ENSF 409 – Principles of Software Development
Summer 2021

Term Project

Due Dates	
Pre-Project Exercise	Submit electronically on D2L **before 1:00 PM on Friday August 6th.**
Demo	In Lab session of **Friday August 6th.**
Project:	Submit electronically on D2L **before 1:00 PM on Monday August 9th.**

This is a Group assignment. Students will work in groups of Two as per the groups created for assignment submissions throughout the semester. Please note, students have the option of working on their own.

Project objectives

1. To gain experience with the design and development of a relatively larger software project.
2. To gain practical experience in programing concepts introduced in ENSF 409 such as object-oriented design and development, client-server architectures, multi-threading, JDBC, and GUI.
3. To gain experience with iterative development process.
4. To develop teamwork and presentation skills as a programmer.

Instructions groups

Students who work in groups, must follow the following guidelines and regulations:

1. The workload **must be divided equally** between group members.
2. Groups must present their work to the TA in the Lab and both members of the group should be present.
3. Only one of you needs to submit the final project on D2L by the due date. However, please make sure all of your names are included in your submission.
4. Teams must use Git to manage versions of their project.

Important Advice: When working as a team, make sure to clearly define your goals and expectations, as well as internal deadlines. Ensure to communicate regularly and schedule times to meet and work together.

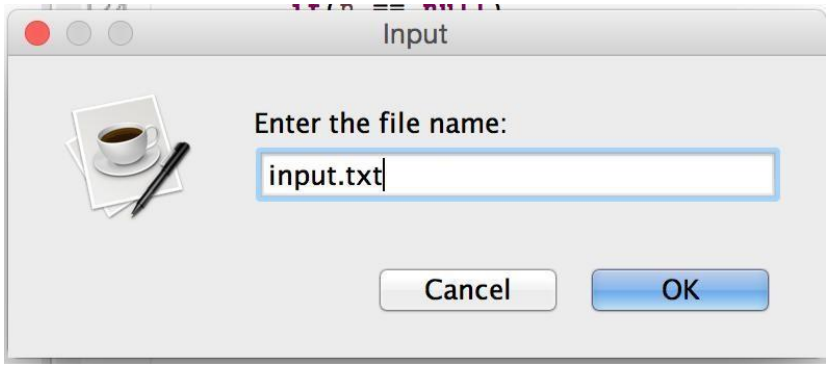
Pre-Project Exercise - 1: An application to Maintain Student Records (~~15~~ 25 Marks)

Introduction:

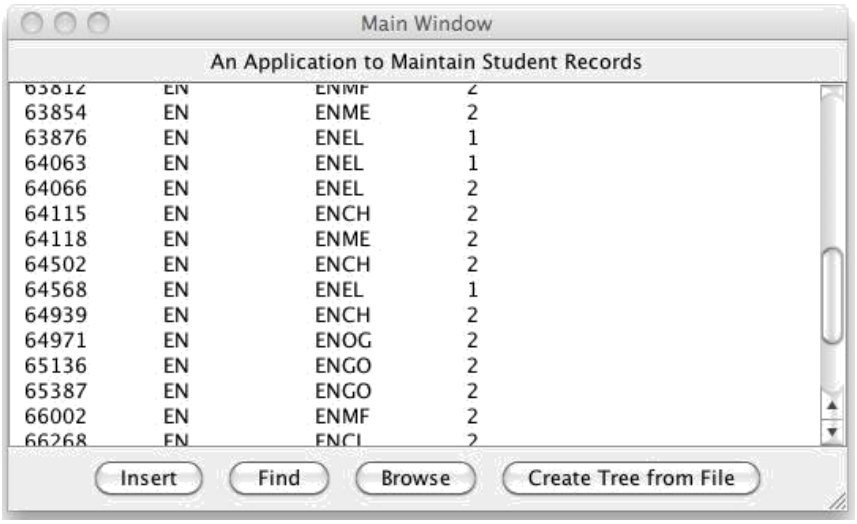
The purpose of this exercise is to get familiar with the Java Graphics User Interface (Java GUI).
In this exercise, you will write a GUI application that manages a binary search tree that maintains the student records. Each record includes the student's: id, faculty, major, and his/her year of study.
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 pointingto 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. You will be using the implemented binary-search tree provided to create your tree and insert information in to it.
It should be able to display the records of students from the tree in memory intoa JTextArea, on the computer screen.
It should be able to prompt the user to enter new student records and insert it intothe tree.
And finally, it should be able to search for a student by their id number and if sucha 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 fewscreenshots.

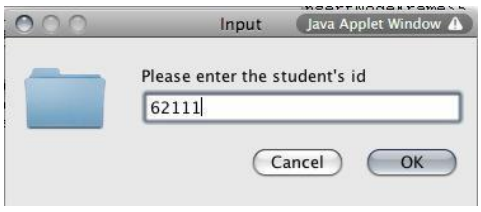
1. When the "Create Tree from File" is pressed, the following dialog box appears that asks you to enter the name of an existing input file. If you enter input.txt, the given input file will be used to create the tree:



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:

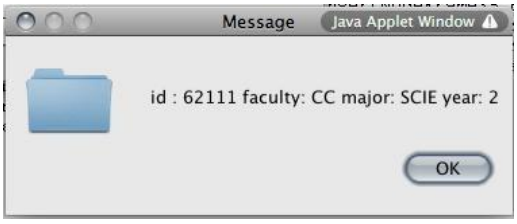


3. If now "Find" button is pressed, the user can enter the id number of a student to be searched for:

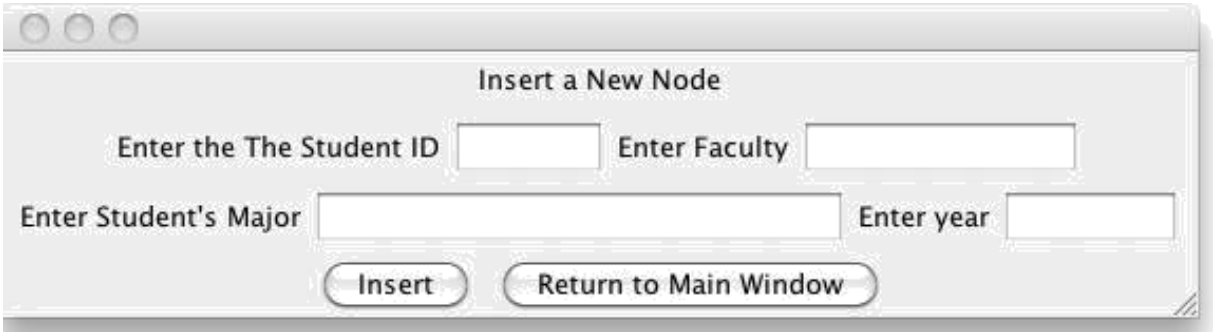


4. When "OK" is pressed, if the student exists in the tree, the full information (student's id, faculty, major, and year) will be displayed. Otherwise give a message

that the target record was not found.



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):

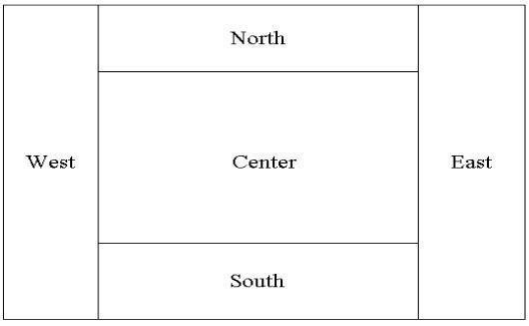


6. Then a new node will be created and inserted into the tree.

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 in 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.

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 and the Javadoc created folder for documentation.

Software Design and Development – Course Registration System (~~70~~ 60 Marks and up to 25 bonus marks)

This project aims to closely simulate the design and development of a real-life software project.For a given software system, you will be given a set of functional requirements, as well as a set of constraints that your design and developed code must adhere to. In the interest of time, certain functionalities have been omitted or simplified. (See notes on this throughout the requirement description). The other notable difference between what is asked in this project and real-life projects is the absence of automated testing. In the interest of time, you are not asked to write J-Unit tests for your application.

Important Note:

This project provides you with the opportunity to use the skills and knowledge you have developed in this course and develop a solution for a problem.
As such you have been presented with a problem statement and no other directions. It is on you to research, design, develop and present your project, as you will have to do throughout your careers as software engineers.

Requirement Description

You have been asked to develop a client-server application for the Course Registration System that you designed and developed in Lab 3.

Your main tasks in this project are:

1. Architecture change – Currently this application is a standalone desktop application. Your task is to change its architecture into a distributed system using the client-server architecture.

2. Concurrency – Multiple users should be able to use your application.

3. GUI – Currently this application has a console-based interface. You will replace it with a GUI. Note: The GUI will be on the client side.

4. Database – You will have an SQL database to maintain a list of students and courses. / or you can use files for information input using the RandomAccessFile class

Functional Requirements:

The functionality of this system is outlined in Lab 3 – Exercise 2. No other functionality needs to be added. However, the new system must adhere to constraints outlined in the following section.

Constraints

Your design must adhere to the following constraints

1. This system must be developed as a client-server application.
2. The client side of this application must have a GUI. The GUI of the system must be organized, and user-friendly. You are strongly urged to do some research and look at similar online systems to come up with ideas for the flow of your GUI.
3. The server must be able to support multiple clients using it at one time.
4. The system must include a database which is populated with the records as explained above.
5. ~~Your design must closely follow the principles of low coupling and high cohesion. You are strongly encouraged to make use of the MVC design pattern (This will be discussed in class).~~

Possible Bonus Marks (Up to 25):

1. Deploy the project; run the server and client on separate machines **(5 marks)**
2. Writing JUNIT tests for at least 2 classes **(5 marks)**
3. Maintain a list of users and develop login/out feature **(5 marks)**
4. Create a separate GUI for an "Admin" with the functionality of creating new courses **(10 marks)**

If you would like to add another feature instead of the ones listed above, let us know! You can get bonus marks for your creativity! Think of a feature to add and you can get some bonus marks depending on the complexity of the future. **Note: it is not possible to get more than 25 bonus marks.**

What to submit:

Design (20 marks):

1. Submit your complete "Design Class Diagrams" in a pdf file on D2L by the due date mentioned above
2. Your UML will show the modifications you have made to switch the program to a distributed system. For example, now that you have a class representing the server and another representing the front-end for the client you need to add these to your UML.

Note: DO NOT show process diagram or threads on your UML for simplicty

~~3. You must clearly demonstrate which classes belong to which packages in your class diagram.~~

~~Important Hint: Depending on your design, it may be more appropriate to have several class diagrams:~~

- ~~One high-level class diagram that shows all packages and the classes within those packages as well as their interactions with other packages. This diagram will give the full design of your system at one glance.~~
- ~~Several low-level class diagrams for each package which show more details about the classes inside each package.~~

3. If you are working with one or two partners, create a cover page (PDF format) with your name and your partner name and submit it the D2L.

~~**NOTE: Your lower level class diagrams must include all attributes and important operations (i.e. not getters and setters) as well as cardinality**~~

Implementation (45 marks and up to 25 bonus marks):

1. A "Read-me" file outlining the following:
   - Full names and emails of all group members
   - Instructions on how to run your application
   - A clear list of bonus features implemented
2. Compress your project folder into a .zip file and upload it to D2L before the deadline

Note: Make sure the directory contains all files and directory in your zipped file (test it).If files are missing or corrupted, your project may be returned unmarked.

3. Submit all HTML files for Javadoc as part of your project compressed file, include the files in a subfolder named documentation(NOTE: 10 marks are allocated to havingproper Javadoc)