

Criterion C: Development

I. Introduction

The product is a Java Program. The application is an IASAS database that manages all students and passport and visa information in an organized way.

Word Count: 25

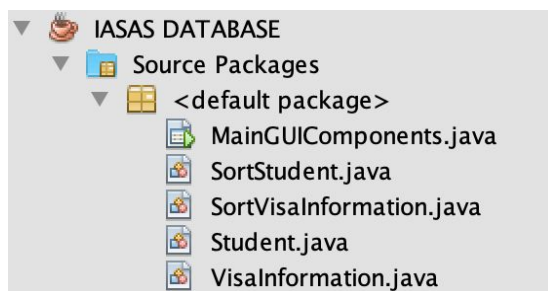
II. Summary List of Key Techniques

- Sorting (Selection/Bubble sort)
- Use of BufferedImage to load a file through fileChooser GUI
- ArrayList/Array of Objects
- Saving and Reading local files
- Linking ArrayLists
- parameter passing
- nested loops
- user defined objects made from an OOP "template" class
- Encapsulation, polymorphism, inheritance
- Error handling (try, catch)
- Option pane generation for communicating with the user
- Use of a flag value (such as 999999999, or "not set yet")
- Specialized library imported (Input Streamer, File, Out Streamer, etc.)
- Linked List functionalities: "add", "check to see if it's empty", remove, etc.
- Parsing value from one data type to another
- split() method is used to separate a String variable into parts.

Word Count: 0 words (Bullet Lists)

III. Structure of the Program

All Classes



All Tabs



The program is a database that stores and organizes students and the corresponding visa information. Each information has its own attributes in the ArrayLists of objects which stores different *types* of information.

My program consists of six different classes; one being the application sample form and the rest being template classes. By breaking my program into classes and packages, I have taken an Object-Oriented approach. The use of template classes allow me to utilize the OOP features of polymorphism and encapsulation, as well as abstraction.

The Student and Visa Information object template classes were made in order to use the attributes and accessor methods in the defined arraylists. Aggregation is applied as Visa Information Class has some student information from the Student Class to add corresponding visa information.

- Polymorphism: The Student and Visa Information class contains an overloaded constructor.
- Encapsulation: The methods and attributes of a class are made private and can only be accessed through public get and set methods.

Two remaining classes SortStudent Class and SortVisaInformation Class are for sorting variables in the other two classes. Each class sorts 4 variables both in ascending and descending order, yielding total 8 methods.

- Abstraction: The separation of classes allows me to only use the features of the algorithm that are relevant to me, This has increased my efficiency with testing, coding and identifying problems by focusing on one problem at a time.

Word Count: 232 words

IV. Key Data Structures Used

1. ArrayList of Objects

ArrayList is a data structure that is used to store elements of objects and strings. This is preferred over arrays as elements can be inserted at or deleted from a particular

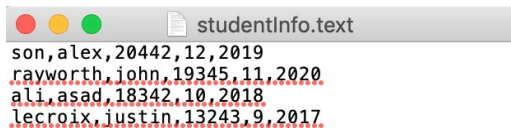
position so as a high variety of methods in manipulating stored methods. Moreover, flag values won't affect sorting for arraylists. It also creates enough space to load all the data from a local file without knowing the size of the data.

```
ArrayList<VisaInformation> visaRequiredStudents = new ArrayList<VisaInformation>();  
ArrayList<Student> students = fileRead_ST();  
ArrayList<VisaInformation> visainformation = fileRead_VI();
```

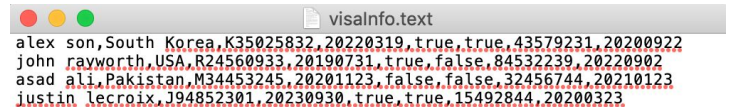
2. File Class

The File class was used to save/store data of attributes and can be accessed after the program stops running and restarts again. This is done through the use of the FileWriter and FileReader classes.

```
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.InputStreamReader;  
import java.io.OutputStreamWriter;
```



studentInfo.txt
son,alex,20442,12,2019
rayworth, john,19345,11,2020
ali,asad,18342,10,2018
lecroix,justin,13243,9,2017



visainfo.txt
alex son, South Korea, K35025832, 20220319, true, true, 43579231, 20200922
john rayworth, USA, R24560933, 20190731, true, false, 84532239, 20220902
asad ali, Pakistan, M34453245, 20201123, false, false, 32456744, 20210123
justin lecroix, J94852301, 20230930, true, true, 15492844, 20200323

Figure 1: Save file for Student Arraylist

Figure 2: Save file for Visa Information Arraylist

3. Array

Array is used for direct access to all the variables in the array. It was used to assign the information in the index of the array to the arraylist for save and read method.

Word Count: 147

V. Algorithms Explained

Input Information (With Error Handling):

```
String lastName = lastNameTF.getText();
String firstName = firstNameTF.getText();
int idNum = Integer.parseInt(idNumTF.getText());
String grade = gradeCB.getSelectedItem() + "";
int year = Integer.parseInt(yearTF.getText());
```

```
Student a = new Student(lastName, firstName, idNum, grade, year);
students.add(a);
```

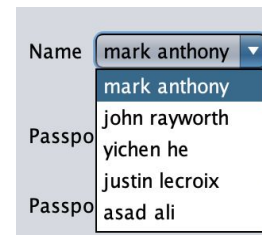
Information is added to the arraylist when the user types in the text fields.

Create a fullName variable and add it to the combo box in the “Visa Information” tab:

```
//add full name to the arrayList and populate it over the combo box.
String fullName = firstName + " " + lastName;
studentNameVisaCB.addItem(fullName);
```

Save the student information in the text file (Error Handling with JOptionPane for not typing full information):

```
fileSave_ST(students);
```



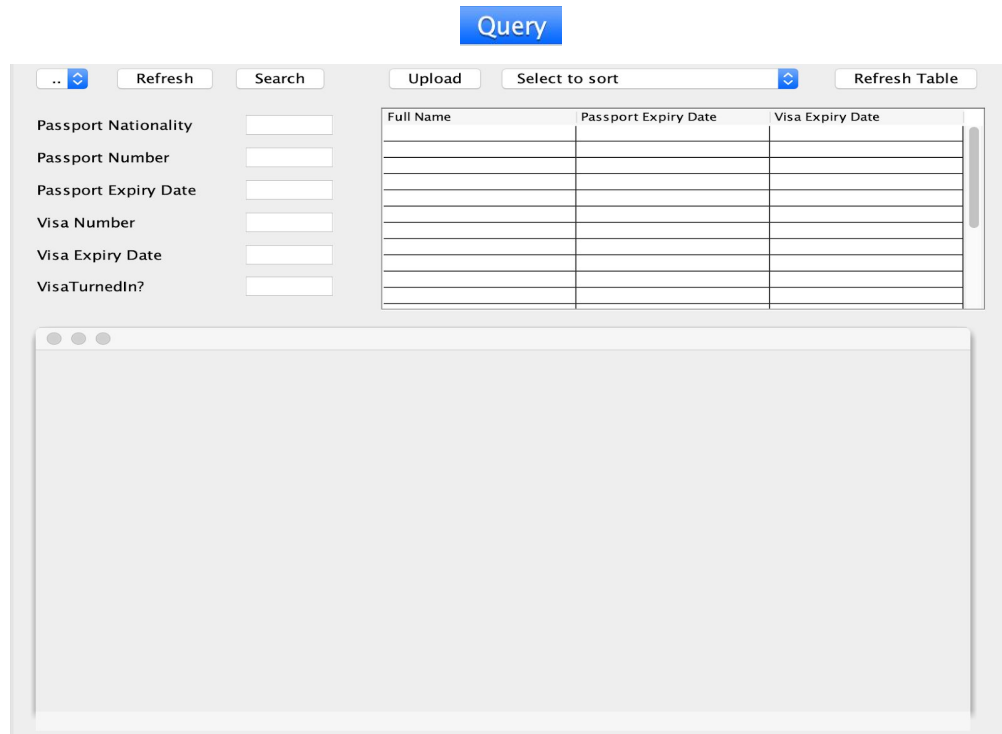
```
if(lastName.equals("") || firstName.equals("")) {
    JOptionPane.showMessageDialog(null, "Please type in all the information!", "Error", ERROR_MESSAGE);
}else {
    Student a = new Student(lastName, firstName, idNum, grade, year);
    students.add(a);
    //add full name to the arrayList and populate it over the combo box.
    String fullName = firstName + " " + lastName;
    studentNameVisaCB.addItem(fullName);
    fileSave_ST(students);
}
}
```

All under if-else statements with JOptionPane error message for error handling. If inappropriate or incomplete information is typed, it is not added to the arraylist at all nor saved.



Utility of Query Tab

This tab is dedicated for students with a “visaRequired” checkbox as “true.” A screenshot is illustrated for clarity in my references.



```
ArrayList<VisaInformation> visaRequiredStudents = new ArrayList<VisaInformation>();
```

Clicking the “Refresh” button, the visaRequiredForStudents() method is runned:

```
private void visaRequiredForStudents() {  
    searchComboBox.removeAllItems();  
    for (int i = 0; i < visainformation.size(); i++) {  
        if (visainformation.get(i).getVisaRequired()) {  
            searchComboBox.addItem(visainformation.get(i).getFullName());  
            visaRequiredStudents.add(visainformation.get(i));  
        }  
    }  
}
```

Students in the VisaInformation arraylist who have the “visaRequired” attribute as true will be added to the visaRequiredStudents ArrayList and the combo box.



Clicking the “search” button sets the appropriate JTextFields to the information from the ArrayList visaRequiredStudents with iteration. Information: visaRequiredStudents.get(i).getVariable.

```
private void visaQueryVisaInformation() {  
    for (int i = 0; i < visaRequiredStudents.size(); i++) {  
        if (searchComboBox.getSelectedItem().equals(visaRequiredStudents.get(i).getFullName()) {  
            searchPassportNationalityTF.setText(visaRequiredStudents.get(i).getPassportNationality() + "");  
            searchPassportNumberTF.setText(visaRequiredStudents.get(i).getPassportNumber() + "");  
            searchPassportExpiryDateTF.setText(visaRequiredStudents.get(i).getPassportExpiryDate() + "");  
            searchVisaNumberTF.setText(visaRequiredStudents.get(i).getVisaNumber() + "");  
            searchVisaExpiryDateTF.setText(visaRequiredStudents.get(i).getVisaExpiryDate() + "");  
            searchVisaTurnedInTF.setText(visaRequiredStudents.get(i).getVisaTurnedIn() + "");  
        }  
    }  
}
```


Clicking the “Upload” Button:

```
private void uploadImageButtonMouseReleased(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    BufferedImage img;  
    File file;  
  
    JFileChooser fileChooser = new JFileChooser();  
    fileChooser.showOpenDialog(null);  
    file = fileChooser.getSelectedFile();  
    try {  
        img = ImageIO.read(file);  
        Image resizedImage = img.getScaledInstance(imageInternalFrame.getWidth(),  
            imageInternalFrame.getHeight(), Image.SCALE_SMOOTH);  
        ImageIcon icon = new ImageIcon(resizedImage); // ADDED  
        jLabel12.setIcon(icon); // ADDED  
    } catch (IOException e1) {  
        JOptionPane.showMessageDialog(null, "Please select an image.");  
    }  
  
    imageInternalFrame.setVisible(true);  
}
```

This program utilizes the BufferedImage subclass which extends the Image superclass. This allows the application to access and operate directly on the image, thus, displaying any uploaded image to the frame. Oversize image will be resized according to the internal frame. The image is uploaded as an icon in JLabel.

Actual Implementation:



Sample image from [Pennsylvania State](http://www.pennsylvania.gov)

information is only if the old name is equal to the preexisting name. This is done so the change of name in student ArrayList would also be applied in VisaInformation.

```
private void searchEditName(String oldName){  
  
    for(int i=0; i<students.size(); i++) {  
        if((visainformation.get(i).getFullName()).equals(oldName)) {  
            visainformation.get(i).setFullName(students.get(i).getFirstName() + " "  
                + students.get(i).getLastName());  
        }  
    }  
}
```

Clicking the “remove” button would remove student information as well as the visa information for that row. Visa information cannot exist without any corresponding student visa information.

```
private void removeStudentButtonMouseReleased(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    JFrame f;  
    f = new JFrame();  
    int row = Integer.parseInt(rowToChooseStudentTF.getText()) - 1;  
    if (JOptionPane.showConfirmDialog(f, "Are you sure?") == JOptionPane.YES_OPTION) {  
        students.remove(row);  
        visainformation.remove(row);  
    }  
}
```

Clicking the “save” button:

```
private void saveStudentButtonMouseReleased(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    JFrame f;  
    f = new JFrame();  
    int row = Integer.parseInt(rowToChooseStudentTF.getText()) - 1;  
    if (JOptionPane.showConfirmDialog(f, "Are you sure?") == JOptionPane.YES_OPTION) {  
        fileSave_ST(students);  
        fileSave_VI(visainformation);  
    }  
}
```

Changes in name in Student ArrayList must also be applied to the Visa Information. They share the “name” variables in common.

Confirm dialog is used to verify a user's decision.



File Save Method

```
// File Save Visa Information. Will save visa information data to a visaInfo Text File
private void fileSaveForVisaInformation(ArrayList<VisaInformation> visainformation) {
    BufferedWriter bw = null; // create bw variable
    String testFile = "./visaInfo.txt"; //create a path
    File file = new File(testFile); // initialize file var
    if (file.exists()) { // if the file with same name exists delete
        file.delete();
    }
    try {
        file.createNewFile(); //create a new file at that location
    } catch (IOException e) {
        e.printStackTrace(); // printing out error
    }

    try {
        bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file, false), "UTF-8"));
        StringBuilder data = new StringBuilder();
        for (int i = 0; i < visainformation.size(); i++) { // save info/data with a comma for each variable
            data.append(visainformation.get(i).getFullName());
            data.append(", " + visainformation.get(i).getPassportNationality());
            data.append(", " + visainformation.get(i).getPassportNumber());
            data.append(", " + visainformation.get(i).getPassportExpiryDate());
            data.append(", " + visainformation.get(i).getVisaRequired());
            data.append(", " + visainformation.get(i).getVisaTurnedIn());
            data.append(", " + visainformation.get(i).getVisaNumber());
            data.append(", " + visainformation.get(i).getVisaExpiryDate());
            data.append("\n");
        }
        System.out.println(data);
        bw.write(data.toString());
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (bw != null) {
                bw.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    System.out.println("File is written successfully");
}
}
```

The file save code uses bufferedwriter and file class. The first few lines of code initializes bufferedwriter and file variable, as well as file path that will be used later. It will create a new file at that path. After data input, it is organized in a way, as illustrated in Figure 3. However, the for loop will iterate the ArrayList and organize all the data in one line with commas separating each object variable, as illustrated in Figure 4. This is done in order to organize one set of student/visa information in each line. Try catch approach was taken for potential error handling.

Figure 3
Before Saving
Son
Ganghee
20442
12
2020

Figure 4
Format of the data in the file
Son,Ganghee,20442,12,2020

File Read Method

```
File Read Visa Information. Will be loaded when the program is runned.
private static ArrayList<VisaInformation> fileReadForVisaInformation() {
    String path = "./visaInfo.txt"; // path
    BufferedReader br = null; // initialize
    ArrayList<VisaInformation> visainformation = new ArrayList<VisaInformation>(); // locally defining visa arraylist
    File file = new File(path);
    String read = ""; // initialize read variable
    try {
        if (file.exists()) {
            FileInputStream fileInputStream = new FileInputStream(path);
            InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream, "UTF-8");
            br = new BufferedReader(inputStreamReader);
            String infoData = null;
            while ((infoData = br.readLine()) != null) {
                read += infoData;
                read += "\n";
            } // input everything in the text file to the read variable
            // after each line put a symbol (means the line has ended)

            br.close(); // finishing inputting data

            br.close(); // finishing inputting data

            String[] infoArr = read.split("\n"); // separate the entire string by a certain symbol
            for (int i = 0; i < infoArr.length; i++) {
                VisaInformation add = new VisaInformation();
                visainformation.add(add);
                String[] stuArr = infoArr[i].split(","); // separate each variable with a comma

                if (stuArr.length > 0) {
                    visainformation.get(i).setFullName(stuArr[0]);
                    visainformation.get(i).setPassportNationality(stuArr[1]);
                    visainformation.get(i).setPassportNumber(stuArr[2]);
                    visainformation.get(i).setPassportExpiryDate(Integer.parseInt(stuArr[3]));
                    visainformation.get(i).setVisaRequired("true".equals(stuArr[4]));
                    visainformation.get(i).setVisaTurnedIn("true".equals(stuArr[5]));
                    visainformation.get(i).setVisaNumber(stuArr[6]);
                    visainformation.get(i).setVisaExpiryDate(Integer.parseInt(stuArr[7]));
                }
                // assign the data in each variable to the ArrayList VisaInformation
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (br != null) {
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    } // Error Handling

    return visainformation;
}
}
```

In a nutshell, the File Read code has a path to “visaInfo.txt” where the saved data is, and recollects the data. The first section is similar to the File Save code. Inside the for loop, the split method will catch the commas and set the format to original like Figure 3. Then the data from each variable will be stored into a locally defined string array for direct access. As each index of the array has the information of the arraylist, the information is “set” to the ArrayList every time the program is implemented.

The following initialization sets the information from the arraylist from the text file.

```
ArrayList<Student> students = fileRead_ST();  
ArrayList<VisaInformation> visainformation = fileRead_VI();
```

Word Count: 504 (excluding fragmented sentences)

VI. SoftWare Tools Used

NetBeans 8.2, an integrated developmental program for java, was chosen due to the advantage of user friendly and powerful GUI Builder, object oriented programming approach, and access to free and open source prewritten code libraries from the extension of java swings components. An extensible platform which allows adding own Netbeans features decreases development time.

Word Count: 55 words

Total Word Count: 963 words