Criterion C: Development

Introduction

This Java program was coded on Netbeans as Netbeans offers Java's swing tools, that enable programmers to create graphical user interfaces for the program with ease. The program is a combination of test maker, test taker, and a test grader for multiple-choices tests. Netbeans organizes classes of the program neatly and allows programmers to easily set up interactions between user and application.

word count: 62

Summary List of All Techniques

- for loops with i++
- nested for loops
- methods that return a value
- methods that take in parameters
- arrays (JComboBox's models and array of solutions)
- ArrayLists of user defined object made from an OOP "template class
- encapsulation of private attributes with accessors and modifier methods
- simple and compound selection (if/else)
- sorting (selection sort) an ArrayList of user defined object based on the object's key attributes
- searching (linear search)
- imported FileWriter for saving to a file
- imported FileReader for opening a file to a table
- error prevention (for example disabling spinners and textfield in the create panel after clicking the render button)
- multiple GUI Windows (setVisible())
- imported JOptionPane to generate option pane for communicating with the user about data removal's confirmation
- imported WindowEvent to allow for the closure of a single window
- use of a flag value for the object's attributes ("Not set", "-99")
- use of JComboBox inside JTable
- parsing and converting values into appropriate data type (Double.parseDouble(), Integer.parseInt(), toString())
- use of setEditable() and setEnabled() GUI Features for disabling textfields and buttons
- JComboBox inside JTable using setCellEditor()
- parsing a file using StringTokenizer
- parameters sent between classes and windows

word count: 0 (bulleted list)

Structure of the Program

What:
The program is divided into five different classes. Three of the classes are GUI classes and the other two are a template class of a user-defined object, and a class with searching and sorting algorithms.

The main class that is run when the user launches the program is a GUI class called LibraryPanel. GUI class is set as the main class as it allows the user to view, manage, and interact with each set of questions using a table and buttons.

LibraryPanel.java

SetOfQs is a template class that was made to allow a set of questions objects to be stored as an ArrayList that would later also be saved as a local file. This means that the attributes of the object can be stored inside each instance of a set of questions, and with encapsulation, the private attributes can only be accessed or changed when certain accessor methods or modifer methods, such as getNumberOfQ(), are triggered.

SetOfQs.java

CreatePanel is a GUI class that allows users to create a new set of questions that can be viewed and managed in the LibraryPanel class. Sets of questions are created by data and input in the CreatePanel class and can be passed onto the LibraryPanel class because a newly created set of questions is initally saved to setofqsfile.txt and solutions for each set of questions are stored in a seperate local file. The local file can then be access by the libraryPanel to construct and reconstruct ArrayList of set of questions object everytime the program is launched or reload.

CreatePanel.java

TestingSessionPanel is a GUI class that allows users to answer the questions in each set of questions. It takes in an ArrayList from It also grades users' answers and calculates results.

TestingSessionPanel.java

SortAndSearchClass is a functional class that contains all of the sorting and searching algorithms that can be used by the LibrayPanel class for sorting displayed sets of questions.

SortAndSearchClass.java

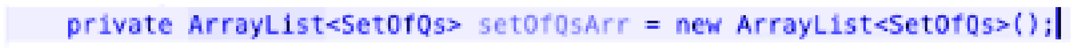
Why:
The ability to have multiple classes allows the use of abstraction, which eases the debuging and maintainance of the program with modularity.

word count: 344 words

Data Structures Used

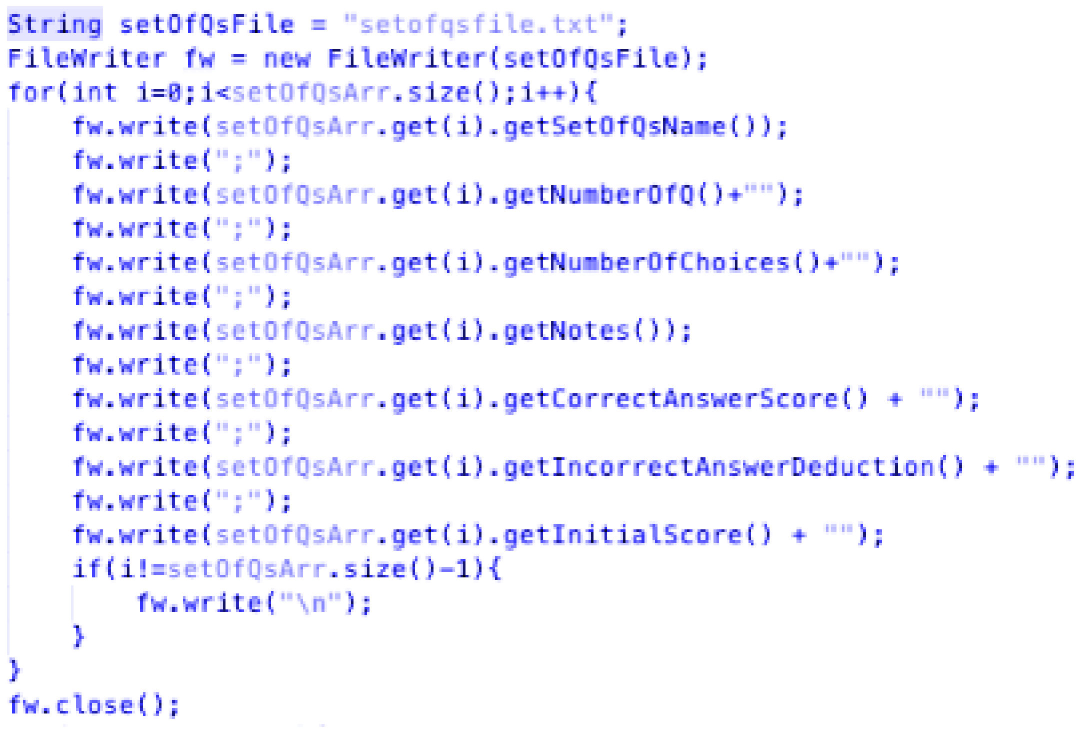
What:

1. **ArrayList**



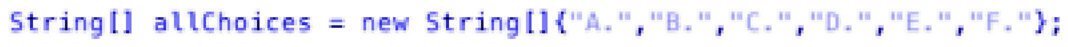
```
private ArrayList<SetOfQs> setOfQsArr = new ArrayList<SetOfQs>();
```

2. **Files**



```
String setOfQsFile = "setofqsfile.txt";
FileWriter fw = new FileWriter(setOfQsFile);
for(int i=0;i<setOfQsArr.size();i++){
    fw.write(setOfQsArr.get(i).getSetOfQsName());
    fw.write(";");
    fw.write(setOfQsArr.get(i).getNumberOfQ()+"");
    fw.write(";");
    fw.write(setOfQsArr.get(i).getNumberOfChoices()+"");
    fw.write(";");
    fw.write(setOfQsArr.get(i).getNotes());
    fw.write(";");
    fw.write(setOfQsArr.get(i).getCorrectAnswerScore() + "");
    fw.write(";");
    fw.write(setOfQsArr.get(i).getIncorrectAnswerDeduction() + "");
    fw.write(";");
    fw.write(setOfQsArr.get(i).getInitialScore() + "");
    if(i!=setOfQsArr.size()-1){
        fw.write("\n");
    }
}
fw.close();
```

3. **Array**



```
String[] allChoices = new String[]{"A.","B.","C.","D.","E.","F."};
```

Why:

ArrayList was used to display existing sets of questions that are in the local file as ArrayList and sets of questions can both change in size.

Example: Set of questions with attributes of number of questions, name, notes, and etc.

Files can be used to save each set of questions' attributes and each set of questions' solutions, all of which can be later accessed when needed. Saving and reading from local file by each class was acheived through the use of the class FileWriter and FileReder

Example: Create a single local file that holds attributes of each set of questions

Array of strings was used to create the comboboxes' models that are used for choice selections.

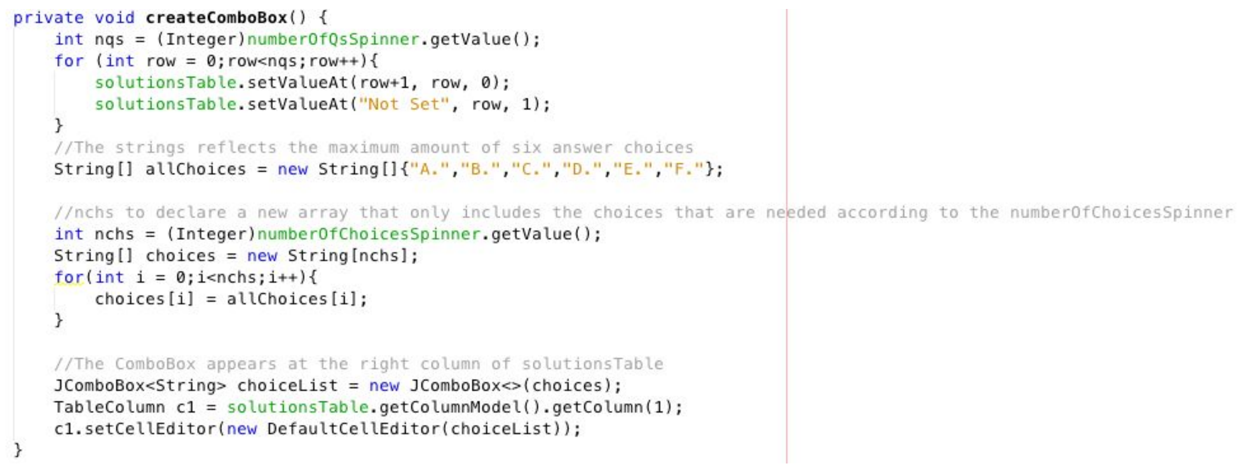
Example: The array allChoices contains all of the six available choices (A., B. C., etc.) that can be assigned as a solution or an answer for each question.

word count: 141 words

Main Unique Algorithms

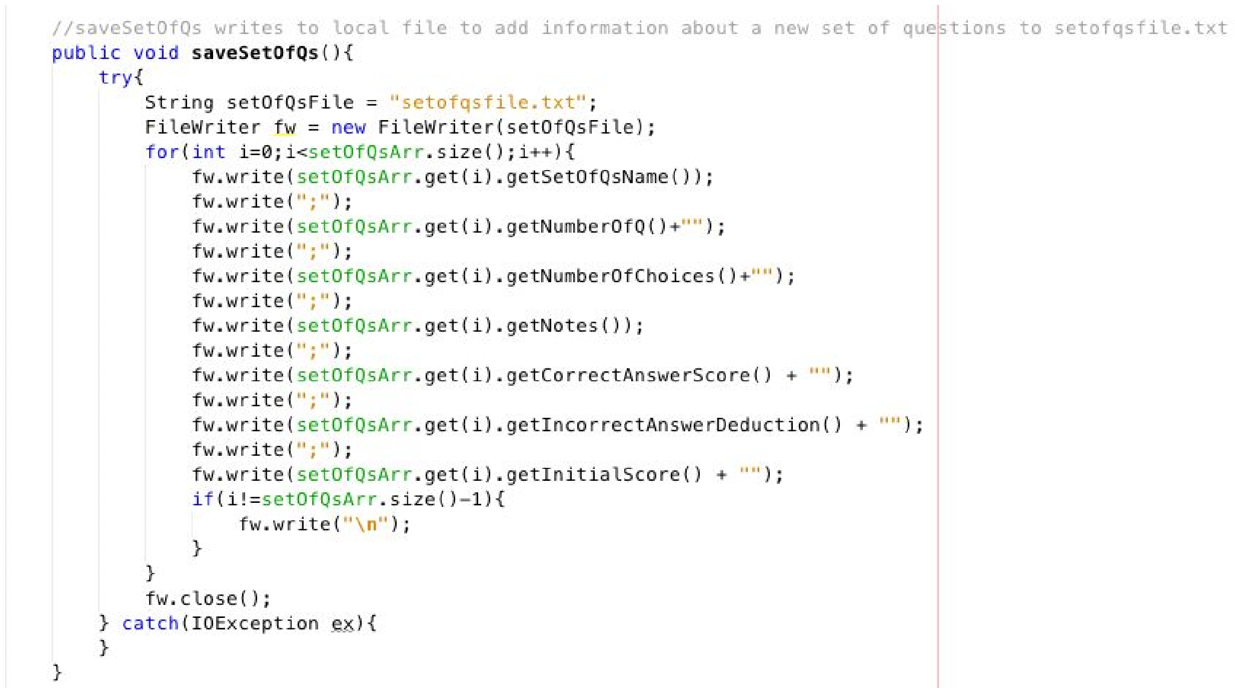
What:

1. **ComboBox Creation**



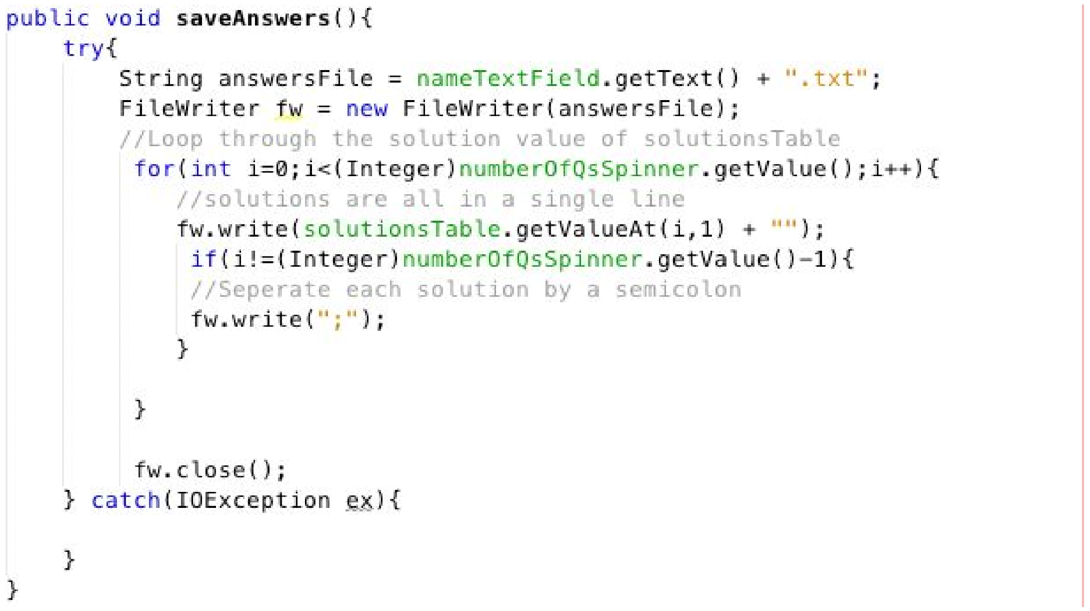
```java
private void createComboBox() {
    int nqs = (Integer)numberOfQsSpinner.getValue();
    for (int row = 0;row<nqs;row++){
        solutionsTable.setValueAt(row+1, row, 0);
        solutionsTable.setValueAt("Not Set", row, 1);
    }
    //The strings reflects the maximum amount of six answer choices
    String[] allChoices = new String[]{"A.","B.","C.","D.","E.","F."};

    //nchs to declare a new array that only includes the choices that are needed according to the numberOfChoicesSpinner
    int nchs = (Integer)numberOfChoicesSpinner.getValue();
    String[] choices = new String[nchs];
    for(int i = 0;i<nchs;i++){
        choices[i] = allChoices[i];
    }

    //The ComboBox appears at the right column of solutionsTable
    JComboBox<String> choiceList = new JComboBox<>(choices);
    TableColumn c1 = solutionsTable.getColumnModel().getColumn(1);
    c1.setCellEditor(new DefaultCellEditor(choiceList));
}
```

2. **File writing #1**



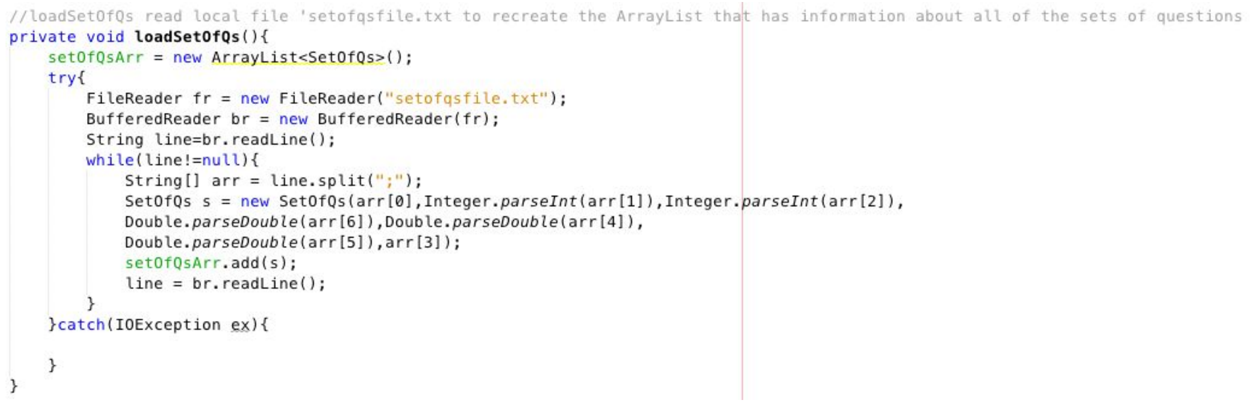
```
//saveSetOfQs writes to local file to add information about a new set of questions to setofqsfile.txt
public void saveSetOfQs(){
    try{
        String setOfQsFile = "setofqsfile.txt";
        FileWriter fw = new FileWriter(setOfQsFile);
        for(int i=0;i<setOfQsArr.size();i++){
            fw.write(setOfQsArr.get(i).getSetOfQsName());
            fw.write(";");
            fw.write(setOfQsArr.get(i).getNumberOfQ()+"");
            fw.write(";");
            fw.write(setOfQsArr.get(i).getNumberOfChoices()+"");
            fw.write(";");
            fw.write(setOfQsArr.get(i).getNotes());
            fw.write(";");
            fw.write(setOfQsArr.get(i).getCorrectAnswerScore() + "");
            fw.write(";");
            fw.write(setOfQsArr.get(i).getIncorrectAnswerDeduction() + "");
            fw.write(";");
            fw.write(setOfQsArr.get(i).getInitialScore() + "");
            if(i!=setOfQsArr.size()-1){
                fw.write("\n");
            }
        }
        fw.close();
    } catch(IOException ex){
    }
}
```

3. **File writing #2**



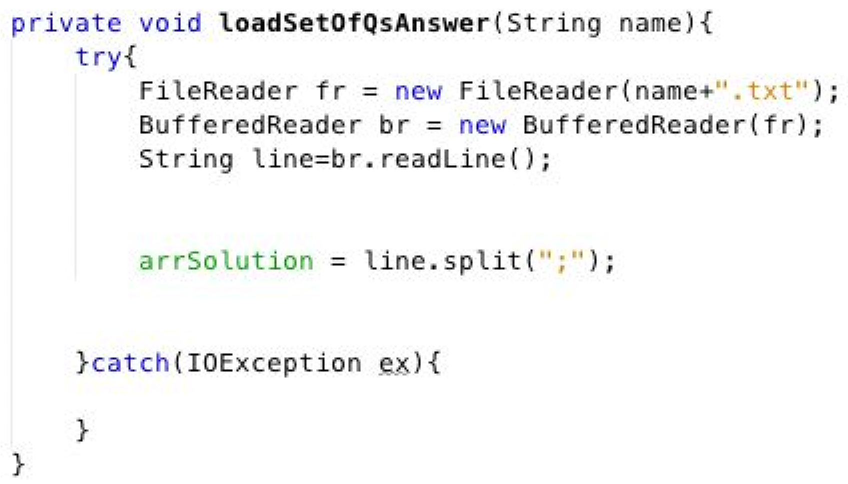
```
public void saveAnswers(){
    try{
        String answersFile = nameTextField.getText() + ".txt";
        FileWriter fw = new FileWriter(answersFile);
        //Loop through the solution value of solutionsTable
         for(int i=0;i<(Integer)numberOfQsSpinner.getValue();i++){
            //solutions are all in a single line
            fw.write(solutionsTable.getValueAt(i,1) + "");
             if(i!=(Integer)numberOfQsSpinner.getValue()-1){
             //Seperate each solution by a semicolon
             fw.write(";");
            }

         }

         fw.close();
    } catch(IOException ex){

    }
}
```

4. **File reading #1**



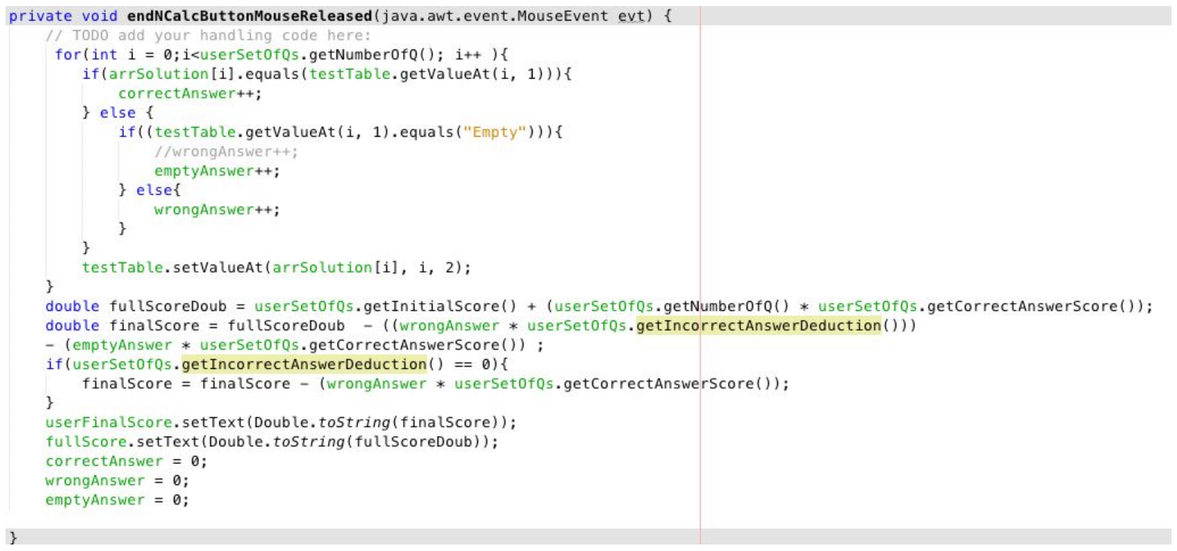
```java
//loadSetOfQs read local file 'setofqsfile.txt to recreate the ArrayList that has information about all of the sets of questions
private void loadSetOfQs(){
    setOfQsArr = new ArrayList<SetOfQs>();
    try{
        FileReader fr = new FileReader("setofqsfile.txt");
        BufferedReader br = new BufferedReader(fr);
        String line=br.readLine();
        while(line!=null){
            String[] arr = line.split(";");
            SetOfQs s = new SetOfQs(arr[0],Integer.parseInt(arr[1]),Integer.parseInt(arr[2]),
            Double.parseDouble(arr[6]),Double.parseDouble(arr[4]),
            Double.parseDouble(arr[5]),arr[3]);
            setOfQsArr.add(s);
            line = br.readLine();
        }
    }catch(IOException ex){

    }
}
```

5. **File reading #2**



```java
private void loadSetOfQsAnswer(String name){
    try{
        FileReader fr = new FileReader(name+".txt");
        BufferedReader br = new BufferedReader(fr);
        String line=br.readLine();


        arrSolution = line.split(";");


    }catch(IOException ex){

    }
}
```

6. **Result Calculation**



```java
private void endNCalcButtonMouseReleased(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
     for(int i = 0;i<userSetOfQs.getNumberOfQ(); i++ ){
        if(arrSolution[i].equals(testTable.getValueAt(i, 1))){
            correctAnswer++;
        } else {
            if((testTable.getValueAt(i, 1).equals("Empty"))){
                //wrongAnswer++;
                emptyAnswer++;
            } else{
                wrongAnswer++;
            }
        }
        testTable.setValueAt(arrSolution[i], i, 2);
    }
    double fullScoreDoub = userSetOfQs.getInitialScore() + (userSetOfQs.getNumberOfQ() * userSetOfQs.getCorrectAnswerScore());
    double finalScore = fullScoreDoub  - ((wrongAnswer * userSetOfQs.getIncorrectAnswerDeduction()))
    - (emptyAnswer * userSetOfQs.getCorrectAnswerScore()) ;
    if(userSetOfQs.getIncorrectAnswerDeduction() == 0){
        finalScore = finalScore - (wrongAnswer * userSetOfQs.getCorrectAnswerScore());
    }
    userFinalScore.setText(Double.toString(finalScore));
    fullScore.setText(Double.toString(fullScoreDoub));
    correctAnswer = 0;
    wrongAnswer = 0;
    emptyAnswer = 0;

}
```

Why:

- ➔ ComboBox Creation - Choices for each question come in the form of an array of strings as JComboBox can only take in an array of strings.
  - ◆ An archetype array of six strings (representing the maximum number of six choices) is created with each element's value assigned as the six available choices (A., B., C., D., etc.).
  - ◆ A **new array** of the amount of strings that is equal to the numberOfChoices attribute for a particular set of questions is created.
  - ◆ A for loop is used to assign the archetype array's elements that are needed to each of the **newly created array** of choices' elements. The amount of elements in the newly created array will again be equal to the number of choices attributed for that particular set of questions.
- ➔ File Writing # 1 (Saving each set of questions) - Each set of questions are basically saved as one single line of parameters that are seperated by a semicolon. The seperations between each parameter (each setOfQs attribute) were acheived by the StringTokenizer method[1]. All of the sets of questions' parameters are saved to a single text file called setofqsfile.txt. The parameters of a particular set of questions are seperated from the parameters of another set of questions by line.
- ➔ File Writing #2 (Saving the sets of questions' solutions) - Sets of questions' solutions are basically saved as **one single line** of parameters that are seperated by a semicolon. The seperations between each parameter (each solution) were acheived by the StringTokenizer method. All of a partcular set of questions' parameters are saved to a single text file that is named after a particular set of questions' setOfQsName attribute.
- ➔ File Reading #1 (Reading the sets of questions from a local file and turning them into ArrayList) - An **array** of setOfQs attributes is created by line.split(). Elements of the array of setOfQs attributes are then parsed into approapriate data type and passed on to form a new element of the setOfQsArr **ArrayList**.
- ➔ File Reading #2 (Reading a particular set of questions' solutions) -Array of solutions is created by line.split().
- ➔ Result Calculation - for loop to compare each element of the array of solutions with each value (each answer) of the table's column.
  - ◆ If the value of both matches, increase the variable correctAnswer by one
  - ◆ If the value of the table's column equals to "Empty", increase the variable emptyAnswer by one
  - ◆ If the value doesn't match any of the above condition, increase the variable wrongAnswer by one

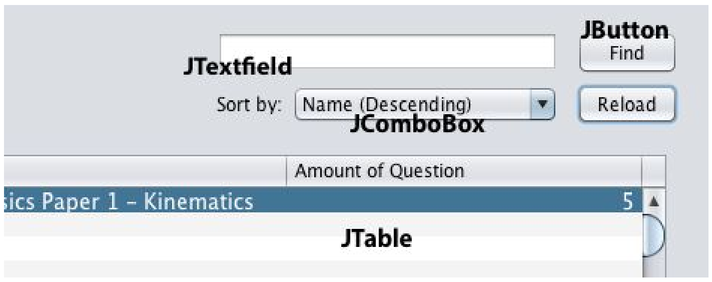
word count: 410

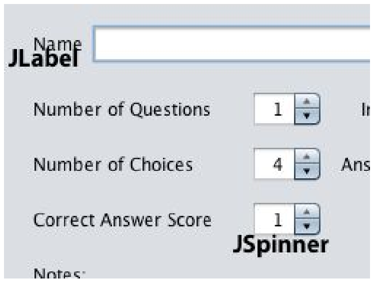
---

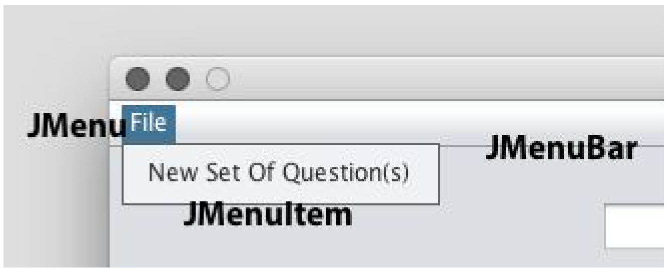
[1] Both file writing and reading code are tweaked from John Rayworth's video. "Full File Writing and Reading Project".
Link:https://www.youtube.com/watch?v=3bcl246jlSg

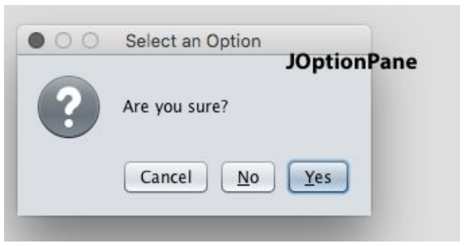
User Interface/GUI Work

What:









Why:

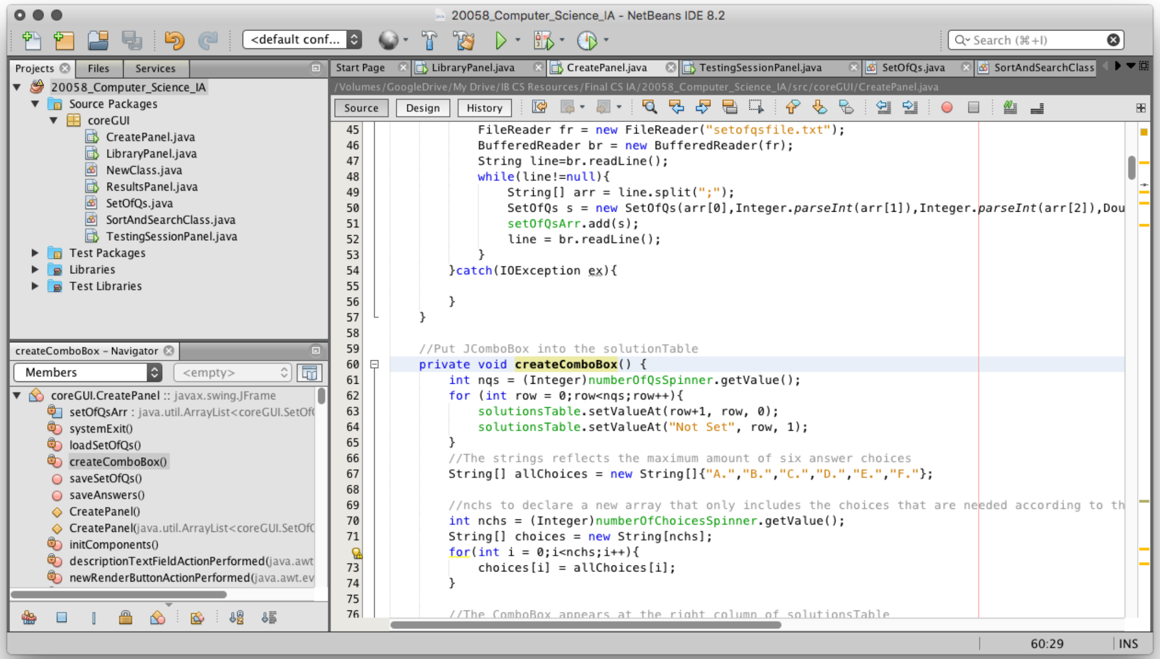
- JTextfield - Can be use to input attributes of string type
- JButton - Allow user to click after entering information
- JComboBox - Can be use to assign a choice either as a solution or an answer
- JTable - Can be use to display sets of questions and questions inside a set of questions
- JMenu, JMenuBar, JMenuItem - Can be use to launch CreatePanel window
- JOptionPane - Can be use to confirm removal of set of questions
- JLabel - Can be use to tell users what each GUI component is for
- JSpinner - Allows user to input set of questions' attributes of double and integer type

word count: 0 (bulleted list)

Software Tools Used

What:

NetBeans, an integrated development environment for Java that is used by Java programmers around the world.



```
            FileReader fr = new FileReader("setofqsfile.txt");
            BufferedReader br = new BufferedReader(fr);
            String line=br.readLine();
            while(line!=null){
                String[] arr = line.split(";");
                SetOfQs s = new SetOfQs(arr[0],Integer.parseInt(arr[1]),Integer.parseInt(arr[2]),Dou
                setOfQsArr.add(s);
                line = br.readLine();
            }
        }catch(IOException ex){

        }
    }

    //Put JComboBox into the solutionTable
    private void createComboBox() {
        int nqs = (Integer)numberOfQsSpinner.getValue();
        for (int row = 0;row<nqs;row++){
            solutionsTable.setValueAt(row+1, row, 0);
            solutionsTable.setValueAt("Not Set", row, 1);
        }
        //The strings reflects the maximum amount of six answer choices
        String[] allChoices = new String[]{"A.","B.","C.","D.","E.","F."};

        //nchs to declare a new array that only includes the choices that are needed according to th
        int nchs = (Integer)numberOfChoicesSpinner.getValue();
        String[] choices = new String[nchs];
        for(int i = 0;i<nchs;i++){
            choices[i] = allChoices[i];
        }

        //The ComboBox appears at the right column of solutionsTable
```

Why:

- Can be coded in Java, which means that programmers have access to Java libraries and programmers can take an object-oriented approach
- GUI components can be implement with an ease
- The program will work across multiple platforms

word count: 16 (ignoring bulleted lists)

full document word count: 973

Crit A + B + C: **1685**