CRITERION C: DEVELOPMENT

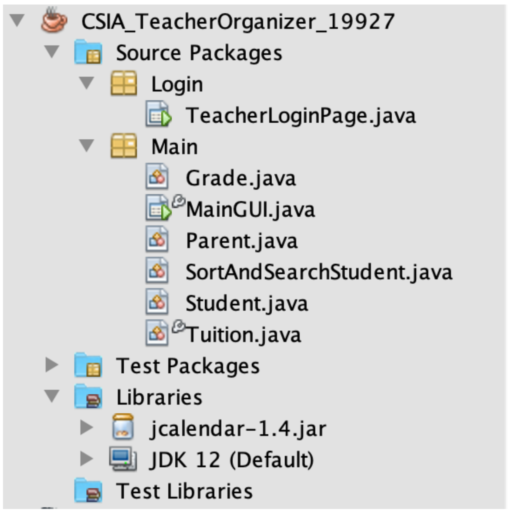
<u>Introduction</u>

This Teacher Organizer Java program was developed via Apache Netbeans. Tools such as Swing Controls and Swing Windows were employed to achieve the final graphical user interface. Remembering that the client required the program to add information on new students and be able to search them.

<u>List of Techniques</u>

- Graphical user interface
- Global and local variables
- Methods
- Parameter passing
- For loop
- While loop
- User-defined objects made from OPP template classes
- Arrays
- Sorting
- Searching
- Array List
- Linked list
- Returning absolute chosen file path
- Encapsulation, using get and set methods
- Aggregation
- Inheritance

<u>Structure of Program</u>



```
CSIA_TeacherOrganizer_19927
  Source Packages
    Login
      TeacherLoginPage.java
    Main
      Grade.java
      MainGUI.java
      Parent.java
      SortAndSearchStudent.java
      Student.java
      Tuition.java
  Test Packages
  Libraries
    jcalendar-1.4.jar
    JDK 12 (Default)
  Test Libraries
```

OPP was employed because it seemed the most natural approach to problem-solving as it allowed the creation of <u>multiple instances</u> of the same kind of object. Furthermore, <u>encapsulating</u> the attributes of specific template classes could be manipulated in its public method.

Noticing the Java classes in Netbeans, the classes with OOP relationships: Grade, Parent, Student, and Tuition. The main relationships between them may vary from Criterion B, however, <u>aggregation</u> was used to create functionality by taking different classes and combining them into a new class: Parent has a Student, Student has a Grade, Student has a Tuition. The other <u>SortAndSearchStudent</u> class was created to organize all sorting and searching programs under one class for organization. Also, two <u>packages</u> were created: Login and Main. Mostly everything is under Main except the TeacherLoginPage GUI which is the first interface for the user before redirecting to the MainGUI. Also, <u>external libraries</u>, such as the JDateChooser, were used in order to meet the client's desired GUI requirements.

Encapsulation and Aggregation Example from Student class

```
public Student(String name, String schoolName, String programName, int gradelevel,Parent
parent){
        this.name = name;
        this.schoolName = schoolName;
        this.programName = programName;
        this.gradelevel = gradelevel;
        this.parent = parent;
    }

    public String getname(){
        return name;
    }
    ...
    public Parent getParent(){
        return parent;
    }
```

Example of Multiple instances of classes

```
if(students.size() < NewStudentTable.getRowCount()){
            for( int row = 0; row < students.size(); row++){
                NewStudentTable.setValueAt(students.get(row).getname(), row, 0);
                ...
                NewStudentTable.setValueAt(students.get(row).getParent().getparentMobile(), row, 5);
            }
        }
```

Variables:
A global variable is used in the main class of the program. The public variables used in the Main class are accessible by all classes. The encapsulation allowed the private variables to only be changed by using the access methods.

```
    private int counter = 0;
```

```
    private String name = "not set yet";
    private String schoolName = "not set yet";
    private String programName = "not set yet";
    private int gradelevel = 0;
    private Parent parent = null;
```

```
    private String parentName = "not set yet";
    private int parentMobile = 0;
    private String parentEmail = "not set yet";
```

```
    private String topicName = "not set yet";
    private int topicTestGrade = 0;
    private String topicNote = "not set yet";
```

Methods:
The main methods used in the program vary from the class diagram presented in Criterion B. Having written only the algorithm for some of these methods, I had to perform a few extra tasks in order to develop the program.

The private methods, AddNewStudentButtonMouseReleased and ADDNewSRefreshButtonMouseReleased, are used to input and output information into the NewStudentTable. This method uses aggregation where the Parent has a Student.

Add New Student Methods

```
private void AddNewStudentButtonMouseReleased(java.awt.event.MouseEvent evt) {
        Parent p = new Parent(ParentNameTF.getText(), Integer.parseInt(ParentMobileTf.getText()),
ParentEmailTF.getText());
        students.add(new Student(NewStudentNameTF.getText(), NewStudentSchoolTF.getText(),
NewStudentProgramComB.getSelectedItem()+"",
Integer.parseInt(NewStudentGradeComB.getSelectedItem()+""), p));

         NewStudentNameTF.setText("");
         NewStudentSchoolTF.setText("");
         NewStudentProgramComB.getSelectedIndex();
         NewStudentGradeComB.getSelectedIndex();
         parents.add(new Parent(ParentNameTF.getText(), Integer.parseInt(ParentMobileTf.getText()),
ParentEmailTF.getText()));
         ParentNameTF.setText("");
         ParentMobileTf.setText("");
         ParentEmailTF.setText("");
   }
}
```

Sorting:

I used selection sorting for all the alphabetical sorting because it is efficient as it only swaps values once every outer pass. This is done when each pass keeps track of which is the smallest and then swaps only once at the end of the full pass. In this example, an outer for loop repeats through the array of classes. When each pass is initised, the program assumes that the initial location is the smallest element. Each element is checked to see if they are smaller than they are assumed to be. IF they are smaller than, the newest smallest element is assigned to be the smallest, estenaillty they are swapped. This is kept track of with the minIndex. This process is repeated until the ArrayList of classes is sorted.

Selection Sort Example

```
public void selectionSortofClasses(ArrayList<Grade> classes) {
        for (int i = 0; i < classes.size() - 1; i++) {
            int minIndex = i;      // Assumed index of smallest remaining value.
            for (int j = i + 1; j < classes.size(); j++) {
                if (classes.get(j).gettopicName().compareTo(classes.get(minIndex).gettopicName()) < 1
){
                minIndex = j;  // Remember index of new minimum
            }
        }
        if (minIndex != i) {
            //Exchange current element with smallest remaining.
            //But note that this only happens once each outer loop iteration, at the end of the inner
loop's looping
            Grade temp = classes.get(i);
            classes.set(i, classes.get(minIndex));
            classes.set(minIndex, temp);
        }
    }
}
```

Searching:

Linear search seemed the most logical. If while doing the search the key is found, then the element where it is located is returned and the program stops. If the key is not found, then -1 is returned. It is a good choice because it doesn’t waste time looping if the key is found. Even though binary search would be the more efficient search by cutting the possible number of elements to loop through in half each time. It is not useful in this case where the array is constantly changing and will not always have even number of elements for balance.

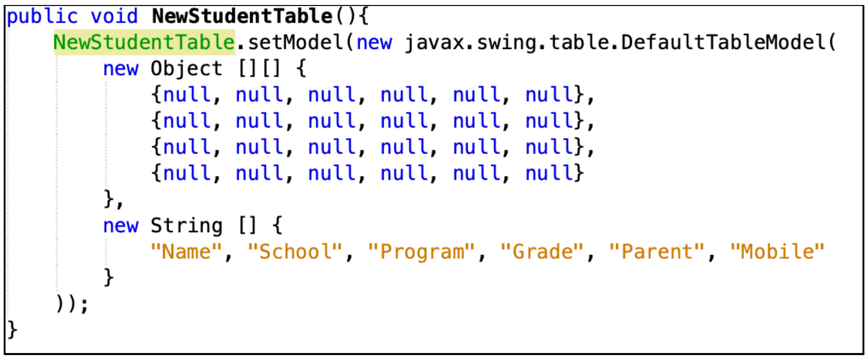
Searching Example

```
public int searchByName(ArrayList<Student> students, String key){
    for(int i = 0; i < students.size(); i++){
        if(students.get(i).getname().equalsIgnoreCase(key)){
            return i;
        }
    }
    return -1;
}
```

Data Structure Used

Two-Dimensional Arrays:

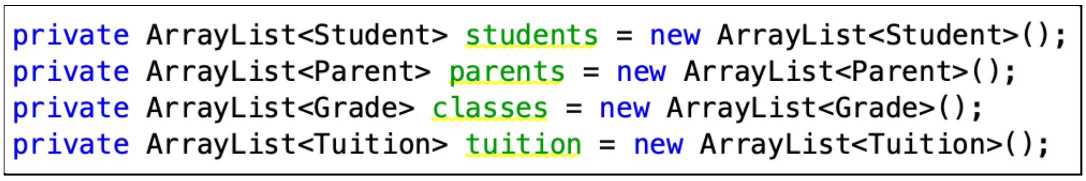
The reason why 2D Arrays were used because it stores an array of arrays. In the program, there are four tables that can be considered as examples. Since there are multiple rows, each row is itself an array of column elements. It is used to store objects; in this case the Student.



```
public void NewStudentTable(){
    NewStudentTable.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null, null, null},
            {null, null, null, null, null, null},
            {null, null, null, null, null, null},
            {null, null, null, null, null, null}
        },
        new String [] {
            "Name", "School", "Program", "Grade", "Parent", "Mobile"
        }
    ));
}
```

Array List:

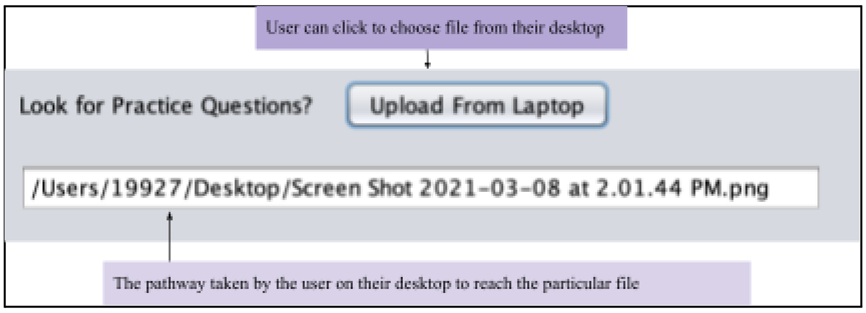
It was sensible to use this because it mimics real-life situations in the Last-In-First-Out(LIFO) or First-In-First-Out(FIFO) manner. It is used to store a dynamic collection of elements. It was chosen because the size of students the client receives is always changing. As it expands by itself when new elements are added, no memory is wasted whereas arrays are helpful when the size is fixed. The disadvantages is that it provide direct access to only the head of the list, which makes searching inefficient.



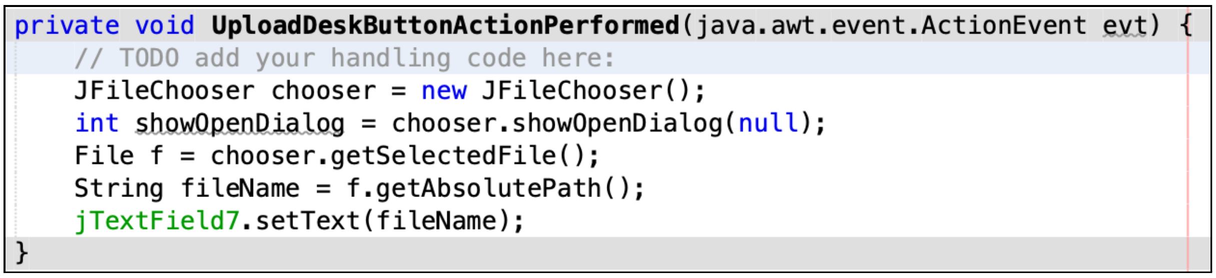
```
private ArrayList<Student> students = new ArrayList<Student>();
private ArrayList<Parent> parents = new ArrayList<Parent>();
private ArrayList<Grade> classes = new ArrayList<Grade>();
private ArrayList<Tuition> tuition = new ArrayList<Tuition>();
```

Uploading File Algorithm

In the Classes panel, the user can choose a file from their desktop and save the pathway that is taken to reach the document for future reference.

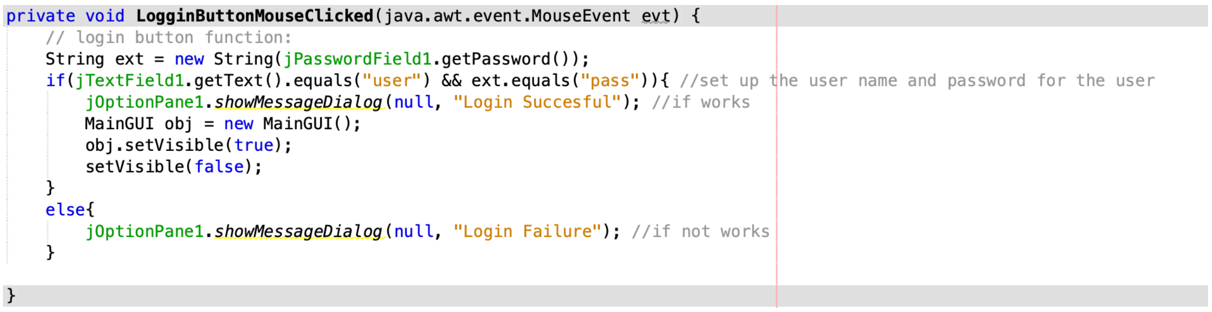

User can click to choose file from their desktop
Look for Practice Questions?
Upload From Laptop
/Users/19927/Desktop/Screen Shot 2021-03-08 at 2.01.44 PM.png
The pathway taken by the user on their desktop to reach the particular file

I imported an external library for the JFileChooser and then created a new object to use as instance, later on, to open the Dialogue Swing Window with all the files from the respective desktop. Then the getAbsolutePath() method pathname of the given file object.



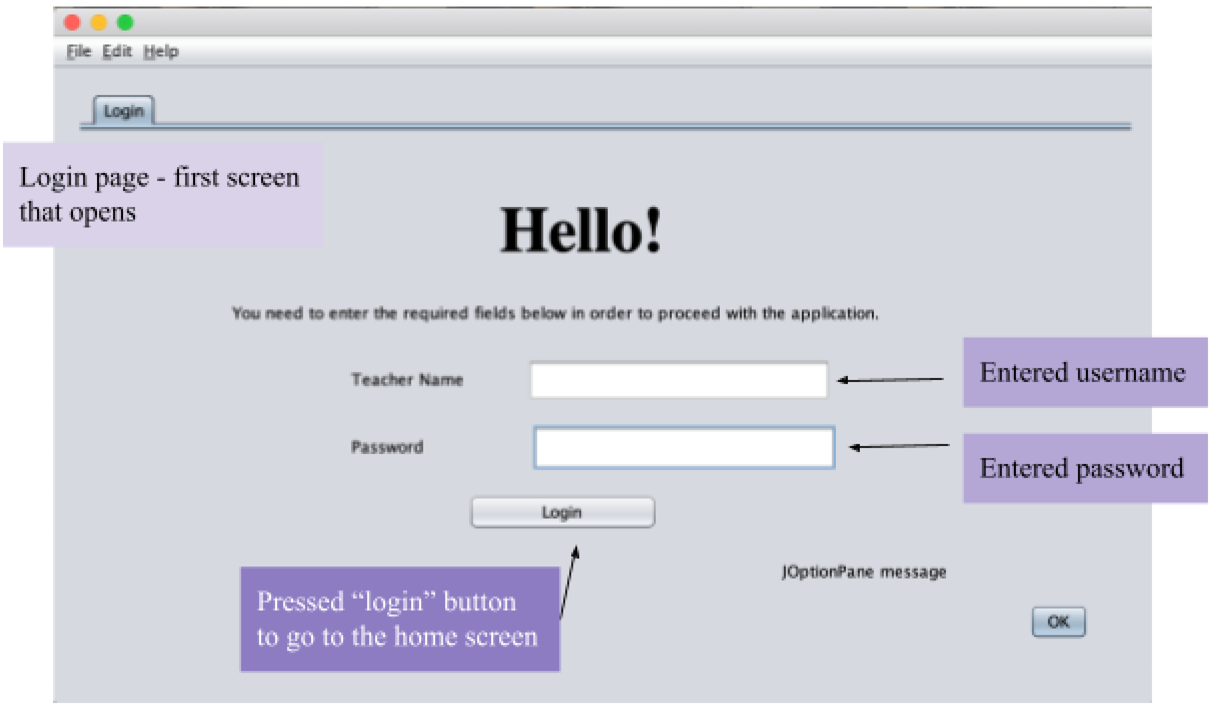
```java
private void UploadDeskButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JFileChooser chooser = new JFileChooser();
    int showOpenDialog = chooser.showOpenDialog(null);
    File f = chooser.getSelectedFile();
    String fileName = f.getAbsolutePath();
    jTextField7.setText(fileName);
}
```

I set up a login page as requested by the client by creating a Java Package. This allows the user to create their username and password for their login. Furthermore, if there is an error a dialogue would pop up with a warning message to enter the login information again.



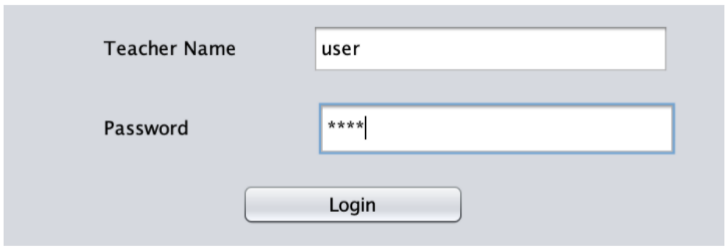
```java
private void LogginButtonMouseClicked(java.awt.event.MouseEvent evt) {
    // login button function:
    String ext = new String(jPasswordField1.getPassword());
    if(jTextField1.getText().equals("user") && ext.equals("pass")){ //set up the user name and password for the user
        jOptionPane1.showMessageDialog(null, "Login Succesful"); //if works
        MainGUI obj = new MainGUI();
        obj.setVisible(true);
        setVisible(false);
    }
    else{
        jOptionPane1.showMessageDialog(null, "Login Failure"); //if not works
    }

}
```



File Edit Help
Login
Login page - first screen that opens
Hello!
You need to enter the required fields below in order to proceed with the application.
Teacher Name
Entered username
Password
Entered password
Login
Pressed “login” button to go to the home screen
JOptionPane message
OK

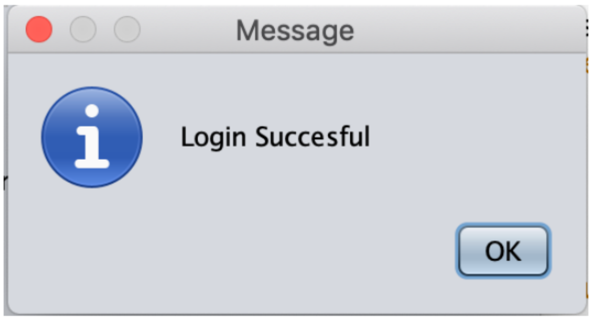
User Interface/GUI



Teacher Name: user

Password: ****

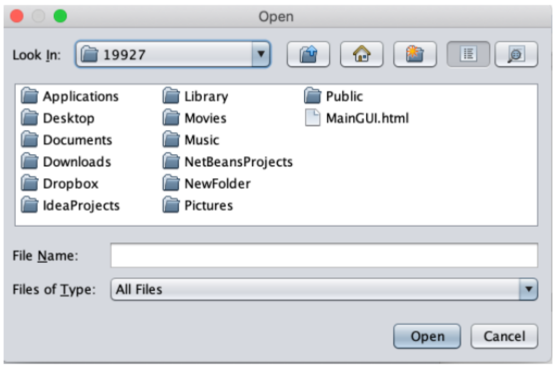
Image 1: Employment of the Password Field from Swing Controls allowed the hiding of the password with the asterisks and also allowed the user to enter the password required.



Message

Login Succesful

Image 2: When the login button is clicked, this Dialogue from Swing Windows shows up in order to verify if the login was successful or not.



Open

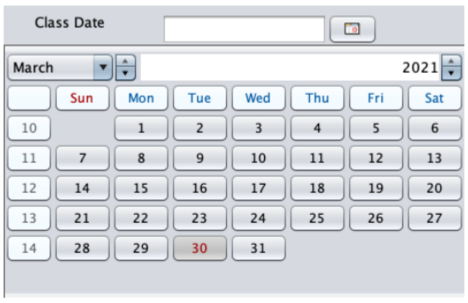
Look In: 19927

Applications, Desktop, Documents, Downloads, Dropbox, IdeaProjects, Library, Movies, Music, NetBeansProjects, NewFolder, Pictures, Public, MainGUI.html

File Name:

Files of Type: All Files

Image 3: The file chooser and the option pane allowed the user to access and choose from all their desktop files.



Class Date

March 2021

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
10		1	2	3	4	5	6
11	7	8	9	10	11	12	13
12	14	15	16	17	18	19	20
13	21	22	23	24	25	26	27
14	28	29	30	31			

Image 4: Employment of the external library of JCalender and future sophisticated tool of JDateChooser which allowed the calendar to be much more compact and according to the user's preference.

Software tools

- Apache Netbeans 11.1- Netbeans, as a development environment, was useful since it had many GUI libraries and free import of external libraries.
- Draw.io - flowchart, class diagram
- Google docs
- Lucidcharts - class diagrams
- iMovie - video
- Otter.ai - transcribe

Word Count: 893