EIE3320 Object-Oriented Programming using Java

Student: Cheung Tin Long

Student ID: 19055971d

Assignment Name: Lab2 Report (**One Person Group**)

Date: 3/10/2022

Introduction

In this lab exercise, students have to implement a Library Admin System GUI version with JAVA AWT and SWING. There are several outcomes that students can achieve from the exercise. The first outcome is understanding the difference between GUI software and command line systems. Students have to code the system's response when a specific user event happens, such as the User clicking the Exit Button. Second, students can experience how to understand others' ideas from their code instead of reading lecture notes or asking others to explain. The third outcome is that students can learn how to debug and google stuff more efficiently. Different students will face various problems in this lab exercise. Students have to learn new things from the Internet to solve the coding challenge. To understand the framework or the library, students have to read the official document to understand the logic of that library. The following section will discuss the schedule of lab 2, the program structure, and the features' logic.

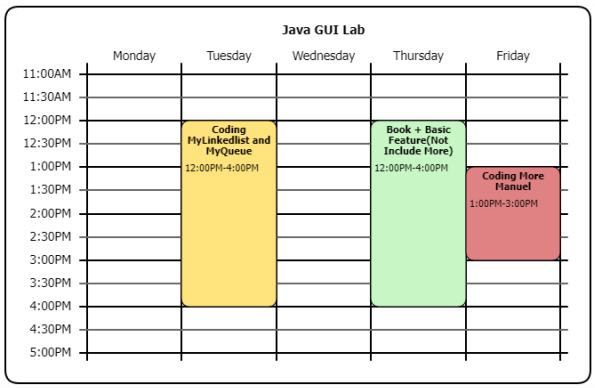
Methodology

- **Work allocation**

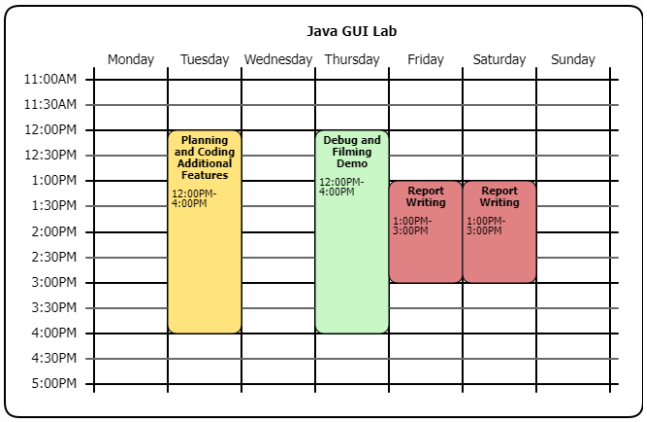
  As this group only has one person, all the work including coding, writing report, and filming demonstration video is done by myself.

- Schedule of Work(Two Weeks)

  Week1:

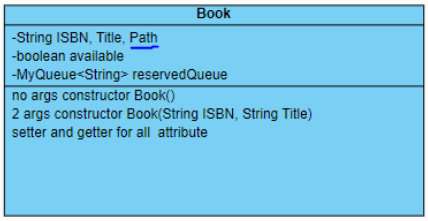


Week2:



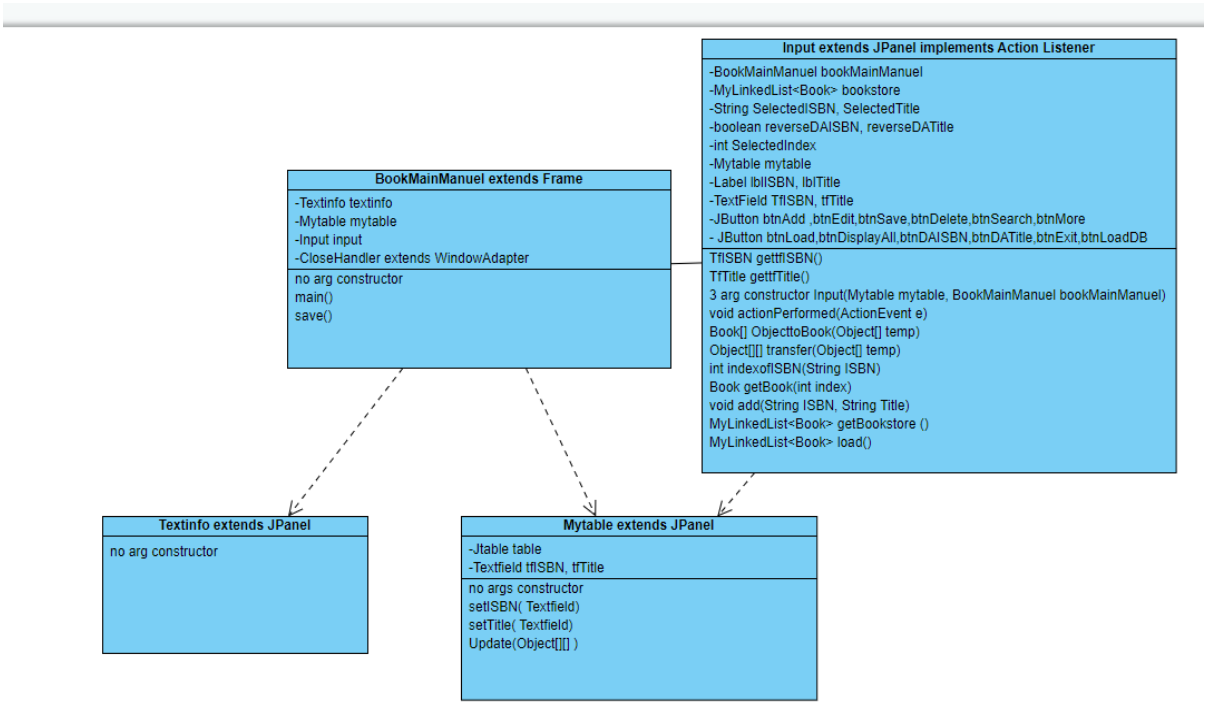
- UML Class Diagram

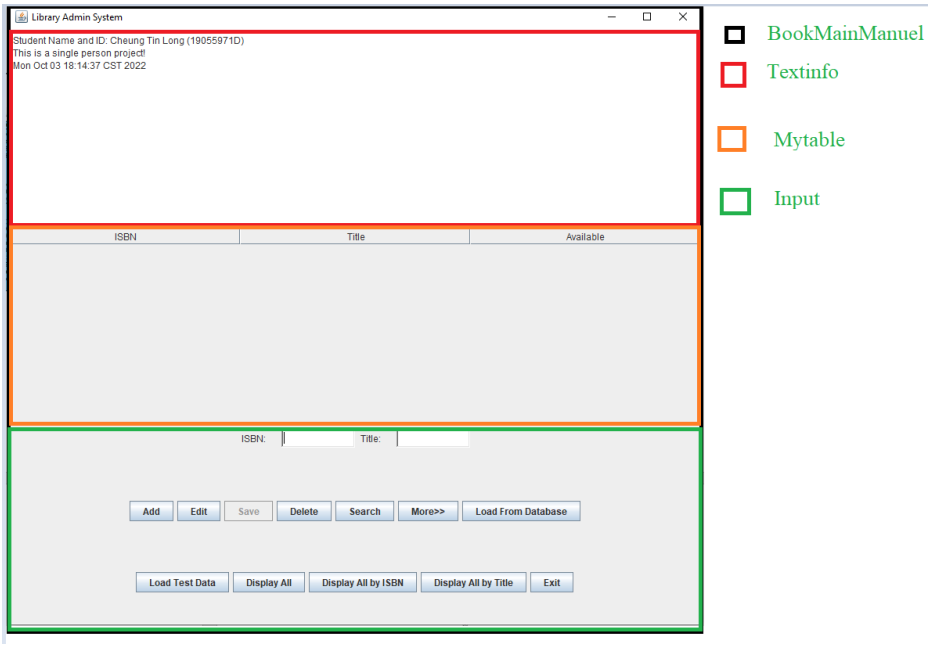
The variable should be private, and the function should be public if not clearly stated.



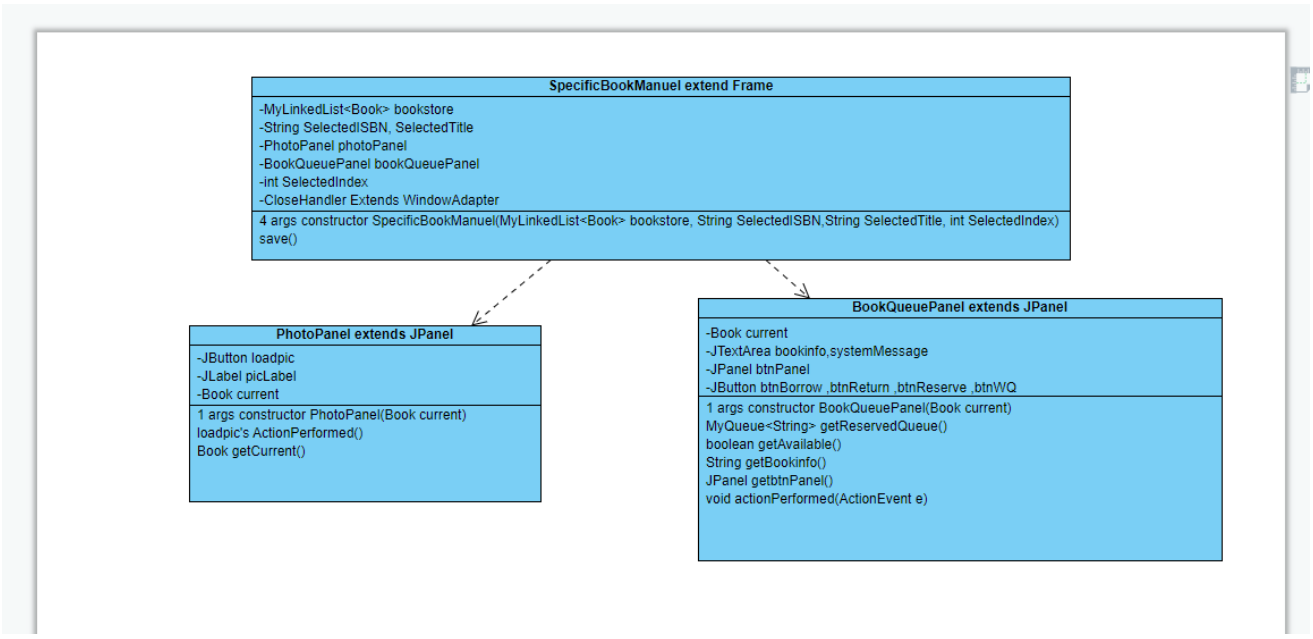
Add new variable **Path** for storing the path of the photo (additional features)

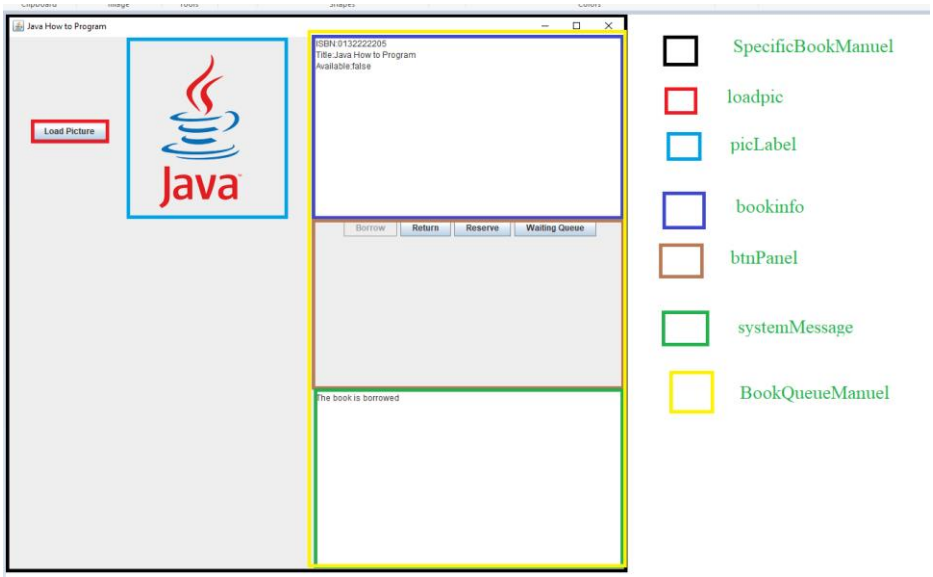
Others follow the requirement.





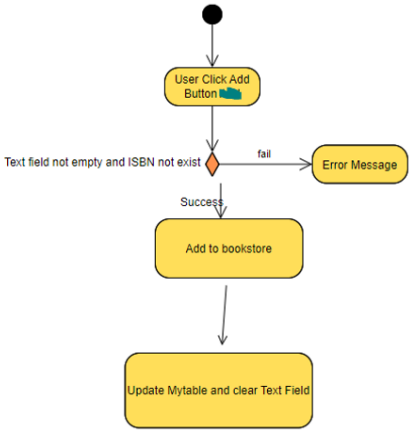
Input contains the logic of most basic features and the MyLinkedList<Book> bookstore. Input has to take BookMainManuel as constructor arguments because when the user clicks the more button, BookMainManuel has to be set to invisible.



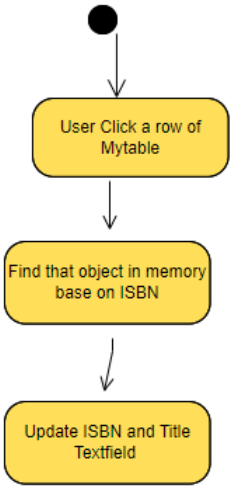


- The flow of execution

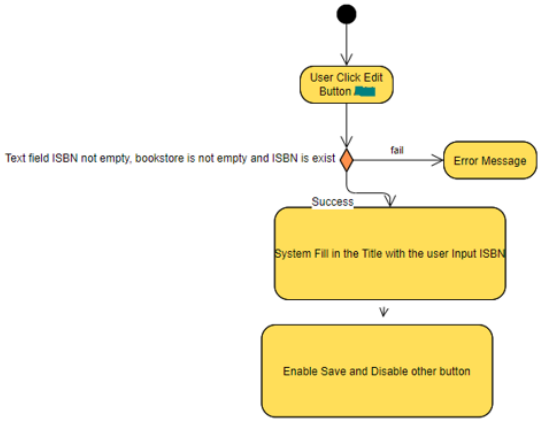
btnAdd:



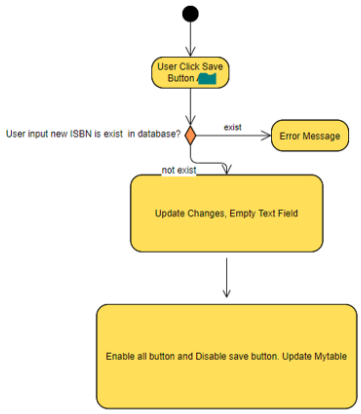
Row Selecting of Table:



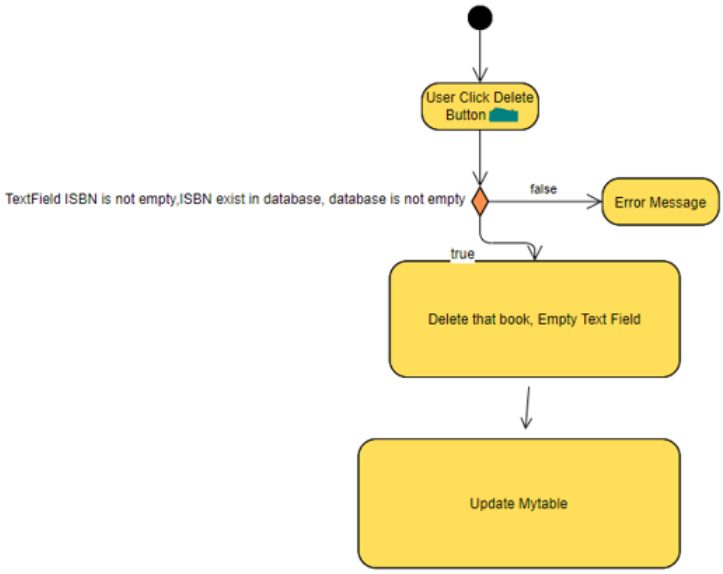
btnEdit:



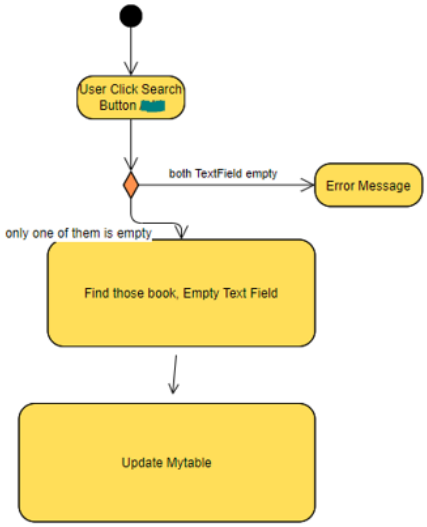
btnSave:



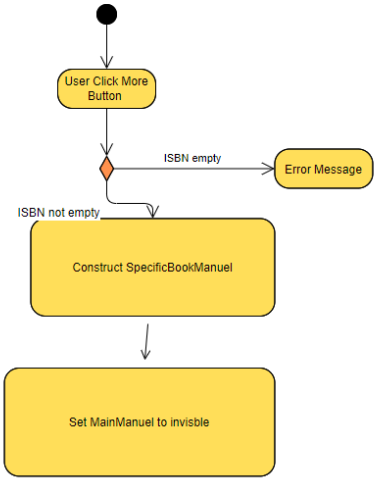
btnDelete:



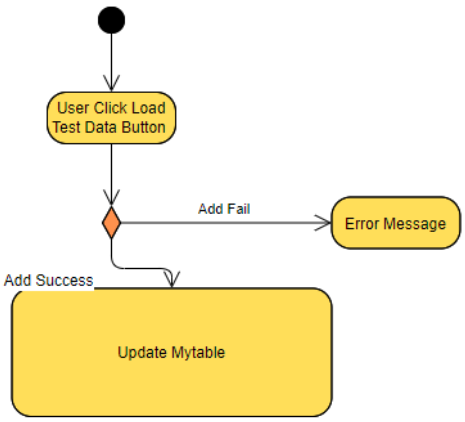
btnSearch:



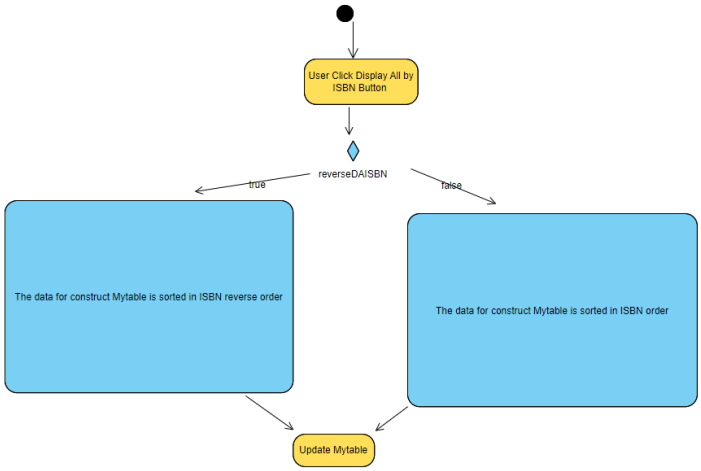
btnMore:



btnLoad Test Data:

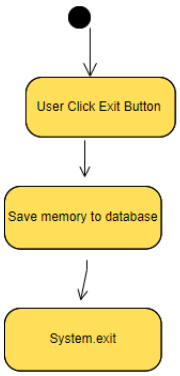


btnDisplayAll by ISBN:

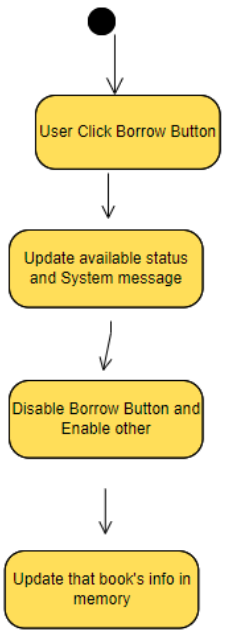


btnDisplayAll by Title <- similar to Display All by ISBN

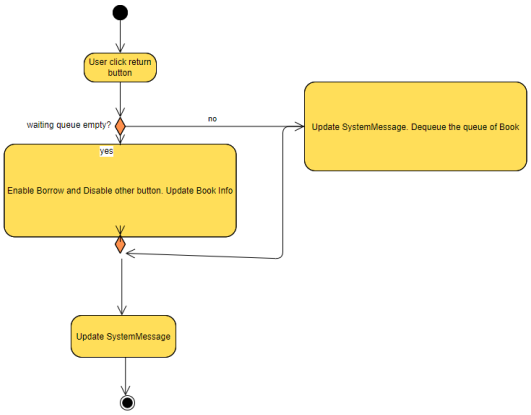
btnExit (Same as click x button):



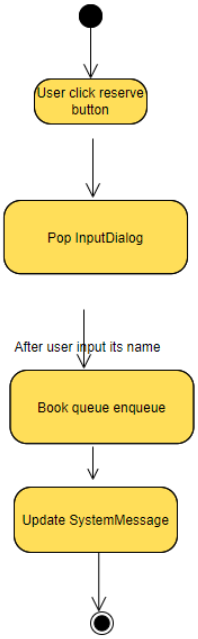
BtnBorrow in More manuel:



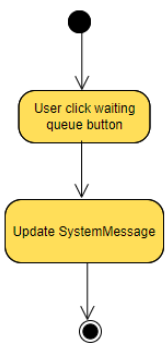
BtnReturn:



BtnReserve:



BtnWaitingQueue:



- Implementation and the use of the linked list and the queue

Linked List

```
/** Create a list from an array of objects */
public MyLinkedList(E[] objects) {
```

Create two Node for the objects[0] and objects[objects.length-1]

Head points to objects[0] and Tail points to other.

Connect the head and tail with for loop

```
Node<E> previous = head;
for(int i=1;i<(objects.length-1);++i){
    Node<E> temp= new Node(objects[i]);
    previous.next=temp;
    previous=temp;
}
previous.next=tail;
size+= objects.length;
```

```
/** Add an element to the beginning of the list */
public void addFirst(E e) {
```

New head.next point to the old head E e.

If size==0 then tail have to point to new head.

++size;

```
/** Add an element to the end of the list */
public void addLast(E e) {
```

Old tail.next point to new tail

Tail point to new tail.

++size;

```
@Override /** Add a new element at the specified index
 * in this list. The index of the head element is 0 */
public void add(int index, E e) {
```

If index=0, addfirst

Index >=size, addLast

Other case, travel to the element position index-1, element.next point to new object

New Object.next point to original position index element.

```
/** Remove the head node and
 *  return the object that is contained in the removed node. */
public E removeFirst() {
```

If size=1, empty list => head and tail point to null

Else head points to head.next

```
/** Remove the last node and
 * return the object that is contained in the removed node. */
public E removeLast() {
```

If size=1, empty the list

Else travel to the element of position list.length-2 and set element.length-2 .next =null;

```
@Override /** Remove the element at the specified position in this
 *  list. Return the element that was removed from the list. */
public E remove(int index) {
```

If index=0, removefirst

Index >=size, removeLast

Other case, travel to the element position index-1, element.next point to position index+1 element.

```
@Override /** Return true if this list contains the element e */
public boolean contains(Object e) {
```

Travel the whole list. If we find object equals to e, return true. After travelling the whole list, we return false

```
@Override /** Return the index of the last matching element in
 *  this list. Return -1 if no match. */
public int lastIndexOf(E e) {
```

Travel the whole list. If we find object equals to e, save the position. After travelling the whole list, we return the final saved value.

```
@Override /** Replace the element at the specified position
 *  in this list with the specified element. */
public E set(int index, E e) {
```

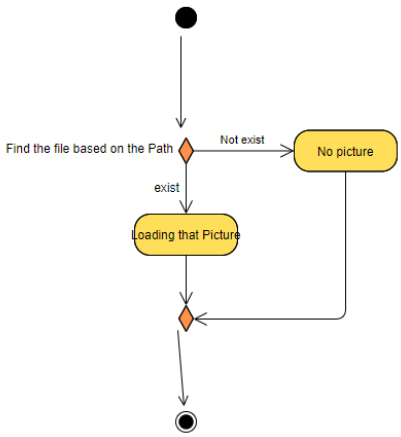
Tavel to the that position, change Node.element to e

The use of LinkedList is to be the core of MyQueue and store Book objects. MyQueue uses the function from LinkedList. The purpose of MyQueue is to enforce users to only use addLast(enqueue) and removeFirst (dequeue).In this project, we use Myqueue for the waiting queue function.

Additional features implemented

Image Retrieval

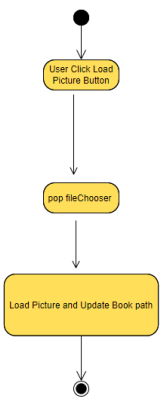
PhotoPanel Initialization:



After user clicking more button, the system will check the file exist or not based on the path. If yes, it will load the photo

```
File tempFile = new File(current.getPath());
if(tempFile.exists()){
    //System.out.println("Getting photo based on db");
    try {
        BufferedImage picture = ImageIO.read(tempFile);
        picLabel.setIcon(new ImageIcon(picture));
```

PhotoPanel Load Picture Button:



User can click the Load picture button to choose new picture for that book.
We use JFileChooser to get SelectedFile. Then we will save the selectedFile path to the book's path.

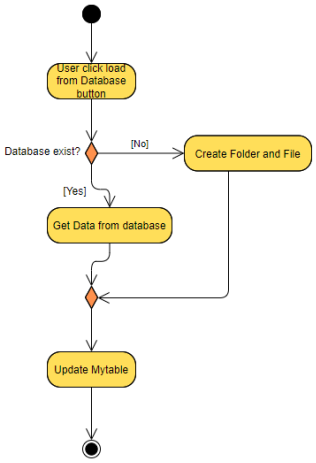
```
JFileChooser fileChooser = new JFileChooser();
int returnValue = fileChooser.showOpenDialog( parent: null);
if (returnValue == JFileChooser.APPROVE_OPTION)
{
    File selectedFile = fileChooser.getSelectedFile();
    //System.out.println(selectedFile.getAbsolutePath());
    try {
        BufferedImage picture = ImageIO.read(selectedFile);
        picLabel.setIcon(new ImageIcon(picture));
        add(picLabel);
        current.setPath(selectedFile.getAbsolutePath());

    }catch (IOException ioe) {
        ioe.printStackTrace();
        JOptionPane.showMessageDialog( parentComponent: null,  message: "ERROR");
```

Data Management:

For the ease of use(reduce dependency), we use file to store data instead of using database like Mysql.

User can load the database data through clicking the button Load From Database.
Button Load From Database in Main Manuel:



If file is not exist, we will create folders and file
Tree view:
C drive ---- library -+----database ---- store.txt
L-----static(user can store photo in this folder)
If file exist, we will create each Book Object based on each line of store.txt .

When the user click exit button or x button, the bookstore data will write to store.txt

The format of store.txt:

```
ISBN`! title `! available `!reservedQueue`!Path
```

`! is the separator

Example:

```
0131450913`!HTML How to Program`!1`!e`!e
0131857576`!C++ How to Program`!1`!e`!e
0132222205`!Java How to Program`!0`!Delon,Tim,User3`!C:\Users\Delon\Downloads\Java_logo.png
```

If available = 1, means true

Else if available= 0, means false

If reservedQueue =e , means empty queue

Delon,Tim,User3 = (Head) Delon -> Tim -> User3 (Tail)

reservedQueue separator is comma

If path =e , means empty path.

We use absolutePath as path.

Benefit to use separator: Easy to load the data

```
while ((line = reader.readLine()) != null) {
    //System.out.println("Loading from database:"+line);
    Book temp = new Book();
    String[] data = line.split( regex: "`!");
    //System.out.println("data 0:"+ data[0]);
    temp.setISBN(data[0]);
    //System.out.println("data 1:"+ data[1]);
    temp.setTitle(data[1]);
    //System.out.println("data 2:"+ data[2]);
    temp.setAvailable(data[2].equals("1"));
    //System.out.println("data 3:"+ data[3]);
    String[] queue = data[3].split( regex: ",");
    MyQueue<String> myQueue = new MyQueue<>();
    if(!(queue.length==1 && queue[0].equals("e"))){
     for(String names : queue){
         myQueue.enqueue(names);
     }
    }
    temp.setReservedQueue(myQueue);
    //System.out.println("data 4:"+ data[4]);
    if(!(data[4].equals("e"))){
        temp.setPath(data[4]);
    }
```

Program Testing

Demo Video(If both link can't view it, you can download it through the google drive link):

GoogleDrive:

https://drive.google.com/file/d/1QPSg1JbO7yTJN6ow1s_pFkcCEajOUVl6/view

OneDrive:
[Lab2Demo.mp4]

Conclusion

After working on this lab exercise, I have learned several things. The first thing that I learned is to code slowly. If we put more times on thinking the logic of the algorithms, we can reduce the time of debugging. Sometimes the problem is caused by one line of code. The second thing that I learned is what is the difference between normal job and software engineer. Software engineer have to learn new things from the Internet and apply those knowledges to solve the current problem. The third thing that I learn is the importance of googling. There are lots of difficulties that we are facing in this lab. The lecture notes can't solve all the problems but the Internet can if we enter the correct key work.

Future Development

1. Make a website version of Library Admin System.
   The Web version and the GUI version should use the same database.

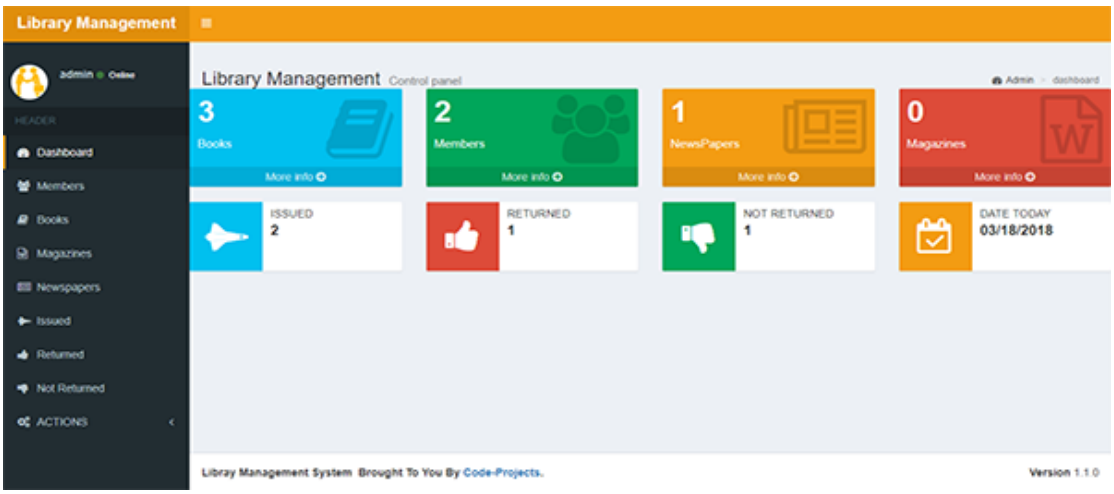


   Photo As example

2. SMS or email remainder system:
   The system should send the email or SMS to tell the user about the book information or other important thing.

Inbox (3775 Messages)

Jobber
To: Robin Schneider
Subject: Visit Reminder

Hi Robin Schneider,

Just a friendly reminder that we have an upcoming appointment.

Jan 10, 2018
Service Call
289 NW 198th St,
Shoreline, WA 98177, USA

If you have any questions or concerns, please don't hesitate to get in touch with us at info@questprovider.com

CLICK LINK BELOW TO CONFIRM YOUR APPOINTMENT:

Confirm Visit

Sincerely,
Quest Provider

3. Support Multi Language