ENSF 593/594 – Principles of Software Development

Fall 2019

Lab Assignment #7: GUI and Event-Handling

Due Dates	
Lab	Submit electronically on D2L. **before 11:59 PM on Wednesday November 6th**

This is an individual assignment.

The objectives of this lab are:

1. Java Graphical User Interface
2. Event-Handling
3. Applications with GUI

<u>The following rules apply to this lab and all other lab assignments in future:</u>

1. Before submitting your lab reports, take a moment to make sure that you are handing in all the material that is required. If you forget to hand something in, that is your fault; you can't use `I forgot' as an excuse to hand in parts of the assignment late.
2. **<u>20% marks</u>** will be deducted from the assignments handed in up to **<u>24 hours</u>** after each due date. It means if your mark is X out of Y, you will only gain 0.8 times X. There will be no credit for assignments turned in later than 24 hours after the due dates; they will be returned unmarked.

Lab (33 marks)

Exercise 1: An application to Maintain Student Records (11 Marks)

Introduction:

The purpose of this exercise is to get familiar with the Java Graphics User Interface (Java GUI).

In this exercise, you will write a GUI application that manages a binary search tree that maintains the student records. Each record includes the student's: id, faculty, major, and his/her year of study.

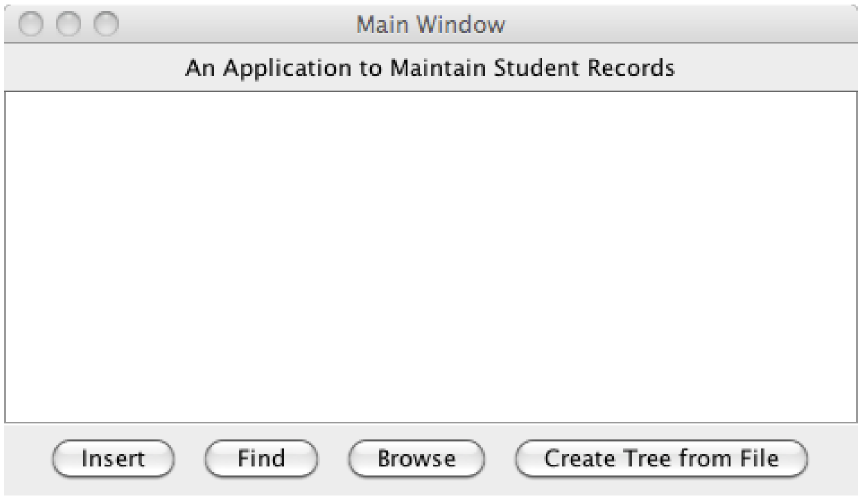
Most of you are most likely familiar with the binary search trees, which is a data structure like a linked list to organize and maintain the data. Where each node in a tree has two pointer pointing to a node on the left and a node on the right. The starting node is the root-node and the nodes with no child at the lower edge of the tree are called leaf-nodes.

For this exercise, you don't need to know more details about a binary search tree, because the entire code related to the binary search tree is given to you and you can download them from D2L. The name of the files to be downloaded are: `Node.java`, `Data.java, and BinarySearch.java`. In addition to these files you have been also supplied with another file called `input.txt`. This is a text file that you can open and browse. It contains several student records and your program should use it to populate a binary search tree**.** The first column of records in this file is the student id, second column is the student's faculty (for example EN for engineering), third column is the student's major (for example ENCM), and finally the last column is the student's year of study (for example 2nd or 3rd, etc.).

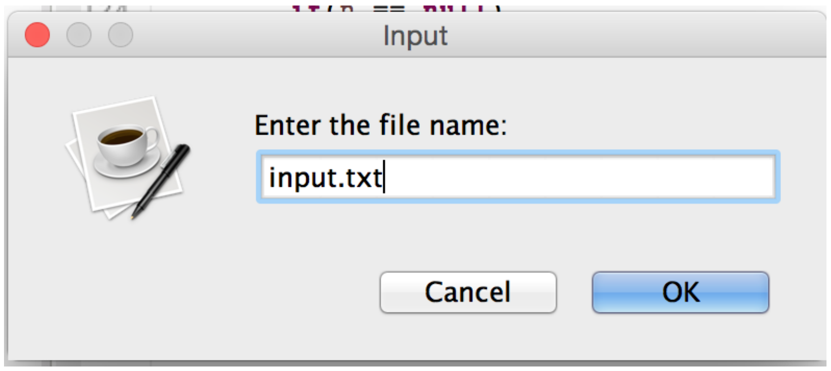
The Java GUI application that you are going to write must have the following functionalities:

- It should be able to read the student information from a given text file (input.txt) and build a binary-search tree into the program memory.
- It should be able to display the records of students from the tree in memory into a JTextArea, on the computer screen.
- It should be able to prompt the use to enter new student records and insert it into the tree.
- And finally, it should be able to search for a student by their id number and if such a student should exist, display their information on the screen.

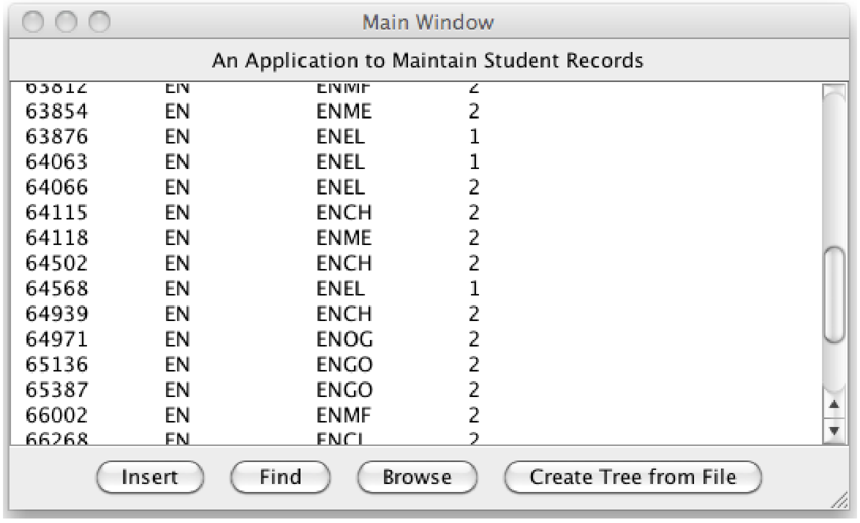
To give you a better idea about how this program runs, here is a sample run with a few screenshots.



1. When the "Create Tree from File" is pressed, the following dialog box appears that asks you to enter the name of an existing input file. If you enter input.txt, the given input file will be used to create the tree:



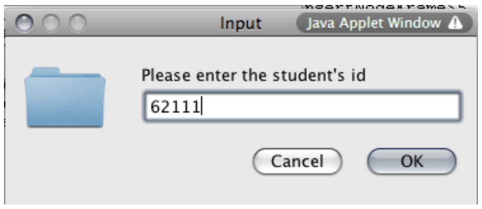
2. Now if one presses the "Browse" button in the main frame, the records from tree will be displayed in the text area on the main frame:



An Application to Maintain Student Records

```
63812   EN   ENMF   2
63854   EN   ENME   2
63876   EN   ENEL   1
64063   EN   ENEL   1
64066   EN   ENEL   2
64115   EN   ENCH   2
64118   EN   ENME   2
64502   EN   ENCH   2
64568   EN   ENEL   1
64939   EN   ENCH   2
64971   EN   ENOG   2
65136   EN   ENGO   2
65387   EN   ENGO   2
66002   EN   ENMF   2
66268   EN   ENCL   2
```

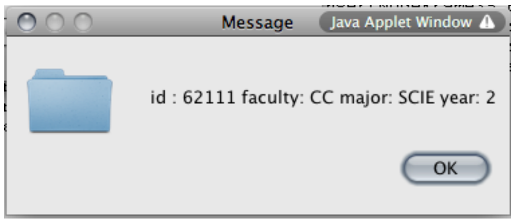
3. If now "Find" button is pressed, the user can enter the id number of a student to be searched for:



Please enter the student's id

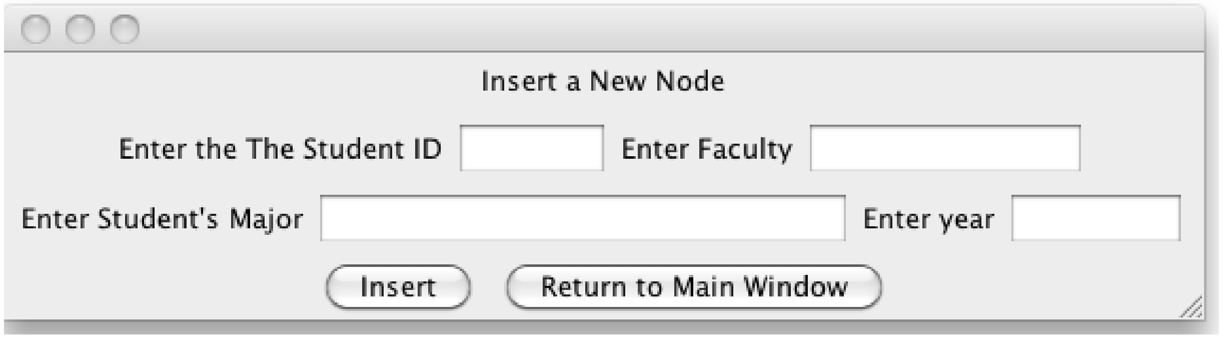
62111

4. When "OK" is pressed, if the student exists in the tree, the full information (student's id, faculty, major, and year) will be displayed. Otherwise give a message that the target record was not found.



id : 62111 faculty: CC major: SCIE year: 2

5. When Insert is pressed the following frame or dialog box should appear and allow the user to enter the information (student id, faculty, major, year):

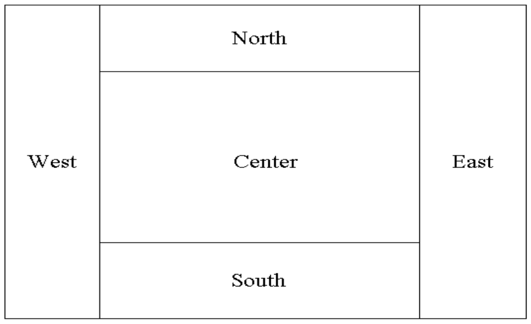


Then a new node will be created and inserted into the tree.

Brief Notes About the Frame Layout.

A frame is a window that has a border, a title, and may contain buttons, text fields, or other GUI components. GUI applications usually use at least one frame. You should derive your own Frame class from Swing library class called Frame.

As explained during the lectures, there are several types of layout that can be used in a Java frame. In this example you will see a border layout that you should use it this exercise. It divides the frame in five areas: north, south, east, west, and center. This is actually the default layout for frames and some other containers. See the following figure:



Then, you can create three panels with the necessary components such as buttons, text fields, text areas and then you should add them to the north, center and the south side of the frame.

The panel on the center is supposed to be a scroll panel that will have a scroll bar, if the number of lines of the text exceeds the height of the panel. Inside the scroll panel we add a text area that can display data. In this exercise, you should disable the edit functionality of the text area, so that it can only be used for displaying.

Also, using the Java GUI component called JLabel, you can add a title into the north panel, and inside the south panel you can add three buttons labeled as **Insert, Find, Browse, and “Create Tree from File”**:

Reminder: to add panels to your container frame, you should get a handle to the content pane. This can be done by a function, called **getContentPane().** This function is inherited from class JFrame and returns an object of Java class **Container**.

Also, remember that each button should be registered with an ActionListener and should implement an **actionPerformed** method**.** This allows a component such as button to handle an event (see the details in the event handling section, below).

For more details please refer to your lecture notes and posted slides on the blackboard.

What to Submit: Please submit all the java files including the files you have created/modified in a zip folder. You need to provide the Javadoc comments in your code, but you don’t need to submit the HTML files.

Lab Exercise 2: Tic-Tac-Toe with GUI (12 Marks)

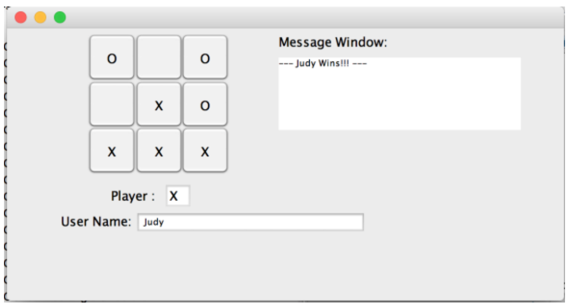
Your task is to create a GUI that allows players to use their mouse to mark the board with an X or O.

Here are the minimal functionalities of the component of your window/frame must have:

- A presentation of the board
- A text area that displays the game messages for the communication or information exchange.
- A text box that players can enter their name.

Note: You have complete freedom to design your GUI for this game differently. You may also choose to have additional components/events in your GUI.

Here is a screenshot of the GUI I had for x-player and o-player:

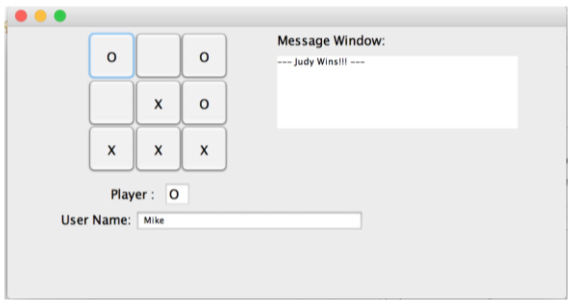


Message Window:

--- Judy Wins!!! ---

Player : X

User Name: Judy



Message Window:

--- Judy Wins!!! ---

Player : O

User Name: Mike

What to Submit: Please submit all the java files including the files you have created/modified in a zip folder. You need to provide the Javadoc comments in your code, but you don't need to submit the HTML files.

How to submit: Include all your files for the lab section in one folder, zip your folder and upload it in D2L before the deadline.