Criteria C

Introduction

I have utilized Netbeans to work on this project. Primarily, I used Java's Graphical User Interface (GUI) in order to create what my client asked for. The basic functions of this program is similar as an actual paper note. For example, my client is a tutor; therefore, this program allows him to keep track on the fundamental informations of his students such as fullname, nickname, grade, start date, etc. Difference is that it has sorting and searching function. Client can find specific student's information by just type in full name or nickname.

Summary List of All Techniques

Major Technique (client visible technique)

- Searching
- Sorting
- Java GUI

Minor Technique (client invisible technique)

- for loop
- if statement
- Get and Set method
- Multiple class (superclass and subclass)
- Error handling
- ArrayList
- Array of the object
- Conversion of String ↔ Int
- if else
- Pop up
- Use of flag value
- Selection sort
- Linear search
- Encapsulation

Algorithm Overview

1: Get all student's information from the text field located in add new student panel to the other jTables.

```
String gender = "Female";
if (radioMale.isSelected())
{
gender = "Male";
}

// select the selected grade from combo box
String grade = "";
grade = gradeBox.getSelectedItem() + "";

// display staart date in form of : month / day / year
String startday = (String) dayBox.getSelectedItem();
String startmonth = (String) monthBox.getSelectedItem();
String startyear = (String) yearBox.getSelectedItem();

String startDate = "" + startmonth + "/" + startday + "/" + startyear;

// Retrieve input → add to the table
students.add(new StudentClass(fullNameTF.getText(),
                nicknameTF.getText(),
                nationalityTF.getText(),
                gender, grade, startDate,
                subjectTF.getText(),
                recommendTF.getText()));

counter++;

// indicate the row
for (int i = 0; i < counter; i++) {
mainTable1.setValueAt(students.get(i).getfullName(), i, 0);
mainTable1.setValueAt(students.get(i).getnickName(), i, 1);
mainTable1.setValueAt(students.get(i).getnationality(), i, 2);
mainTable1.setValueAt(students.get(i).getgender(), i, 3);
mainTable1.setValueAt(students.get(i).getgrade(), i, 4);
mainTable1.setValueAt(students.get(i).getsubject(), i, 5);
mainTable1.setValueAt(students.get(i).getstartDate(), i, 6);
mainTable1.setValueAt(students.get(i).getwhoRecommend(), i, 7);
}        // move to next row
```

```
DefaultTableModel model = (DefaultTableModel) mainTable1.getModel();
model.addRow(new Object[]{null, null, null, null, null, null, null, null, null, null, null, null, null,
null});

//reset the textfield to blank
fullNameTF.setText("");
nicknameTF.setText("");
subjectTF.setText("");
recommendTF.setText("");
nationalityTF.setText("");

//reset RadioButton and ItemBox
gradeBox.setSelectedIndex(0);
monthBox.setSelectedIndex(0);
dayBox.setSelectedIndex(0);
yearBox.setSelectedIndex(0);

// update the number of students
int numberOfStudent = counter;
displayTotalTF1.setText(Integer.toString(numberOfStudent));
displayTotalTF2.setText(Integer.toString(numberOfStudent));

radioMale.setSelected(true);

// sync mainTable1 and editTable1
SyncTable();
}
```

2: Sorting Table

```
String boxItem = (String) sortingBoxEdit.getSelectedItem();

// When inside of the box is " Alphabet "
if (boxItem.equals("Alphabet")) // if selected item in combo box is search by : Full Name
{
SortAndSearch s = new SortAndSearch();
s.selectionSortOfStudentName(students); // change row to the i / maybe row is better

if (students.size() <= editTable1.getRowCount()) {
for (int i = 0; i < students.size(); i++) {
        editTable1.setValueAt(students.get(i).getfullName(), i, 0);
        editTable1.setValueAt(students.get(i).getnickName(), i, 1);
        editTable1.setValueAt(students.get(i).getnationality(), i, 2);
```

```
            editTable1.setValueAt(students.get(i).getgender(), i, 3);
            editTable1.setValueAt(students.get(i).getgrade(), i, 4);
            editTable1.setValueAt(students.get(i).getsubject(), i, 5);
            editTable1.setValueAt(students.get(i).getstartDate(), i, 6);
            editTable1.setValueAt(students.get(i).getwhoRecommend(), i, 7);
    }// search the editTable1 and then display on table2
    }
    }
```

3: Searching Table

```
    String boxItem2 = (String) searchBoxEdit.getSelectedItem();

    // Search by full name when box displays "Full Name"
    if (boxItem2.equals("Full Name"))
    {
    SortAndSearch s = new SortAndSearch();
    int result = s.searchByFullName(students, searchTF.getText());

    if (result == -1) {
    JOptionPane.showMessageDialog(null, "No results found");
    } else {
    int secondCounter = 0;
    for (int i = 0; i < students.size(); i++) {

            if (students.get(i).getfullName().equals(searchTF.getText())) {
            editTable2.setValueAt(students.get(i).getfullName(), secondCounter, 0);
            editTable2.setValueAt(students.get(i).getnickName(), secondCounter, 1);
            editTable2.setValueAt(students.get(i).getnationality(), secondCounter, 2);
            editTable2.setValueAt(students.get(i).getgender(), secondCounter, 3);
            editTable2.setValueAt(students.get(i).getgrade(), secondCounter, 4);
            editTable2.setValueAt(students.get(i).getsubject(), secondCounter, 5);
            editTable2.setValueAt(students.get(i).getstartDate(), secondCounter, 6);
            editTable2.setValueAt(students.get(i).getwhoRecommend(), secondCounter, 7);
            secondCounter++;
            }
    }
    }
    }
```

This applies for search by nickname and nationality

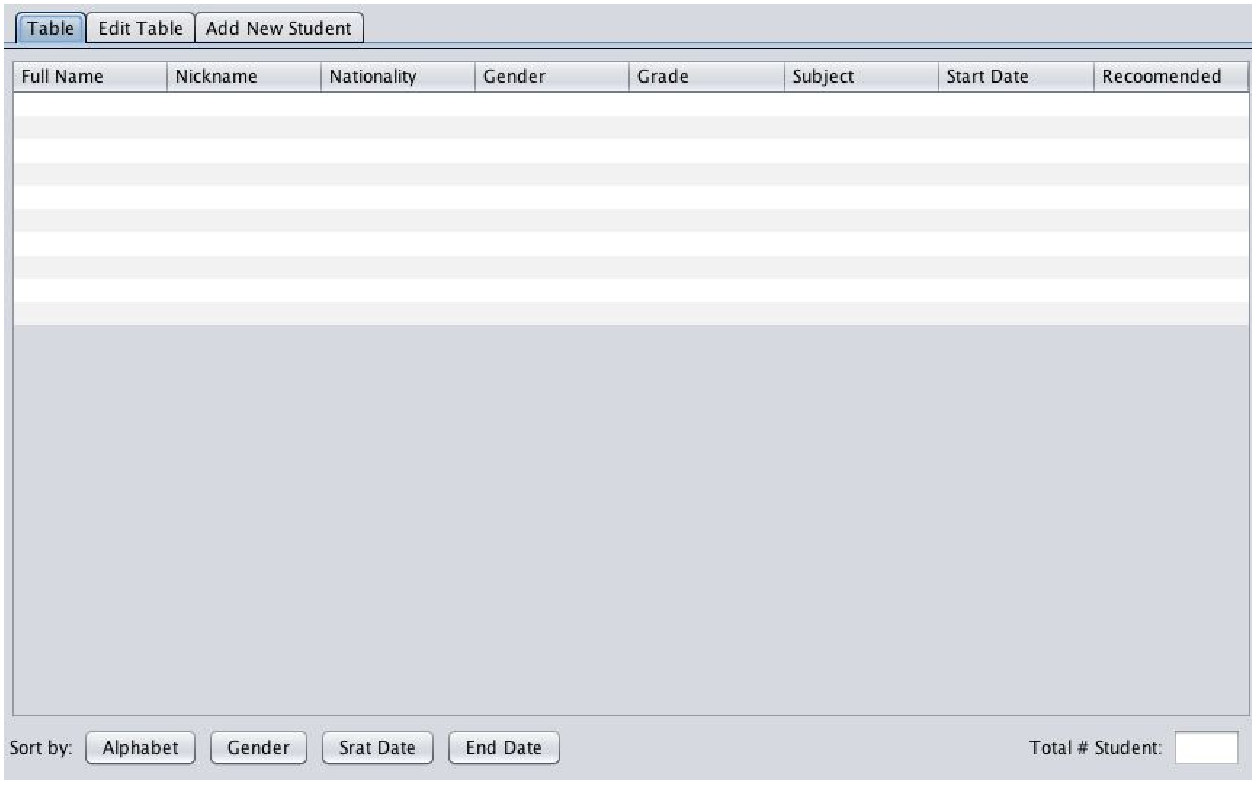
4: Retrieve all information for edit

```
updatePanel.setVisible(true);
panelSelect.setVisible(false);

// Display current student's information
DefaultTableModel model = (DefaultTableModel)editTable2.getModel();
int i = editTable2.getSelectedRow();

fullNameTF1.setText(model.getValueAt(i, 0).toString());
nicknameTF1.setText(model.getValueAt(i, 1).toString());
nationalityTF1.setText(model.getValueAt(i, 2).toString());

if (radioMale.isSelected()) {
radioMale1.setSelected(true);
}
else
{
radioFemale1.setSelected(true);
}

// get first two letter from the mm/dd/yyyy
String s = editTable2.getValueAt(i, 6).toString();
String upToCharacterMonth = "";
String upToCharacterDay = "";
String upToCharacterYear = "";
upToCharacterMonth = s.substring(1, Math.min(s.length(), 2)); // mm
upToCharacterDay = s.substring(3, Math.min(s.length(), 5)); // dd
upToCharacterYear = s.substring(6, Math.min(s.length(), 10)); // yyyy

gradeBox1.setSelectedItem(model.getValueAt(i, 4));
monthBox1.setSelectedItem(""+upToCharacterMonth);
dayBox1.setSelectedItem(""+upToCharacterDay);
yearBox1.setSelectedItem(""+upToCharacterYear);

subjectTF1.setText(model.getValueAt(i, 5).toString());
recommendTF1.setText(model.getValueAt(i, 7).toString());

}

private void updateNewButtonActionPerformed(java.awt.event.ActionEvent evt) {

// Update the selected row
```

```
DefaultTableModel model2 = (DefaultTableModel)editTable1.getModel();
DefaultTableModel model3 = (DefaultTableModel)mainTable1.getModel();
int i = editTable2.getSelectedRow();

String gender = "Female";
if (radioMale1.isSelected()) {
gender = "Male";
}

// select the selected grade from combo box
String grade = "";
grade = gradeBox1.getSelectedItem() + "";

// display staart date in form of : month / day / year
String startday = (String) dayBox1.getSelectedItem();
String startmonth = (String) monthBox1.getSelectedItem();
String startyear = (String) yearBox1.getSelectedItem();

String startDate = "" + startmonth + "/" + startday + "/" + startyear;

if ( i <= 0 )
{
editTable2.setValueAt(fullNameTF1.getText(), i, 0);
editTable2.setValueAt(nicknameTF1.getText(), i, 1);
editTable2.setValueAt(nationalityTF1.getText(), i, 2);
editTable2.setValueAt(""+gender, i, 3);
editTable2.setValueAt(""+grade, i, 4);
editTable2.setValueAt(subjectTF1.getText(), i, 5);
editTable2.setValueAt(""+startDate, i, 6);
editTable2.setValueAt(recommendTF1.getText(), i, 7);

}
updatePanel.setVisible(false);
panelSelect.setVisible(true);
```

Structure of the Program

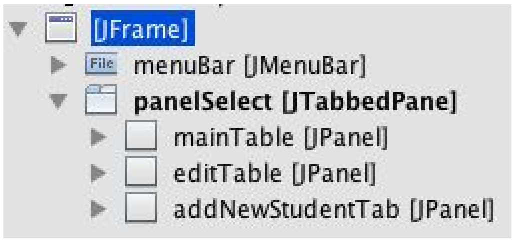
Figure 1: Main screen



Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoomended

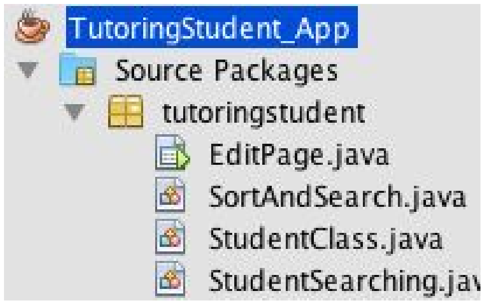
This is the main screen of this program. From the upper part of the figure 1, there are three tabs which user can selection either, just view table, edit table, or add new student to the table. Instead of using different jFrames, I used panels to simplify its codes.

Figure 2: jFrame and panels



Since all panels are included in one jFrame, I do not have to use multiple jFrame and codes that connects each jFrames. In other words, every codes written in this jFrame can be shared at any class in this jFrame.

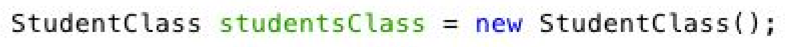
Figure 3: Classes



```
TutoringStudent_App
  Source Packages
    tutoringstudent
      EditPage.java
      SortAndSearch.java
      StudentClass.java
      StudentSearching.jav
```

I have used multiple class to create this program; therefore, it requires me to convert each class as object.

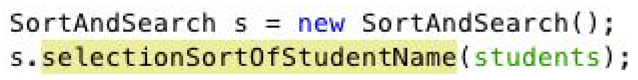
Figure 4: Create object in class



```
StudentClass studentsClass = new StudentClass();
```

This code is used in EditPage.java (figure3). Without using this object, all codes in StudentClass.java (figure3) can not be used in EditPage.java. It is possible to combine all class in EditPage.java, but this is more organized than stuffing all code in a one place.

Figure 5: Create object in class part 2



```
SortAndSearch s = new SortAndSearch();
s.selectionSortOfStudentName(students);
```

This is the different code for creating an object in a class. I have used this to retrieve the a mechanism of searching and sorting from EditPage.java.

Overall, this is how each class are linked together.

The major reason why I have separated each mechanism in different classes is for simplicity. If all mechanisms are stuffed in one class, I have to keep track on all codes at one time when adding or deleting the codes. If there are errors, I need to look over how and why its occur. Encapsulating the logics, which means separating into small pieces, is more efficient than working on all codes at once.

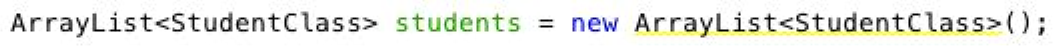
Figure 6: List of logic contained in each class (Encapsulation)

- EditPage.java : GUI, the actual application.
- SortAndSearch.java : Logic of sorting / searching
- StudentClass.java : get and set method

This type of program is called Object-Oriented Programming (OOP). In my opinion, the special thing about OOP is encapsulation because dividing the program into small piece is very efficient and organize. I can avoid finding through every single lines to detect the mistakes. In addition, each classes are different thing. Hence any errors in one of the classes will not affected. So any errors in StudentClass.java does not directly affect on the codes written in EditPage.java.

Data Structure Used

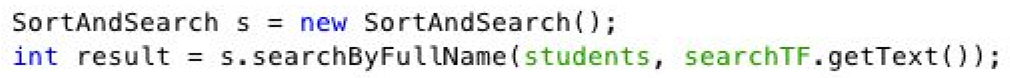
Figure 1: ArrayList



```
ArrayList<StudentClass> students = new ArrayList<StudentClass>();
```

I used an ArrayList of students object. This object contains multiple attributes, which is the basic information of the students. For instance, full name, nickname, gender, grade, start date, nationality, subject of study, person who recommend.

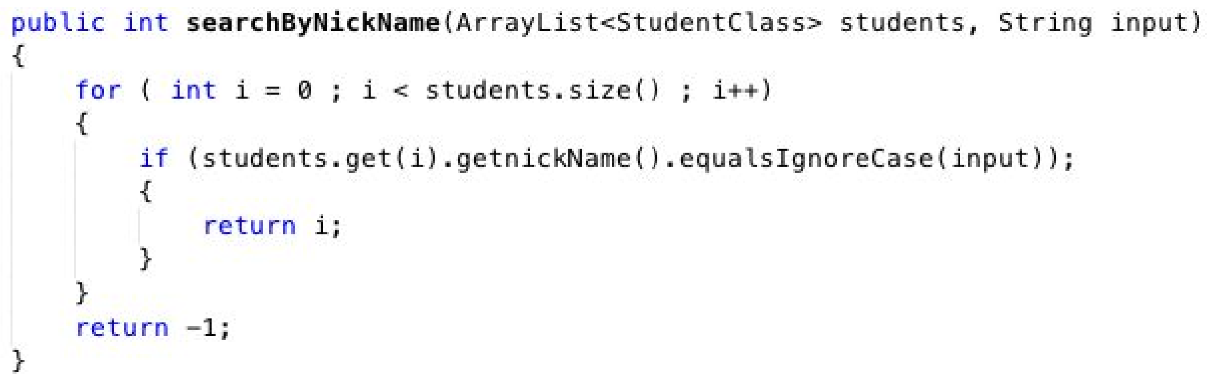
Figure 2: Using ArrayList



```
SortAndSearch s = new SortAndSearch();
int result = s.searchByFullName(students, searchTF.getText());
```

For example, when the client wants to search from the table, the comparison between input and student object is necessary.

Figure 3: Algorithm for searching



```
public int searchByNickName(ArrayList<StudentClass> students, String input)
{
    for ( int i = 0 ; i < students.size() ; i++)
    {
        if (students.get(i).getnickName().equalsIgnoreCase(input));
        {
            return i;
        }
    }
    return -1;
}
```

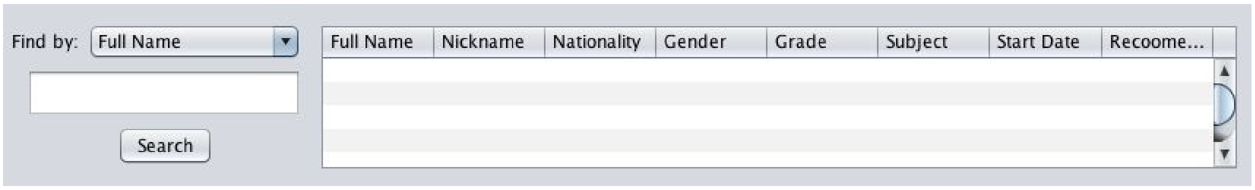
ArrayList(StudentClass) = students object. String input = searchTF.getText() = what user just typed in. This is comparison between full name of the student that is in students object and what user typed in.

Figure 4: Example of students object

# of student	Full name	NickName	Nationality	Grade	… and more
1st student	John Smith	John	USA	12	...
2nd student	Satou Hiroshi	Hiro	Japan	10	...
3rd student	Jungseung Kim	Kim	Korea	8	...

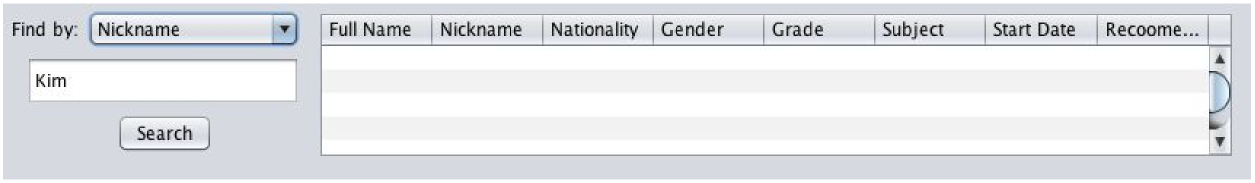
I used ArrayList because I wanted to store each students' information as sets. From figure 4, first row of ArrayList students (figure1) stores all information of 1st student's, which is John Smith. His nickname, nationality, grande, and more details are only available in this row. The second row of ArrayList students stores all information of 2ns student's, which is Satou Hiroshi. Again, his nickname, nationality, grade, and more details are only available in this row. This process would repeated as long as the client are adding new students.

Figure 5: Searching inside of ArrayList



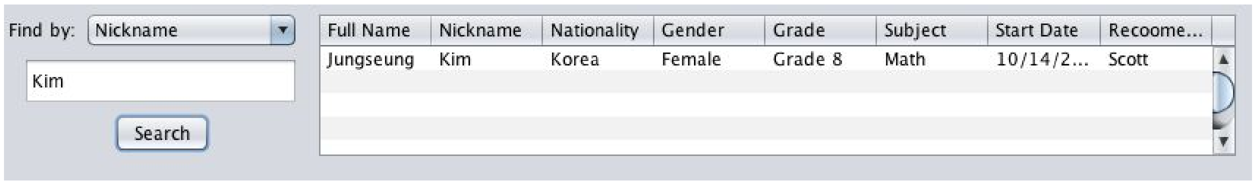
Assuming that my client want to get the information of Jungseung Kim, this is where ArrayList is useful.

Figure 6



After client inputted Jungseung Kim's nickname "Kim" in a text field and then pressed Search button, this "Kim" will be compared with "Kim" that is stored in ArrayList somewhere. First, "Kim" would compared with 1st ArrayList's nickname "John", which is wrong. Second, "Kim" would be compared with 2nd ArrayList's nickname "Hiro", which is also wrong. Lastly, “Kim”would be compared with 3rd ArrayList’s nickname “Kim”, which is correct.

Figure 7: Result

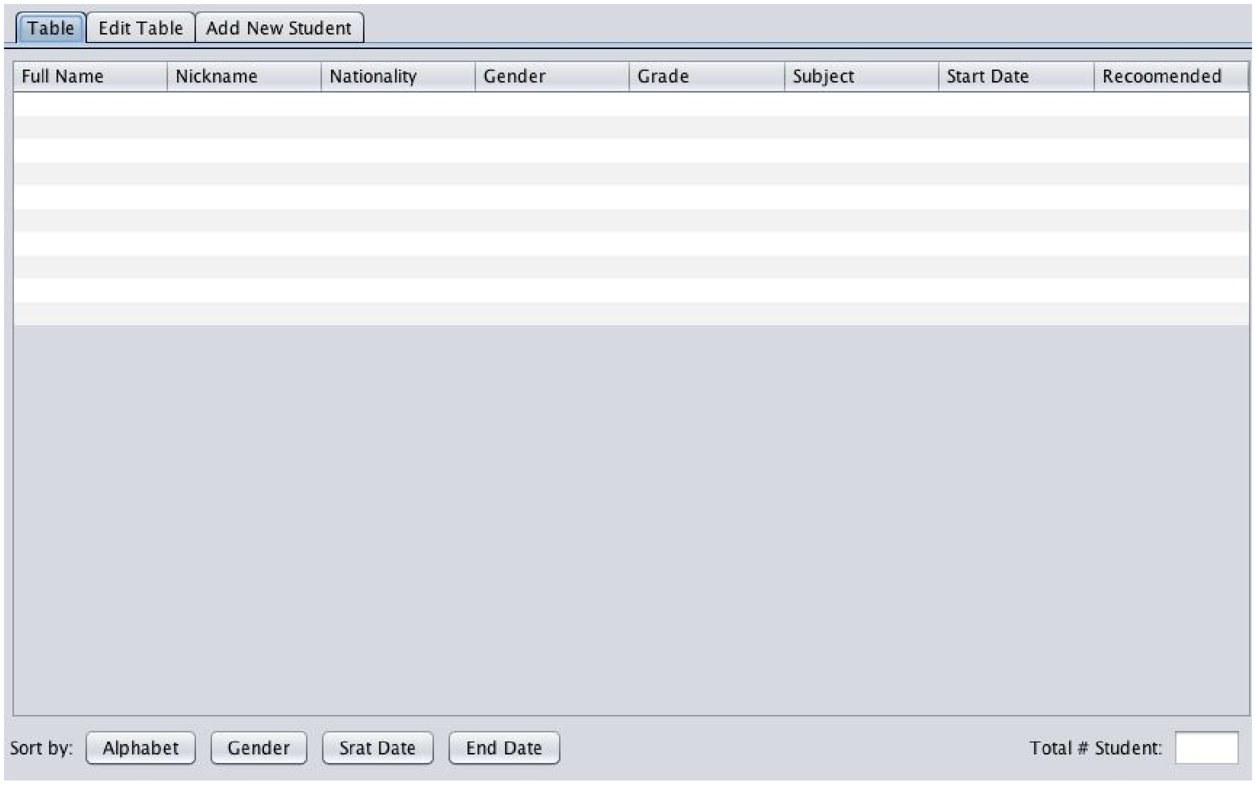


Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoome...
Jungseung	Kim	Korea	Female	Grade 8	Math	10/14/2...	Scott

As a result, 3rd ArrayList students will be appear on a table because this ArrayList contain what user is looking for while rest of ArrayList does not. Overall, every single line of Arraylist Student is the informations of the different students.

User Interface / How GUI Works

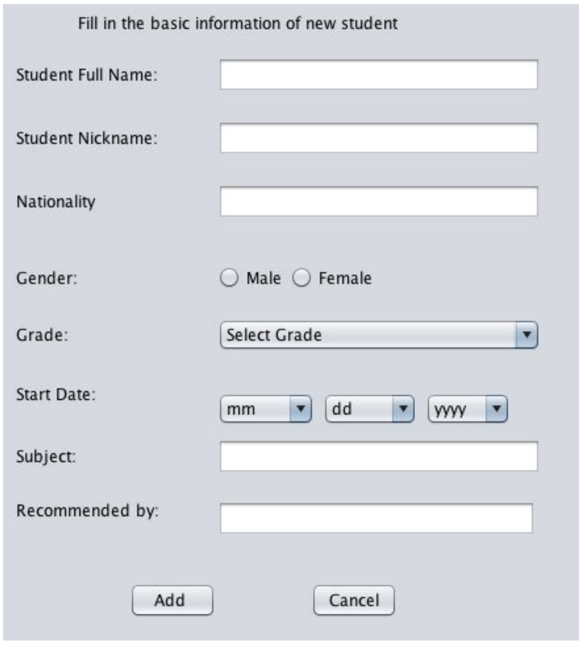
1) Main Page



Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoomended

This is the main page.

2) Add information of students

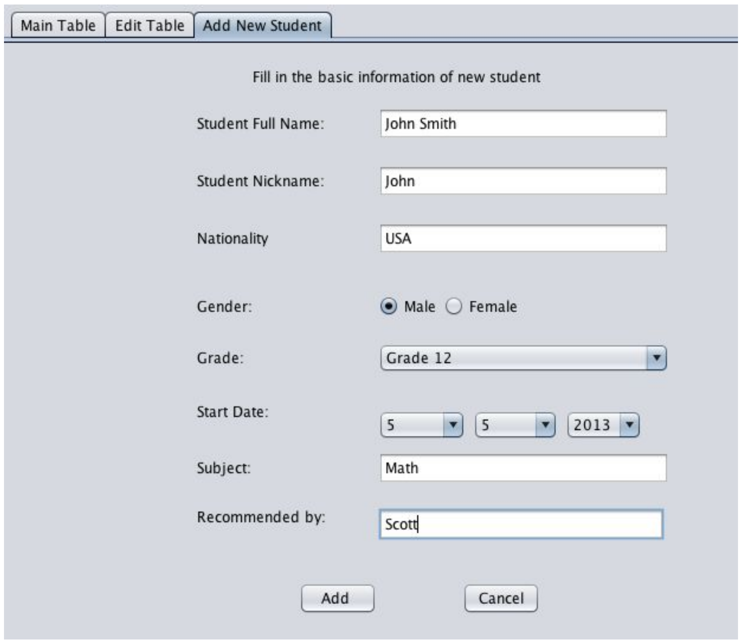

Fill in the basic information of new student

Student Full Name:	
Student Nickname:	
Nationality	
Gender:	Male Female
Grade:	Select Grade
Start Date:	mm dd yyyy
Subject:	
Recommended by:	

This appears at “Add New Student” tab where client can input his student’s information.

---

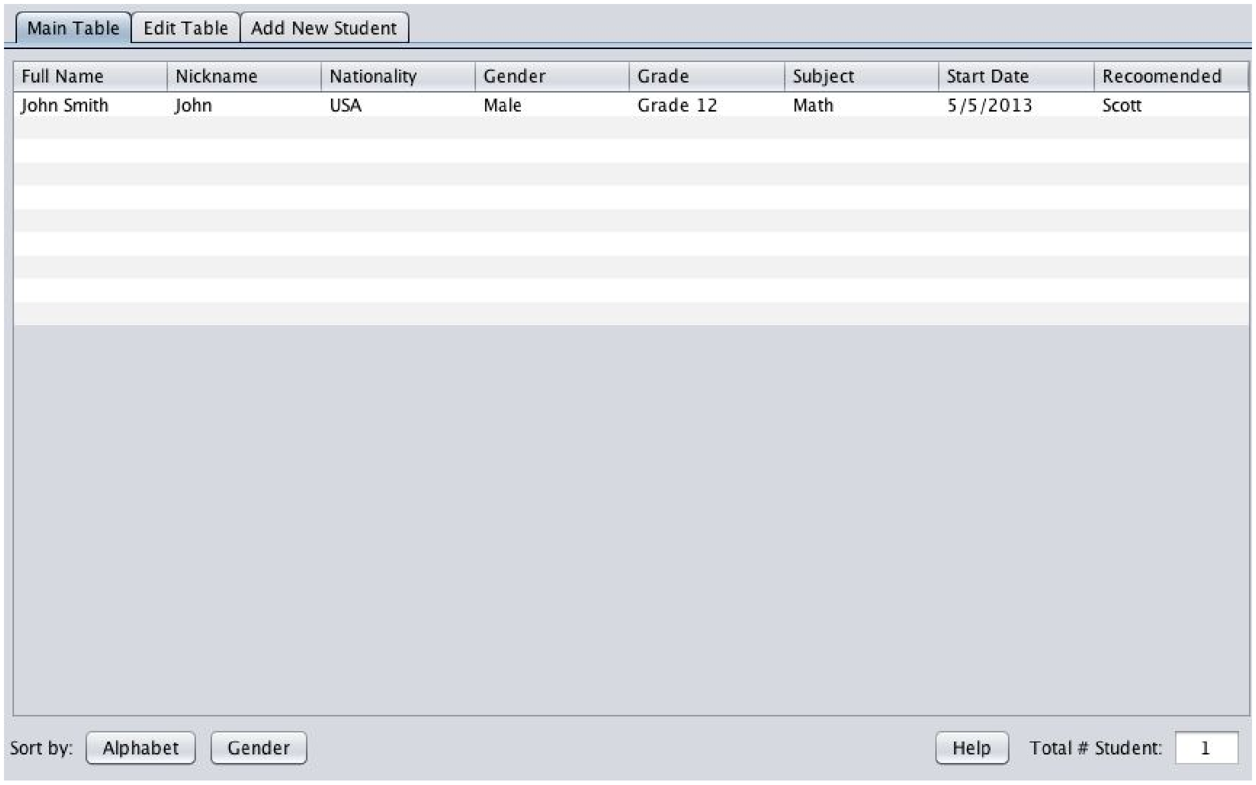
3) Adding information of students



Fill in the basic information of new student

Student Full Name:	John Smith
Student Nickname:	John
Nationality	USA
Gender:	[✓ Male] Female
Grade:	Grade 12
Start Date:	5 / 5 / 2013
Subject:	Math
Recommended by:	Scott

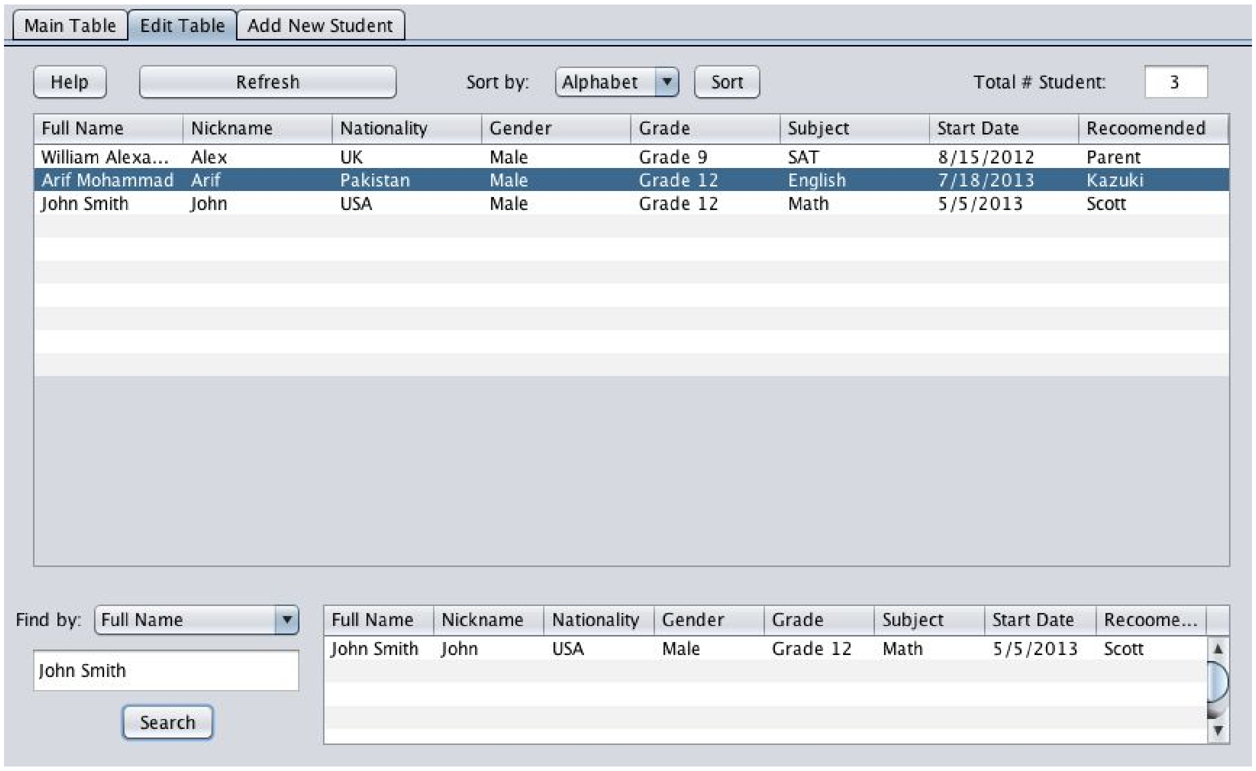
When user clicked Add button, all inputted information will be appear in Main Table.



Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoomended
John Smith	John	USA	Male	Grade 12	Math	5/5/2013	Scott

Total # Student: 1

3)Edit Page Overview

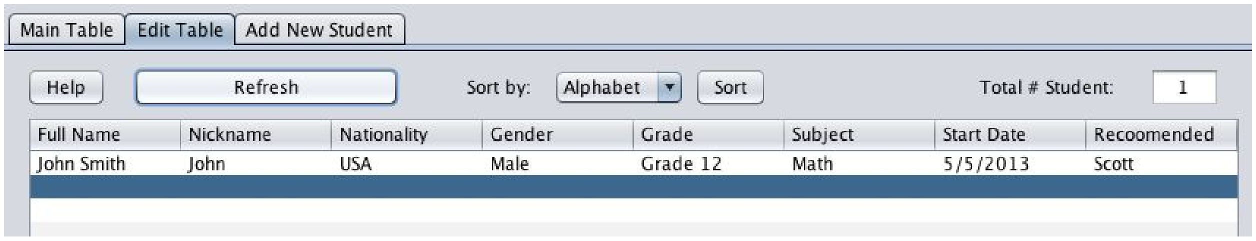


Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoomended
William Alexa...	Alex	UK	Male	Grade 9	SAT	8/15/2012	Parent
Arif Mohammad	Arif	Pakistan	Male	Grade 12	English	7/18/2013	Kazuki
John Smith	John	USA	Male	Grade 12	Math	5/5/2013	Scott

Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoome...
John Smith	John	USA	Male	Grade 12	Math	5/5/2013	Scott

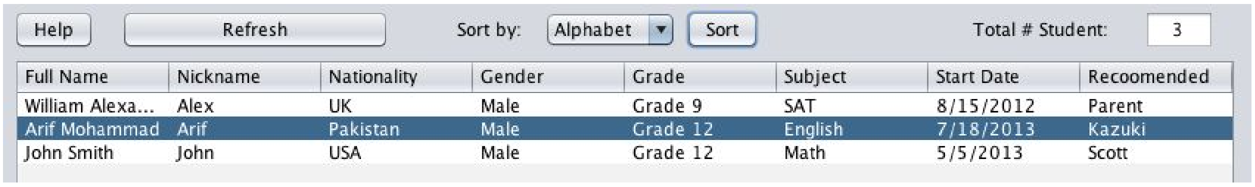
This is the overview of the Edit Table tab. Upper part is for sorting. Buttom part is for searching. This photo have “Refresh button”, but client is no longer require to press this button due to automation.

4) Sorting



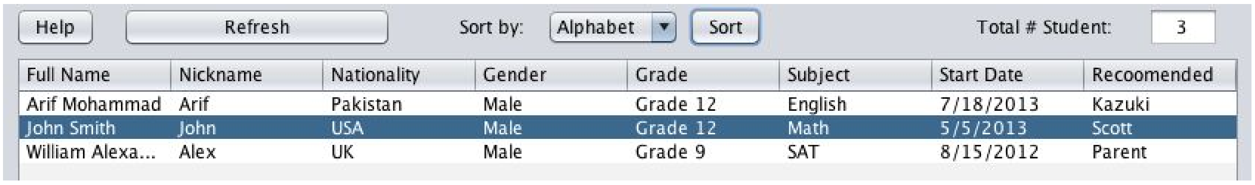
Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoomended
John Smith	John	USA	Male	Grade 12	Math	5/5/2013	Scott

At “Edit Table” tab, the other table will appear. Once user clicked Refresh button, all information from Main Table would appear in this page. Where user can sort and search. The table will be sorted by alphabetical order, nickname, grade, gender, etc.



Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoomended
William Alexa...	Alex	UK	Male	Grade 9	SAT	8/15/2012	Parent
Arif Mohammad	Arif	Pakistan	Male	Grade 12	English	7/18/2013	Kazuki
John Smith	John	USA	Male	Grade 12	Math	5/5/2013	Scott

↓

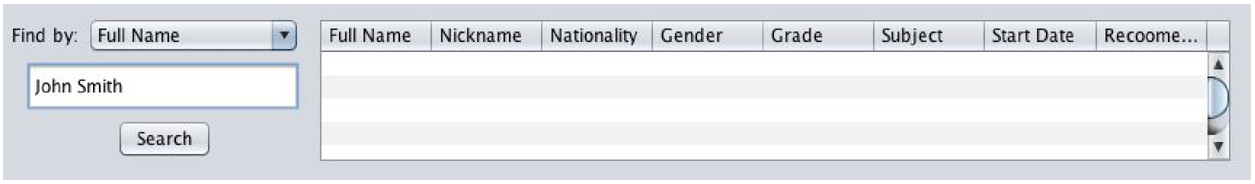


Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoomended
Arif Mohammad	Arif	Pakistan	Male	Grade 12	English	7/18/2013	Kazuki
John Smith	John	USA	Male	Grade 12	Math	5/5/2013	Scott
William Alexa...	Alex	UK	Male	Grade 9	SAT	8/15/2012	Parent

Sorted.

---

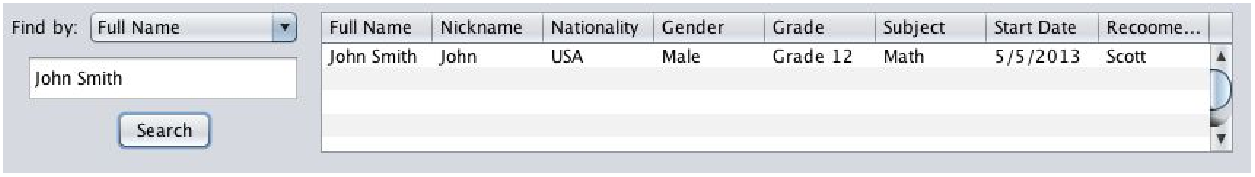
5) Searching



John Smith

Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoome...

In this text field, user can search student's information by type in full name. The client can switch between full name and other conditions.



John Smith

Full Name	Nickname	Nationality	Gender	Grade	Subject	Start Date	Recoome...
John Smith	John	USA	Male	Grade 12	Math	5/5/2013	Scott

If inputted name does exist in ArrayList, the result will appear.

Software Tools Used : NetBeans
According to techpedia:

NetBeans is an open-source integrated development environment (IDE) for developing with Java, PHP, C++, and other programming languages. NetBeans is also referred to as a platform of modular components used for developing Java desktop applications (techpedia).