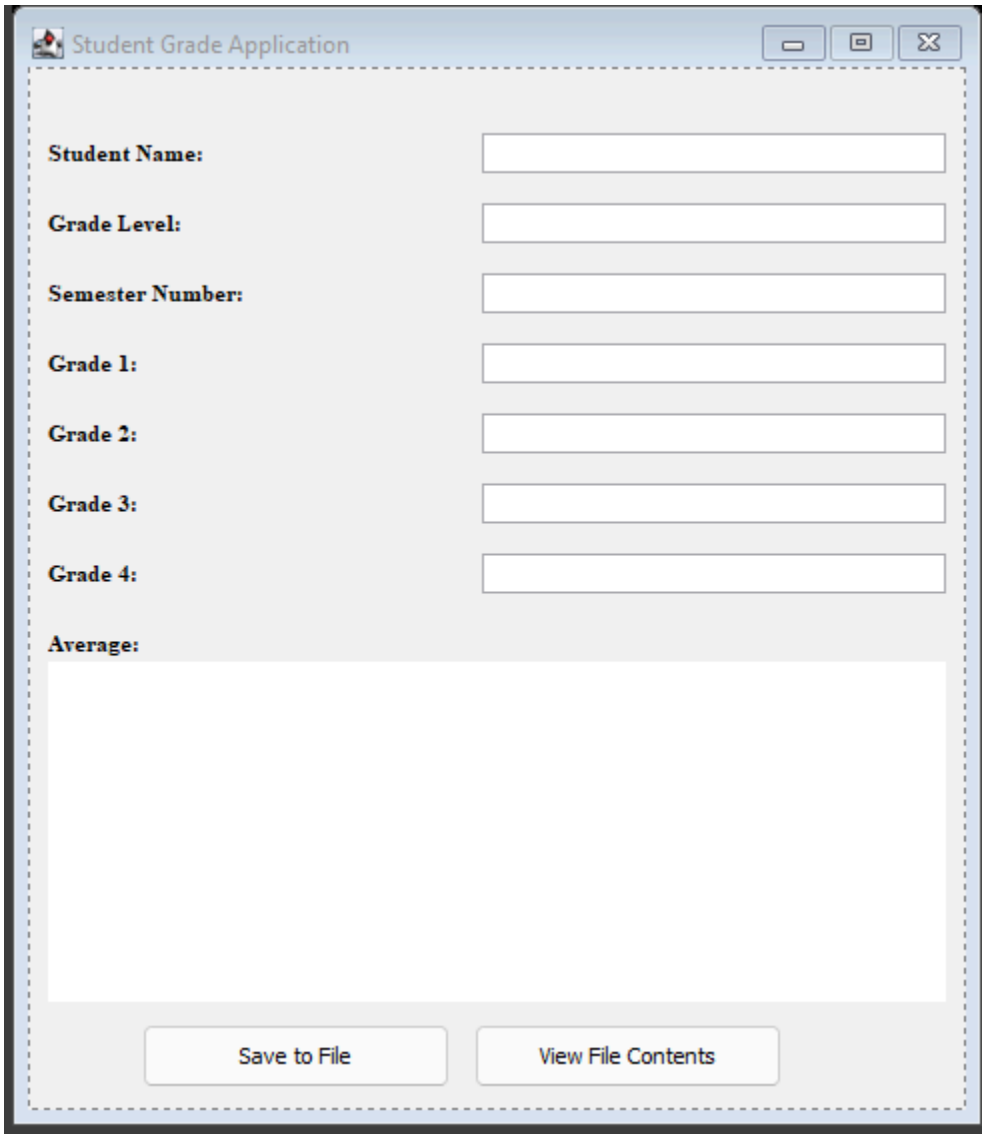
Credit Name: CSE 2130- File Structure and Exception Handling

Assignment: studentSemesterAverageGUI

How has your program changed from planning to coding to now? Please Explain



As usual with a GUI, I started by designing the interface: I declared and labelled text fields and JLabels to outline the inputs that need to be entered. I added a JLabel next to the 'Average: ' to be the location where the student's average grade is recorded.

Then i used a JTextArea as the region where the file contents can be displayed. It was important to use a text area rather than any other component because we need separate people's info to be on new lines.

```java
public class StuInfo {
    String Name,gr1,gr2,gr3,gr4,grl,semNo;

    DecimalFormat df = new DecimalFormat("0.00");

    public StuInfo(String name, String gl,String sem, String g1,String g2,String g3,String g4)
    {
        Name = name;
        gr1 = g1;
        gr2 = g2;
        gr3 = g3;
        gr4 = g4;
        grl = gl;
        semNo = sem;

    }
```

I then created a new class which holds and processes the Student's information.

It takes in all the relevant information from the main GUI as parameters and will then use it for the following actions.

```java
public String getAvg()
{
    double avg = 0.25*(Double.parseDouble(gr1)+Double.parseDouble(gr2)+Double.parseDouble(gr3)+Double.parseDouble(gr4));

    return df.format(avg) + "%";
}
```

We use it to calculate the average by adding all 4 entries and dividing by 4 (or multiply by 0.25). We format it to 2 decimal places as concatenate it with a '%' symbol. Using the decimal format converts the average value from a double to a string as we need it to be.

```java
public String getInfo()
{

    return "Name:" + Name + ", Grade Level: " + grl + " , Semester: " + semNo +
            " , Grades: " + gr1 +", " + gr2 + ", " + gr3 + ", " + gr4 + ", Average: " + getAvg();
}
```

Now we have the getInfo() method which concatenates all the information into a single line.

```java
File file = new File("C:\\Users\\Aryan Kapoor\\git\\cs30p32025-ak0122\\Chapter11\\src\\Mastery\\StudentInfo");
```

Then we link a file where we will write and read information to and from.

```java
saveBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {


        student = new StuInfo(name.getText(), grade.getText(), semNo.getText(), grade1.getText(), grade2.getText(), grade3.getText(), grade4.getText());

        avgScore.setText(student.getAvg());

        String info = student.getInfo();
```

Then we set up the actions when we click the save button. We create a StuInfo object with all the information. We print the average score on the JLabel by using the getAvg

method from the StuInfo class. Then we also create an info variable with our student's information formatted in the way we need.

```
try {

        out = new FileWriter(file,true);
        writeFile = new BufferedWriter(out);

        writeFile.write(info);
        writeFile.newLine();

        writeFile.close();
        out.close();

         JOptionPane.showMessageDialog(frmStudentGradeApplication,
                "Successfully written to file!",
                "Success",
                JOptionPane.INFORMATION_MESSAGE);
```

Then we set up a bufferedWriter so we can efficiently write strings to the file. We then write out our formatted information to the file and then create a new line for the next student.

Afterwards, we close our file writing objects.

Lastly for this button, we created a popup window by using a JOptionPane which links to our main frame and displays the information "Successfully written to file!"

```
catch(FileNotFoundException e2)
{
    System.out.println("File was not found.");
}

catch(IOException e2)
{
    System.out.println("Error reading the file.");
}

name.setText(null);
grade.setText(null);
semNo.setText(null);
grade1.setText(null);
grade2.setText(null);
grade3.setText(null);
grade4.setText(null);
avgScore.setText(null);
```

We ensure all exceptions are handled correctly and then we set our text fields to blank values for the next person

Now we move on to our view file contents button:

```java
fileContentBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        try {


            in = new FileReader(file);
            readFile = new BufferedReader(in);


            String line;
            String output = "";


            while(((line = readFile.readLine()) != null))
            {
                output += (line+"\n");
            }
            avgText.setText(output);
```

We set up a bufferedReader for efficient reading of the file.
We declare a variable for our current line while the bufferedReader reads and for the output we will show the user. We have a while loop which runs while our file line is not null. In the loop, we add each line read to the output variable with a newline between each student entry. After the loop is done we set the text area with the information to the output variable's contents.

```java
    } catch (FileNotFoundException e1)
    {
        System.out.println("File Not Found.");
        e1.printStackTrace();
    }catch (IOException e1)
    {
        System.out.println("Error reading the file.");
    }
}
```

We lastly handle all exceptions.