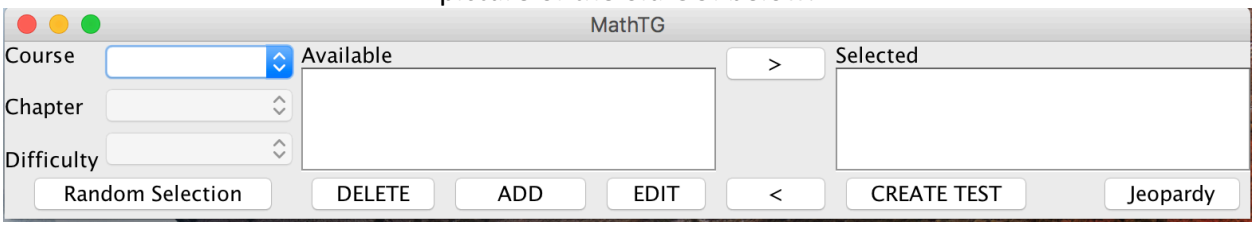
MathTG – Final Progress Report
11/30/17
Client: Luis A. Cueva-Parra
Developers:
Jonathan Miller
Robert Redus
Brian Long

Our part of the project is to build or modify a Graphic User Interface (GUI) which will be displayed to the user the steps needed to make a Test or Jeopardy Board.

(Jonathan Miller) Overall Concept:

Introduction

The first thing we did as a group was to look at the previous semesters Graphic User Interface (GUI) to see whether we should modify or start from scratch. As you can see below this is a picture of the old GUI below.



First thing that my group noticed right off the bat was that the old GUI seem to be bit over complicated. So, we ultimately decided to make our GUI from scratch and make it simpler than the previous GUI.

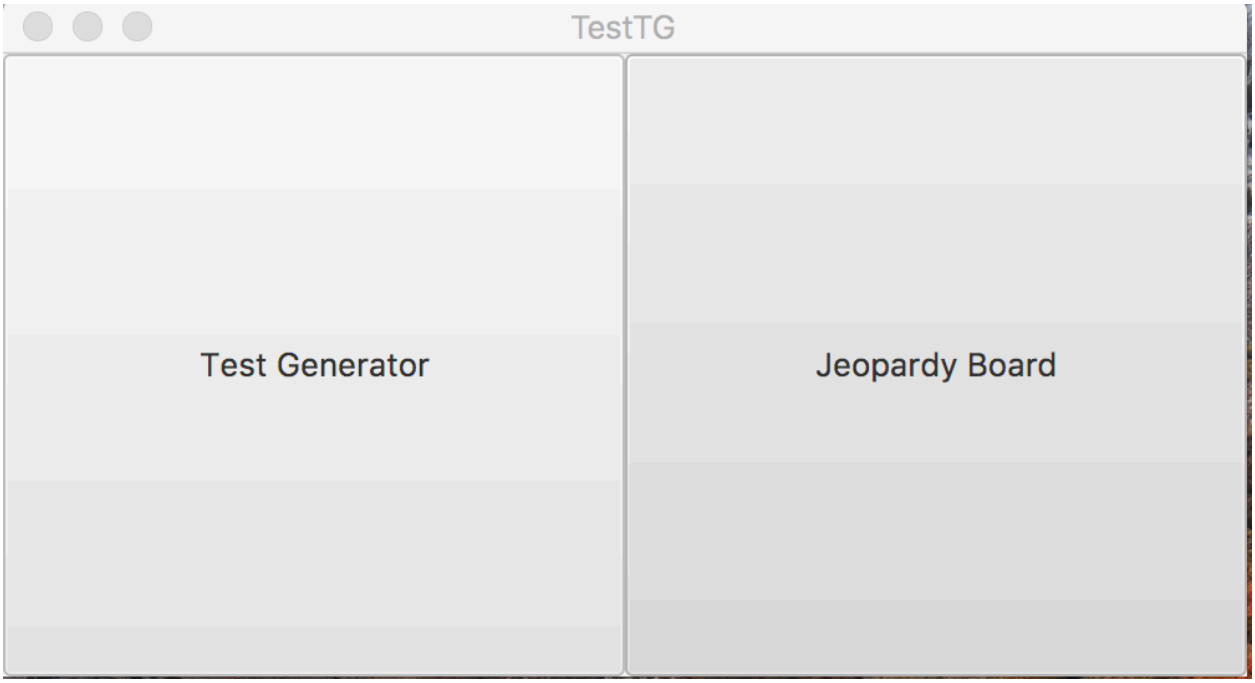
Basic Design

Since we decided to start from scratch we build a basic concept of what we wanted our GUI to look like and how it should operate. Below is our basic design that we are modeling our GUI on.

Phase 1:

Will consist of two buttons were Button 1 (Test generator) will lead to another window with the options to make the test, and the other button (2) will open another window with the jeopardy board options (jeopardy might change this).

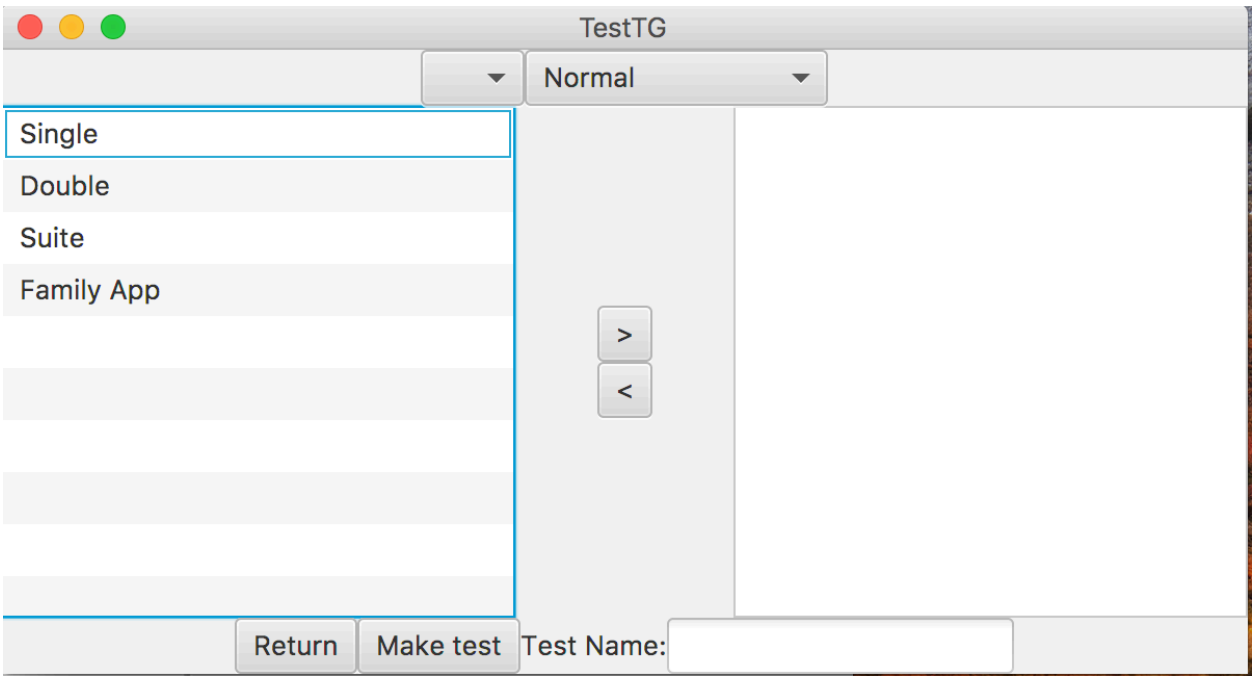
Final product: No changes to our initial concept .



Phase 2 (Test Generator)

This window will consist of two list views a couple of buttons, text view, and some dropdown menus(More detail in the Component section). The user will select a subject from the database and the chapter will appear in the left list view, after the user selects a chapter they will be able to move it over to the other list view and vice versa, with the two arrow buttons in the middle of the GUI.

Final Product: This is our outcome that we finished the semester with. It is quite similar to our basic discussion during the semester.



(Above)End Phase
We will have a return button and a Make Test which will also be in the Test Generator window.

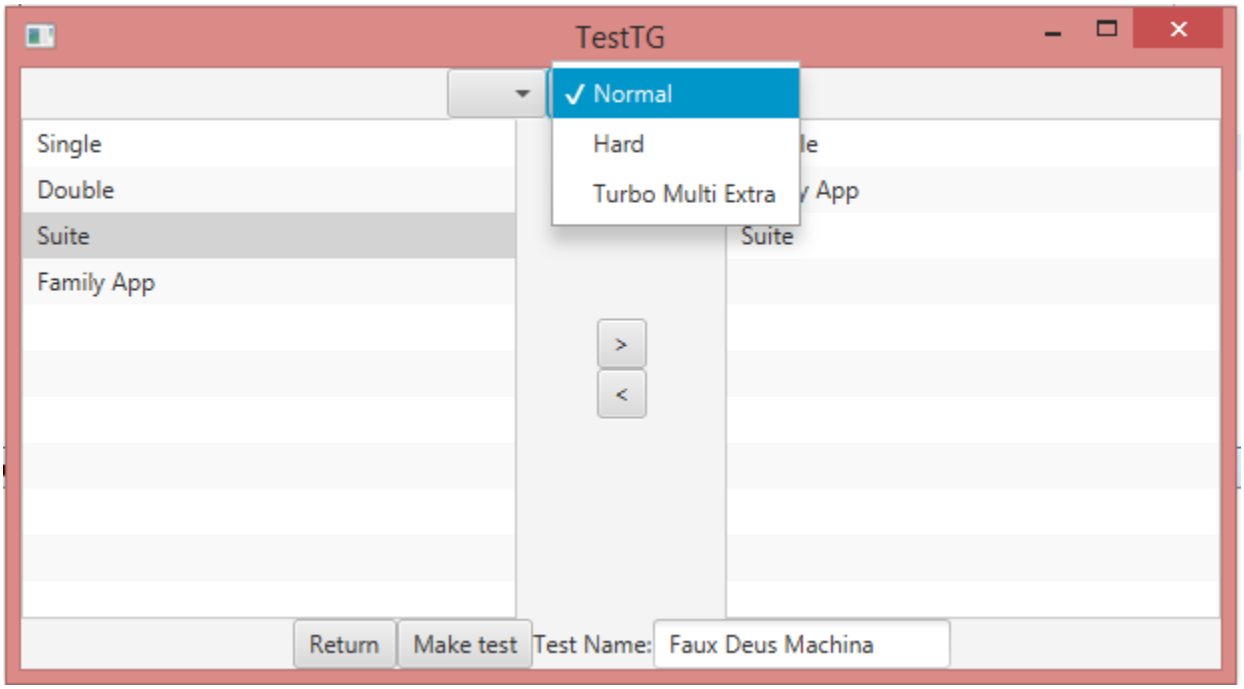
(Bryan Long) Component 1:

In this Project I, Brian Long, was assigned to the 4th group that dealt with the GUI. In the beginning I had not done much because my knowledge of Java was non-existent at the time. The few times my group met during this time, we dissected the GUI of predecessors to try and gleam any useful information. The code we had inherited was really just a blank GUI with nearly no functionality. The one thing it could do was create a document. However, the code we had inherited didn't suit our tastes. So from that point we decided to build from the ground up with new code so that everything would be nicer. Later on, I received the role of managing the component with the options on it. Using Netbeans I slowly learned how to use the GUI builder in Java applications. Even though I was still inexperience with java I believe I created a decent looking Frame that including a list of subjects, a number spinner for the amount of questions, a list for the chapters, a combo-box for the difficulty, and toggle boxes for the output buttons. This was the first prototype of the component of Options. I was going to begin adding methods to the components but realized that I still didn't understand Java well enough to do it on my own, so I consulted my team. At this time we were convinced that using a GUI to build our GUI was not the route we wanted to take so all of us decided to study JavaFX and rebuild the GUI. After rebuild the project we decided to add functionality as we went. Starting with the Chapter

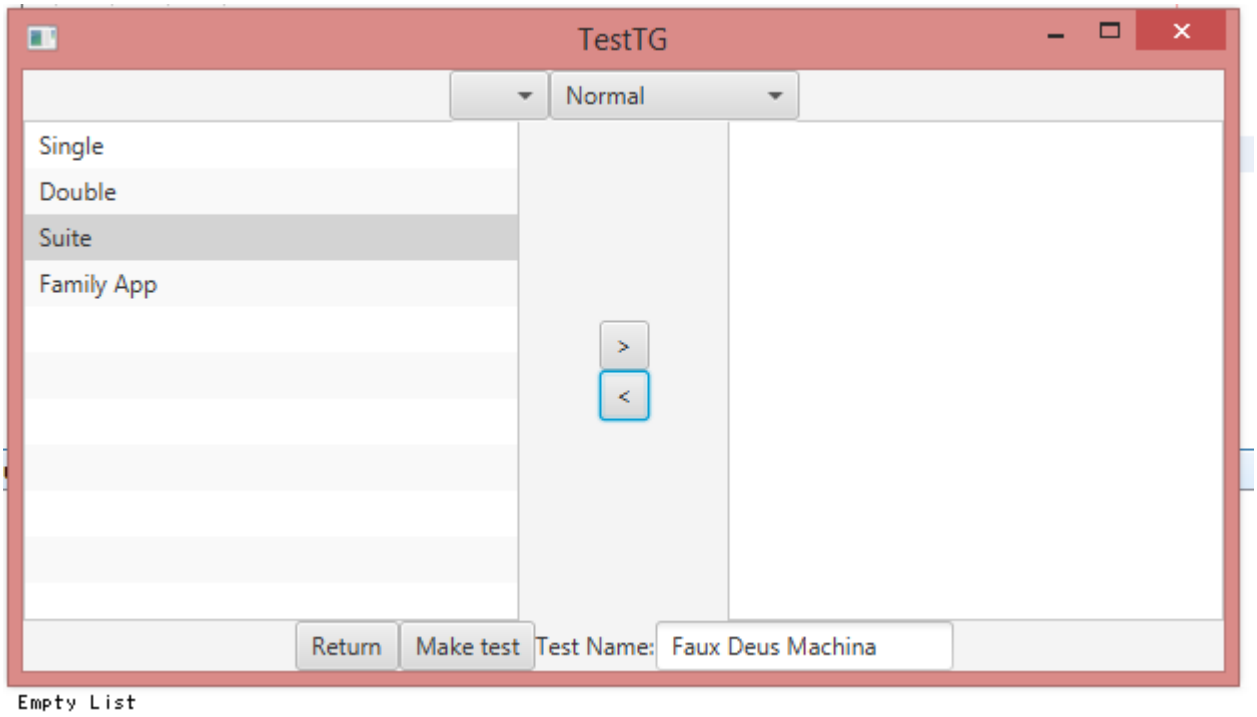
selection we created two Choicebox(es), placed some placeholder names in the left box, and then implemented arrows that added and removed items to/from the right box. Originally, the use of these arrows created an error but after a short if – else statement it worked properly. Afterwards there were small advancements with the program. A Text field where you can add the name of the text, a list where select the difficulty, and some other buttons to set up for more options.

Most of my work was add the return button, the difficulty list, the name test text field , and fixing an error that was in the code. When I received the basis of the GUI from Jonathan, there were two separate frames. The first frame had a make test button that sent it to the next frame. I thought to start off I would add a return button for the second frame as to allow a two-way flow between the frames. Setting up the button really only took two lines of code. The first line created the button and the second line set an action linking the two frames.

Next the list for the difficulty took three lines of codes. The first line creates the list, the second puts the elements on the list, and the last line chooses a default choice for the list.



The text field code uses three lines. The first line creates a label called “Test Name”, the next lines creates the text field for the user to type in, and the last lines adds the both together in the second frame.



Overall there was only one error show up that took a long time to fix. When testing the Combobox(es) sending items from the left to right was fine, but when removing items from the right side a error occurred when the Combobox was empty. In the end, I integrated a counter for the Combobox so when the counter was 0 its output would be "Empty List".

The way this was meant to work is, a user would come to the frame they would choose a subject from the list in the upper left corner. After choosing a subject, the left Combobox would display the available chapters. The user could then choose the chapters the user wanted by using the arrow buttons in the middle to place them in the right Combobox. In the upper right corner, the user could choose the difficulty of the test questions from a list. Lastly, at the bottom of the frame the user could use the return button to go back to the previous frame, the

Return / Make Test / Test Name:

a) Button 1 (Return) – gives the user the option to go back to the first scene. (Finished)
b) Button 2 (Make Test) – Creates a test with the desired variables (in progress)
c) Text Field (Test Name) - The user will enter the name of the test.

(Robert Redus) Methods:

When we initially began the project, we wanted to try and recreate the main methods from scratch. However, shortly after we began working, we decided that it would actually be easier to try and re-use the methods from the previous repository and try and fit them with our GUI.

Most of the original methods were kept as is, the primary changes made were the actual test generator methods, LatexFile and htmlFile.

```
public LatexFile(String dpath) throws IOException{
        XMLretriever = new RetrieveXML(dpath);
        latex_file = new File("/Desktop");
        if(!latex_file.exists()){
            latex_file.createNewFile();
        }
        latex_file_io = new Formatter(latex_file.getAbsolutePath());
        qcount=1;
}
```

The main issue with the test generator previously was that the latex file in the old repository was the actual pdf generation. In this version, we've reworked it so that the latex file is created only on the desktop. This prevents java from not finding the file correctly. The rest of the LatexFile methods, WriteLatexHead, WriteLatexQuestions, WriteLatexFoot, and WriteLatexInstructions remain as they were.

```
public htmlFile(String dpath) throws IOException{
        XMLretriever = new RetrieveXML(dpath);
        html_questions = new List();
        tex_file = new LatexFile("temp.tex");
        tex_file.WriteLatexHead("temp");
}
```

The html constructor was only slightly changed by removing the path from the LatexFile method.

The RetrieveXML functions were mostly kept the same. The "ReturnBy" functions within that class all deal directly with the database, and work correctly because they correspond to the format of the database. These functions might have to be changed depending on the format of the new database, but currently that is not needed.

Our GitHub: https://github.com/jmcore/corerepo