This method initializes and sets up all the components of the login page, such as labels, text fields, buttons, and their properties. It is automatically generated by the NetBeans IDE's Swing GUI builder based on the design of the form.

### jButton1ActionPerformed(java.awt.event.ActionEvent evt): This method is an event handler for the "Submit" button (jButton1). It is called when the button is clicked. It retrieves the entered username and password from the corresponding text fields (user and pass) and checks if they match the expected values. If the values are correct, it creates an instance of the MenuPage class and opens it, then disposes the login page. Otherwise, it displays an error message using a message dialog.

### jButton2ActionPerformed(java.awt.event.ActionEvent evt): This method is an event handler for the "Submit" button (jButton2) on the "Forgot password?" section. It is called when the button is clicked. It retrieves the entered secret code from the sc password field and checks if it matches the expected value. If the code is correct, it displays a message dialog with the username and password. Otherwise, it displays an error message.

### main(String args[]): This is the entry point of the application. It sets the look and feel of the UI, creates an instance of the LoginPage class, and makes it visible.

### These are the main methods used in the code. The other methods in the code are generated by the NetBeans IDE's Swing GUI builder and are responsible for setting up the user interface components.

### *MenuPage.java*

### initComponents(): This method is responsible for initializing the form components and setting their properties such as size, position, text, and event listeners. It is automatically generated by the NetBeans IDE's GUI builder based on the design of the form.

### jLabel2MouseClicked(): This method is an event handler for the mouse click event on the "GO BACK" label. When the label is clicked, it creates an instance of the LoginPage class, makes it visible, and disposes of the current MenuPage instance, effectively going back to the login page.

### jButton1ActionPerformed(): This method is an event handler for the action performed on the "Data Entry" button. When the button is clicked, it creates an instance of the DataEntry class, makes it visible, and disposes of the current MenuPage instance, allowing the user to access the data entry functionality.

### jButton2MouseClicked(): This method is an event handler for the mouse click event on the "Departments" button. When the button is clicked, it creates an instance of the LibDepartment class, makes it visible, and disposes of the current MenuPage instance, allowing the user to access the library departments.

### These methods are automatically generated and provide the necessary functionality to handle user interactions with the graphical user interface (GUI) components. They create and manage instances of other classes, control the visibility of different windows, and allow the user to navigate between different parts of the application.

### *NewspaperDep.java*

### initComponents(): This method is responsible for initializing the form components and setting their properties such as size, position, text, and event listeners. It is automatically generated by the NetBeans IDE's GUI builder based on the design of the form.

### subActionPerformed(): This method is an event handler for the action performed on the "ok" button. When the button is clicked, it retrieves the selected item from the news combo box using news.getSelectedItem().toString(), sets the text of the sub button to the selected item, and displays a message dialog using JOptionPane.showMessageDialog() to indicate that the newspaper has been issued.

### jLabel3MouseClicked(): This method is an event handler for the mouse click event on the "GO BACK" label. When the label is clicked, it creates an instance of the LibDepartment class, makes it visible, and disposes of the current NewspaperDep instance, allowing the user to go back to the library department selection.

### These methods provide the necessary functionality to handle user interactions with the graphical user interface (GUI) components. They retrieve user input, display information, and control the flow of the application based on the user's actions.

### *chemdep.java*

### initComponents(): This method is responsible for initializing the form components and setting their properties such as size, position, text, and event listeners. It is automatically generated by the NetBeans IDE's GUI builder based on the design of the form.

### jLabel2MouseClicked(): This method is an event handler for the mouse click event on the "GO BACK" label. When the label is clicked, it creates an instance of the LibDepartment class, makes it visible, and disposes of the current chemdep instance, allowing the user to go back to the library department selection.

### subActionPerformed(): This method is an event handler for the action performed on the "OK" button. When the button is clicked, it retrieves the selected item from the chem combo box using chem.getSelectedItem().toString(), sets the text of the sub button to the selected item, and displays a message dialog using JOptionPane.showMessageDialog() to indicate that the book has been issued.

### These methods provide the necessary functionality to handle user interactions with the graphical user interface (GUI) components. They retrieve user input, display information, and control the flow of the application based on the user's actions.

### *phydep.java*

### initComponents(): This method is responsible for initializing the form components and setting their properties such as size, position, text, and event listeners. It is automatically generated by the NetBeans IDE's GUI builder based on the design of the form.

### subActionPerformed(): This method is an event handler for the action performed on the "OK" button. When the button is clicked, it retrieves the selected item from the phy combo box using phy.getSelectedItem().toString(), sets the text of the sub button to the selected item, and displays a message dialog using JOptionPane.showMessageDialog() to indicate that the book has been issued.

### jLabel3MouseClicked(): This method is an event handler for the mouse click event on the "GO BACK" label. When the label is clicked, it creates an instance of the LibDepartment class, makes it visible, and disposes of the current phydep instance, allowing the user to go back to the library department selection.

### These methods provide the necessary functionality to handle user interactions with the graphical user interface (GUI) components. They retrieve user input, display information, and control the flow of the application based on the user's actions.

# **D. Lessons learned**

## **I. Learning about the concepts of OOP**

When Mr Jude started to teach us about the principles of Object-Oriented Programming, I found myself a bit confused at first. After all I suddenly got introduced to new concepts such as encapsulation, abstraction, inheritance, and polymorphism. But, after working at this project, I now understand when to implement these concepts on a program. I feel like I have become a more efficient coder, and I have a better understanding on how classes, packages, attributes, and methods work.

## **II. Error Handling and Debugging**

One of the most frustrating parts when becoming a coder is having to deal with debugging your own program. I feel the same way too, as having to solve a problem that you caused can be quite confusing. I have devised my own way to figure out what went wrong in my code to lead to something not going the way I want to. First, I will check in what line is the error in. Then I will try checking if I have a typo in my code. Maybe I accidentally missed a bracket, comma or a semicolon when writing it. If there is nothing wrong, then there is something wrong with the logic of the code and I will need to check what kind of error is the code having. Usually when running a problematic code, the terminal will give you a description of the error, and usually I will try to fix based on what the problem is based on what the terminal said. If I can’t fix it, I will just copy what the terminal said and paste it on Google to find out more on that type of error. If it is too complicated to fix, I will just try switching the logic of my code.

# **E. Evaluation**

## **I. Does the program work properly?**

This library management system is essentially just a code I copied from the internet, in which I added further improvements such as fixing some buggy things here and there and simplifying the code so that it is more readable. The program works well as a management system and has basically close to no errors that can occur that hasn’t been dealt with. However, the management system is unfinished and still missing a few features here and there which I do plan on adding in the future once I have found the time to. Overall, I would say that this program is good in terms that it reaches its goal as a program.

## **II. Future Improvements that can be done**

The improvements I would like to add to the library management system is a feature to add new books into the system, as well as a reminder feature for books that are overdue that will send a notification to both the librarian and the department or person who is currently borrowing the book, and another feature that I would like to impellent into the LMS is a fee system where the institute or person will be charged based on the amount of time the book has been overdue for.

## **III. Reflection**

With this project I now know my limitations as a programmer, and I wish that I could improve on these limitations. I am satisfied with how my program ended up being, even though I know that there are many improvements that can still be done with the existing program. I will try to continue to work on this program after submitting this report and I will keep improving it so that it can be something that I am proud of. After all this is one of my first times implementing OOP concepts on Java projects and I will remember the lessons I learnt when making this all throughout my life.

# **F. Evidence of working program**







