Introduction

- The product is a net beans guidedUserInterface. It accepts student data. Arrange students in different orders and display them on charts. Users can also used the program so search pre existing students to send them premade auto generated texts.

2. **Summary List of All Techniques**

- parameter passing

- for loop

- while loop

- do while loop

- nested loops

- method returning a value

- arrays, 2d arrays

- arrayLists

- user defined objects made from an OOP "template" class

- encapsulation of private methods that work on public attribute of a "template" class

- making an array of objects

- simple and compound selection (if/else)

- bubble sort

- searching (binary search)

- boolean recognition

- Option pane generation for communicating with the user

- GUI tabs

- GUI popup menus

- Use of a flag value (such as -999, or "not set yet")

- overloaded constructors

- parsing

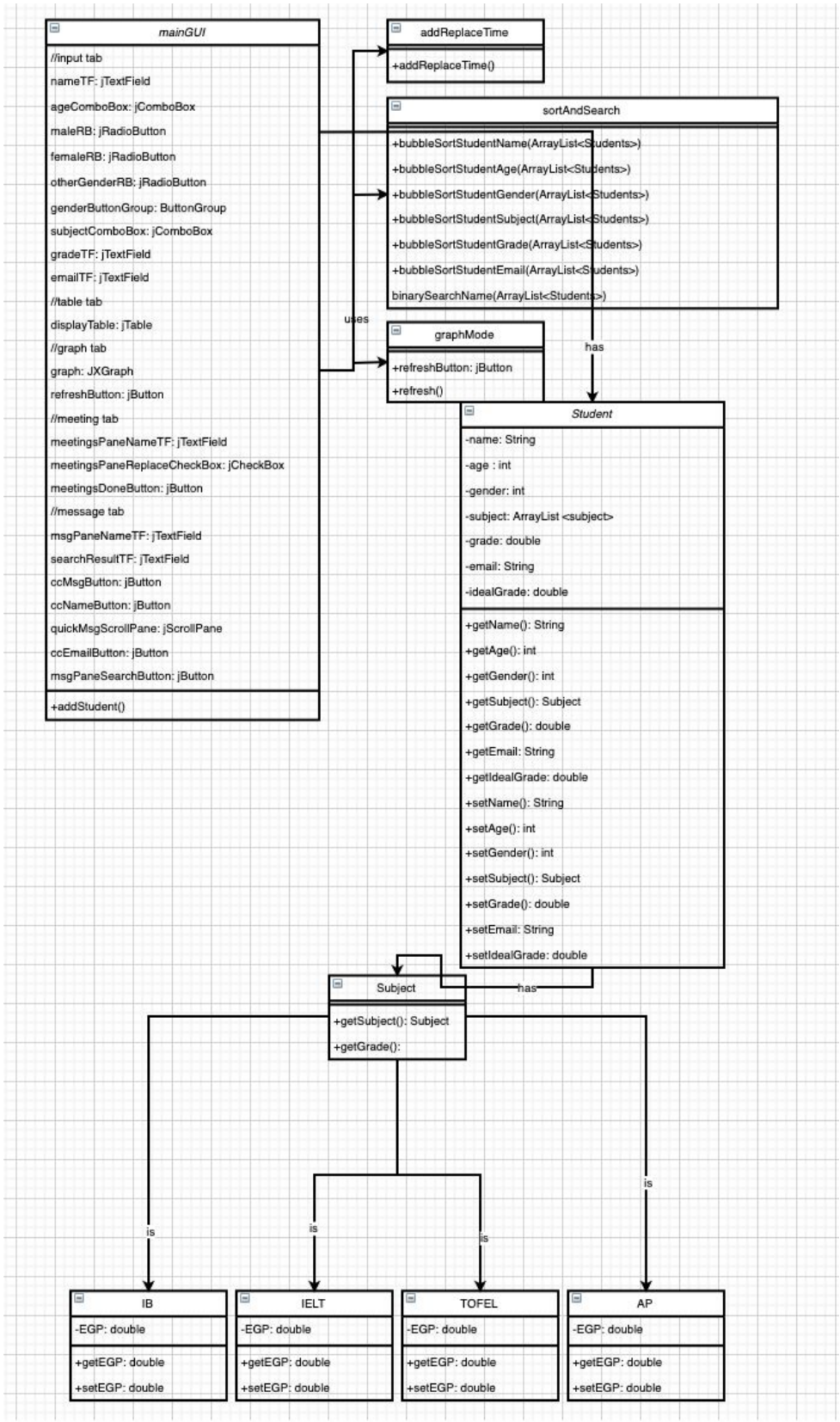
- inheritance between a superclass and a subclass

- use of specialized imported library

3. **Structure of the Program** (including **OOP** Design)

What:

- The “mainGUI” is the main class of the programm, it connects with all the classes and is where the appearance of the program interface is managed.
- “Student” class is a **OOP** class and it is reference from all over the programm to make calculations and display
- “graphStudent” class is an alternative **OOP** class it functions similarly to “Student” class but its abstracted specifically for the “graph” class
- “sortAndSearch” stores and points to all the sorting and searching methods in the programm, it is called from the “mainGUI” class when buttons are pressed
- “Graph” class controls the declaration of a jfree-barChart, a chart with inputs from “graphStudent” class


mainGUI
//input tab
nameTF: jTextField
ageComboBox: jComboBox
maleRB: jRadioButton
femaleRB: jRadioButton
otherGenderRB: jRadioButton
genderButtonGroup: ButtonGroup
subjectComboBox: jComboBox
gradeTF: jTextField
emailTF: jTextField
//table tab
displayTable: jTable
//graph tab
graph: JXGraph
refreshButton: jButton
//meeting tab
meetingsPaneNameTF: jTextField
meetingsPaneReplaceCheckBox: jCheckBox
meetingsDoneButton: jButton
//message tab
msgPaneNameTF: jTextField
searchResultTF: jTextField
ccMsgButton: jButton
ccNameButton: jButton
quickMsgScrollPane: jScrollPane
ccEmailButton: jButton
msgPaneSearchButton: jButton
+addStudent()
addReplaceTime
+addReplaceTime()
sortAndSearch
+bubbleSortStudentName(ArrayList<Students>)
+bubbleSortStudentAge(ArrayList<Students>)
+bubbleSortStudentGender(ArrayList<Students>)
+bubbleSortStudentSubject(ArrayList<Students>)
+bubbleSortStudentGrade(ArrayList<Students>)
+bubbleSortStudentEmail(ArrayList<Students>)
binarySearchName(ArrayList<Students>)
uses
graphMode
+refreshButton: jButton
+refresh()
has
Student
-name: String
-age : int
-gender: int
-subject: ArrayList <subject>
-grade: double
-email: String
-idealGrade: double
+getName(): String
+getAge(): int
+getGender(): int
+getSubject(): Subject
+getGrade(): double
+getEmail: String
+getIdealGrade: double
+setName(): String
+setAge(): int
+setGender(): int
+setSubject(): Subject
+setGrade(): double
+setEmail: String
+setIdealGrade: double
has
Subject
+getSubject(): Subject
+getGrade():
is
is
is
is
IB
-EGP: double
+getEGP: double
+setEGP: double
IELT
-EGP: double
+getEGP: double
+setEGP: double
TOFEL
-EGP: double
+getEGP: double
+setEGP: double
AP
-EGP: double
+getEGP: double
+setEGP: double

Why:

- Splitting my program into different classes is necessary because our problems are complex; therefore they are best solved one piece at a time. Doing this, I won't be overwhelmed or distracted by the large amount of things to keep track of in a programm. This technique also helps me modify each class separately to help add new ideas as I go.**OOP** is a great way of doing **abstraction** because it features divided **classes** that are collectively sent to a single class a, **object oriented** nature of the **OOP** is **abstraction** by default.
- The reason why i had "Student", "graphStudent" as **classes** and separate **classes** are that first, i will be able to make multiple **instances** of the same kind of **object**, **encapsulation** the **attributes** of that "template" class can only be manipulated the way that its **public methods** allow and that new **attributes** and **attributes** can be added and r**efactored** with ease.

4. **Data Structures Used**

Use of User Defined Objects

- Array
    - Student - the list of students, include name, age,gender, subject,grade, email and expected grade
        - global
    - graphStudent - same list of students, only has name, grade and expected grade, used exclusively on "graph" class
        - global
- Array List
    - times - an array list that records all scheduled times in the programm
        - Declared after input button pressed

Imported Libraries

- CategoryDataset
    - dataset - and list of inputs ordered for the displayed graph
        - global

Why:

- Array

- Arrays are used because of the fact that they are basically containers of data and instead of writing and using the actual value, it is a pointer to where the data is stored, this makes using, changing objects in the array more convenient. Student data stored this way so that every student's attribute is “stored” in a package, which reduces the chance of data contamination, it also gives the user no option but to enter all attributes of a student before moving on to record the next student.

- Arraylist
  - Arraylist is used in this program because of its extensibility, the size of the container and where each data is placed is very flexible. This is perfect for the “scheduled time” attribute/feature in the program because not even the user knows how many scheduled times are going to be made, using arraylist allows the user to add new times as they go.

5. **Main Unique Algorithms**

the key section of the login system, if either username isn’t “Jacob” or password isn’t “password” the mainGUI window would not be visible.

```
private void loginButtonMouseReleased(java.awt.event.MouseEvent evt) {

    if(userNameTF.getText().equals("Jacob") && passowrdPasswordField.getText().equals("password")){
        NewMainGUI show = new NewMainGUI();
        show.setVisible(true);
        WindowEvent winClosing = new WindowEvent(this, WindowEvent.WINDOW_CLOSING);
        winClosing.getWindow().hide();
    }
}
```

After logging in, the user accesses the input tab. Data like name, grade are collected from the text fields to templates then the templates are assigned with designated attributes in student . The field is cleared after. “Time” is different, it is added to a globally declared arraylist. Every new time is added at position 0, this is because the program later would display the most recent time added to the user.

```
private void inputButtonMouseReleased(java.awt.event.MouseEvent evt) {
    String name = nameTF.getText();
    int age = Integer.parseInt(ageTF.getText());

    String[] gender = {"", "", ""};
    if (maleRadioButton.isSelected()) {
        gender[0] = "male";
    }
    if (femaleRadioButton.isSelected()) {
        gender[1] = "female";
    }
    if (otherGenderRadioButton.isSelected()) {
        gender[2] = "other";
    }
    String subject = subjectComboBox.getSelectedItem() + "";
    double grade = Double.parseDouble(gradeTF.getText());
    String email = emailTF.getText();
    double expectedGrade = Double.parseDouble(expectedGradeTF.getText());

    students[counter] = new Student(name, age, gender, subject, grade, email, expectedGrade);


    String timeString = timeTF.getText();

    times.add(0, timeString);

    counter++;

    nameTF.setText("");
    ageTF.setText("");
    maleRadioButton.setSelected(false);
    femaleRadioButton.setSelected(false);
    otherGenderRadioButton.setSelected(false);
    subjectComboBox.setSelectedIndex(0);
    gradeTF.setText("");
    emailTF.setText("");
    expectedGradeTF.setText("");
    timeTF.setText("");

}
```

After the student class is filled with attributes, it is transferred to graphStudent. After that, the attributes go into the graph data set. Data set is eventually plugged into the graph as data. Here we use a for-loop so that only the number of student numbers bars are created. The y value of the graph is pointed to gGrade and gExpectedGrade in the graphStudent class.Name for graph use the name of the student referenced.

```
public CategoryDataset gCreateDataset() {
    String grade = "grade";
    String expectedGrade = "expected grade";

    for (int i = 0; i < graphStudents.length; i++) {

        String nameForGraph = graphStudents[i].getGName();
        dataset.addValue(graphStudents[i].getGGrade(), grade, nameForGraph);

        dataset.addValue(graphStudents[i].getGExpectedGrade(), expectedGrade, nameForGraph);

    }
    return dataset;

}
```

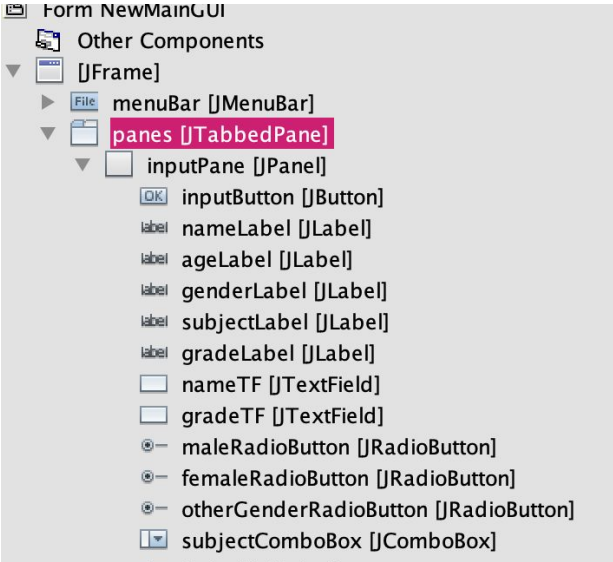
The chart takes in data to configure the properties of the graph created, this includes the title of window, title of graph, data set created earlier.

```
public class graph extends ApplicationFrame {

    public graph(String applicationTitle, String chartTitle, CategoryDataset dataset) {

        super(applicationTitle);

        JFreeChart barChart = ChartFactory.createBarChart(chartTitle,"Category","Score",dataset,PlotOrientation.VERTICAL,
                true, true, false);

        ChartPanel chartPanel = new ChartPanel(barChart);
        chartPanel.setPreferredSize(new java.awt.Dimension(560, 367));
        setContentPane(chartPanel);
    }
}
```

Why: For designated time, I chose arraylist over array because designated time is uncertain in the number of them. This means that the user needs to be able to add new “time”s as they go about using the programm. I chose to alter the format of the way the dataset is inputted into the graph class, because this allows me to declare a global dataset, making reference of attributes from graphStudent possible.

6. **User Interface/GUI Work**

This is a example of how the GUI structures work

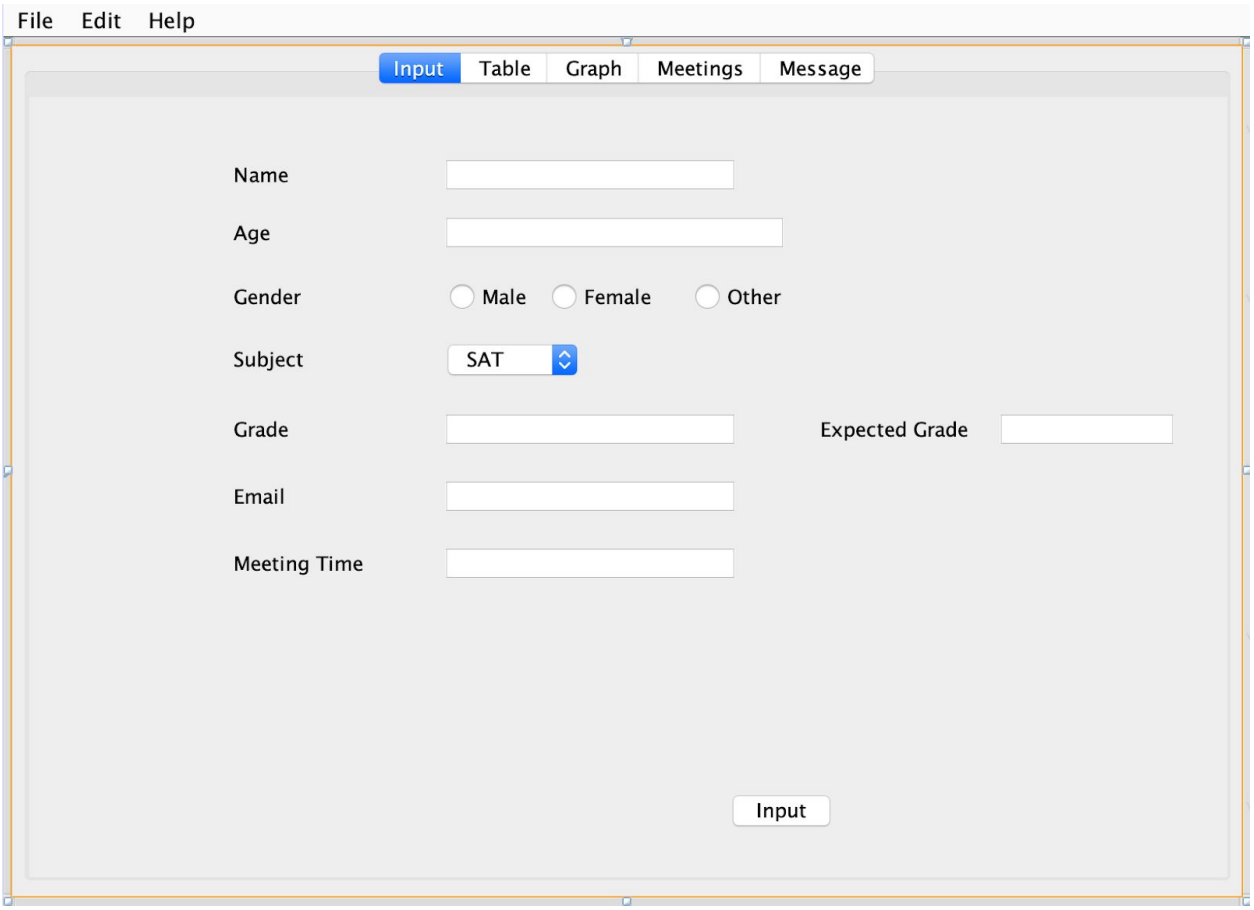


What:

- I used jpanels to divide the user interface into different section with distinct functions
- jbuttons to execute specific commands(event of mouse release linked with methods executing)
- jlabel to explain to the user the functions of the program
- jTextField used to show/ enter data
- jRadioButton to input data where multiple answers/choices is accepted
- jComboBox to input data where single answer is accepted
- jScrollPane so that the user and scroll to see more of the textarea or table
- jTable to display data from user input

Why:

- GUI interface is good in terms of functionally on both developer and user side because of its detailed, thorough yet minimalistic(in terms of graphics and appearance) making it easy for computers of almost any end to load the interface smoothly without the sacrifice of lack of direction.



7. **Software Tools Used**

What:

- NetBeans is an integrated development environment for Java. NetBeans runs on Windows, macOS etc.it's one of the most popular Integrated Development Environment (IDE) used by programming professionals all over the world.

Why:

- Netbeans is great because of its low requirement for the system and a well developed environment because the program has been around for a long time, many users have made external library that accelerate the rate programmers design their own programs

- Netbeans is especially great for this project as I'm a beginner in GUI and programming. Netbeans is simplistic, easy to understand and has extra details in terms of description of every function in the program.