Criterion C - Development

Introduction

The product is a Java program coded on Netbeans that allows Mr.Callahan to do the following:

- Import csv file of the spreadsheet he uses currently to keep track of tutors and tutees
- Manually input new tutors and tutees to the database
- Manually edit the data of existing tutors and tutees
- Provide a list of tutors available for a new unpaired tutee based on subject area
- Pair tutors and tutees

The product was designed to allow Mr.Callahan to pair new tutees with tutors more efficiently by narrowing down tutor options for tutees based on subject and availability.

In the development process, the final product was simplified from the final prototype after realizing unfulfilled needs or encountering difficulties in code.

Word Count: 121

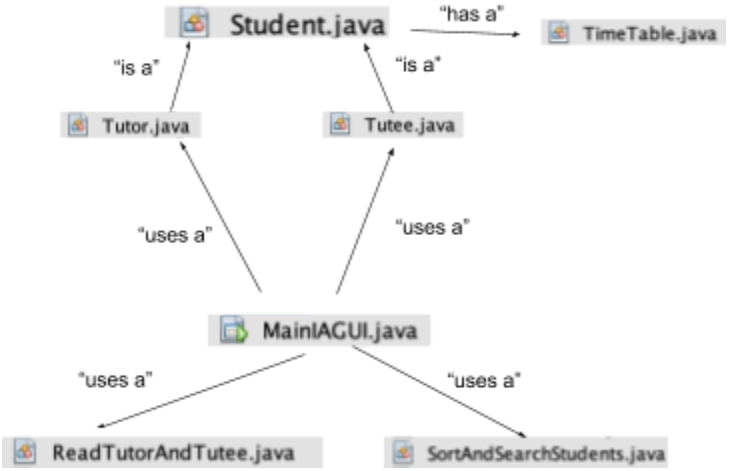
List of Techniques

- Sequential Search
- User created ToString methods
- Overloaded constructors
- User defined objects Tutees, Tutors, TimeTables
- Array of Objects
- for/while/nested loops, if/else statements
- Error Handling
- Parsing int/double/string/files
- File Parsing using Scanner and String Tokenizer (scanner.nextLine(), st.countTokens(), st.nextTokens())
- GUI Elements: JTable, JTextPane, JComboBox, JRadioButton, JCheckbox
- Decomposition by declaring and calling methods
- Overriding global variables
- Use of flag values
- Encapsulation
- Inheritance

Word Count: 0 (bulleted list)

Structure of the Program

In this program, there are a total of 7 classes. The MainIAGUI class is the class which is run when the user uses the program. The Tutor and Tutee class are classes used when creating instances of new Tutors and Tutees in the program. The Student class is the parent class of the tutor and tutee. The timeTable class is where instances of timeTables were created. Aggregation was applied as the Tutor and Tutee class both "had an" an array of timeTables as an attribute. Finally, the ReadTutorAndTutee class was used for reading csv files and the SortAndSearchStudents class was created for any sort and search methods used in the program. Overall the relationship between the classes in the program can be explained using the diagram below.



There were several reasons why an Object Oriented Program was developed for this program. Firstly, having several classes divided in the way above allowed there to be better modelling of real world relationships. Secondly, the application of inheritance by having a superclass of Students and subclasses of Tutor and Tutee reduced redundancy, consequently allowing faster development, increased reliability, and efficient testing. Thirdly, the use of encapsulation allowed data to be manipulated with security and stability and lastly, by decomposing the program into separate classes, it allowed each part of the program to be created and tested separately before putting them altogether, making the production process efficient.

Word Count: 233

Data Structures Used: Arrays

For this program, arrays of objects (Tutees, Tutors, timeTableArray) and arrays of strings (subjects, tutorTypes) were predominantly used instead of arrayLists for several reasons. First of all, being a program with a limited set of tutors and tutees, both which has numbered around 30 at our school, it was judged that there would be no need to change the length of the arrayList of tutees and tutors and as a result, tutor and tutee arrays of length 50 were created instead of an arrayList of variable Length.

Secondly, because the program required direct access to the attributes of tutors and tutees searched, it was judged that arrays would allow better direct access as there were limitations in direct access with arrayLists.

```
public class MainIAGUI extends javax.swing.JFrame {

    private Student[] students = new Student[100];
    private Tutor[] tutors = new Tutor[50];
    private Tutee[] tutees = new Tutee[50];
    private int tutorCounter = 0;
    private int tuteeCounter = 0;
    private TimeTable[] timeTableArray = new TimeTable[4];
    private Tutee searchedTutee;
    private Tutor searchedTutor;

    private String[] subjects = new String[17];
    private String[] tutorType = new String[5];
```

Word Count: 125

Key Algorithms

This program can be largely divided into three parts, manual data manipulation (manually changing/adding data), file import and tutor tutee pairing. In this section, the algorithms behind the three different parts of the program will be explained.

1. Manual Database Manipulation

To successfully operate the 'Manual Input Tab' and 'Change Data Tab', 6 main blocks of code were repeatedly used in combination with one another: (1) retrieving data input from the tab, (2) retrieving data from the arrays to the tab, (3) creating an instance of tutor/tutee, (4) setting values of an instance of tutor/tutee, (5) refreshing the tab, (6) searching for tutor/tutee.

The 'Manual Input Tab' used block 1, 3, 5 respectively and the 'Change Data Tab' used block 6, 2, 3, 5, respectively. The overlaps in code allowed the blocks of code to be often tested independently resulting in smoother testing.

To highlight a nuanced method used in the manual data manipulation part of the program, the 'ShowTutor(/Tutee/Student)Subjects' method uses a series of if/else statements in a for loop so that the subject data imported from the spreadsheet, which is not in the order of the buttons in the change data tab, can still be loaded to the radio buttons. This change was made after I realized that not putting the if/else statements in a for loop results in the program to not load subject elements not originally intended to be in its place.

```
public String[] ShowTuteeSubjectData(String[] subjects) {
    //ISSUE HERE: If else statement for ESL doesnt work when english is not selected:
    for(int i =0; i < subjects.length; i++){
        if (subjects[i].equals("English")) {
            TuteeEnglishButton.setSelected(true);
        }
        if (subjects[i].equals("ESL")) {
            TuteeESLButton.setSelected(true);
        }
        if (subjects[i].equals("Math")) {
            TuteeMathButton.setSelected(true);
```

2. Import Spreadsheet Tab

As the import tab was added to maximize usability of my program for Mr.Callahan, who had already been using a csv file to pair tutors and tutees, I needed to respect the file format which Mr.Callahan was using. The image below shows the file format which Mr.Callahan had been using.

Jin Wang	9	Chinese/Math/Art	20815@students.isb.ac.th	MS	Dani Hansberry	18033@students.isb.ac.th	T	Chinese	
Tai Yi Lao	9	Chinese/Science	19902@students.isb.ac.th	MS			X		
Nicole Villars	11	Spanish/French/Math	20271@students.isb.ac.th	MS			X		
Dong Soo Kang	11	Math/Science	19978@students.isb.ac.th	MS/PB	Uday Badola	19917@students.isb.ac.th	F	Math/SS	
Kankanit Duan Jongrengpian	9	Science	16716@students.isb.ac.th	MS			T		YES
Sirin Nanny Thammakaison	9	Math	16749@students.isb.ac.th	MS	Patrick Andrist	19051@students.isb.ac.th	W	Science	
Amalie Poret	9	English/Math	19414@students.isb.ac.th	MS	Yui Tominaga	19929@students.isb.ac.th	T	English/SS	
Niharika Soni	9	Math/Science	20742@students.isb.ac.th	MS			X		
Szu Hua Paula Wu	9	Math/Science	18946@students.isb.ac.th	MS			W/F		YES
Saksham Singh Birla	9	Math/Science	20655@students.isb.ac.th	MS			M/F		YES
Song Yi Rachel Hyun	11	Math/Bio/Chem	20447@students.isb.ac.th	HS	Jeremy Stuit	19216@students.isb.ac.th	M	Math/General	
Ganghee Alex Son	12	Math	20442@students.isb.ac.th	HS	Vincenzo Tejawinata	20743@students.isb.ac.th	TH	Math	
Josephine Nahoum	12	Spanish	20318@students.isb.ac.th	HS	Oishika Mukherjee	20136@students.isb.ac.th	X	Spanish	
Raaid Raiyan Tanveer	12	Math/Physics/Chemistry	20507@students.isb.ac.th	HS			X		
Brian Yoon	12	Math/Physics	19826@students.isb.ac.th	HS/MS/PB			F		
Ben Homan	11	Math/Science/French	20773@students.isb.ac.th	HS/MS			T, TH		YES
Jeslyn Brouwers	11	English/Math/World Studies	14804@students.isb.ac.th	MS/HS	Seoha Hong	20233@students.isb.ac.th	T	Writing/Reading Science	
Arno Melkonyan	10	Math/Music	20468@students.isb.ac.th	PB/MS			M/TH		YES
Tachpan Wendy Poommarapan	9	Thai/Math/Science	20551@students.isb.ac.th	MS/PB			W/FPB/TH		YES
Rhea Kapoor	11	Physics	20452@students.isb.ac.th	PB			W/F		
Craig Dawe	12	English/Psych/Math	16886@students.isb.ac.th	PB			W		
Man Geun Chun	12	Math/Physics/Chemistry	18812@students.isb.ac.th	PB	Jaehee Jung	21252@students.isb.ac.th	T	Math	
Yeonie Heo	12	Chem/Math	19772@students.isb.ac.th	PB			F		
Penelope Lugo	10	English/Spanish	20746@students.isb.ac.th	PB			X		
Dohwan Kim	12	Bio/English	18250@students.isb.ac.th	PB			X		
Wendy Poommarapan	9	Thai/Math/Science	20551@students.isb.ac.th	MS After School			M		
Kankanit Duan Jongrengpian	9	Science	16716@students.isb.ac.th	MS After School			T		
Ben Homan	11	Math/Science/French	20773@students.isb.ac.th	MS After School			T		
					Nicholas Garrigan	20373@students.isb.ac.th	T	French	
					Grace Homan	20771@students.isb.ac.th	X	French	
					Rocco Abate	20172@students.isb.ac.th	T	Mandarin	
					Sam Moreno (6)	20668@students.isb.ac.th	M/TH	Mandarin	

The Blue box represents all the tutors, the red box indicates the unpaired tutees and the black boxes indicate unavailable tutors and tutees which have already been paired. By using the scanner.nextLine(), each time I looped through the rows in the table, the program encountered either 9, 7, 6, 4 elements, thus I decided to construct if/else statements for the 4 different cases. Tutors were added when there were 9, 7 or 6 tokens and tutees were added when there were 9 or 4 tokens. Below, only the code for readingTutorFile class is explained as the readingTuteeFile has the same algorithm with small adjustments.

```java
public static Tutor[] readingTutorFile(Tutor[] tutors, int tutorCounter) throws FileNotFoundException {

    Scanner scanner = new Scanner(new File("PeerTutoring.csv"));
    String tutorName = "not set yet";
    int grade = -9;
    String subjectsString = "not set yet";
    String tutorID = "not set yet";
    String tutorTypeString = "not set yet";
    String availableDaysString = "not set yet";
    TimeTable[] timeTableArray = {new TimeTable(), new TimeTable(), new TimeTable(), new TimeTable()};


    boolean isMSOnCall = false;
    boolean isCurrentlyAvailable = true;
    while (scanner.hasNextLine()) {
        String line = scanner.nextLine();
        StringTokenizer st = new StringTokenizer(line, ",");
        String[] subjects = new String[17];

        String[] tutorType = new String[4];
        for(int b =0; b< tutorType.length; b++){
            tutorType[b] = "not set yet";
        }
        for(int k =0; k<subjects.length; k++){
            subjects[k] = "not set yet";
        }
        while(st.countTokens() == 9 || st.countTokens() == 6 || st.countTokens() == 7){

            if(st.countTokens() == 9){
                tutorName = st.nextToken();
                grade = Integer.parseInt(st.nextToken());
                subjectsString = st.nextToken();
                tutorID = st.nextToken().substring(0, 5);
                tutorTypeString = st.nextToken();
                st.nextToken();
                st.nextToken();
                availableDaysString = st.nextToken();
                isCurrentlyAvailable = false;

            }
            else if(st.countTokens() == 6){
                tutorName = st.nextToken();
                grade = Integer.parseInt(st.nextToken());
                subjectsString = st.nextToken();
                tutorID = st.nextToken().substring(0, 5);
                tutorTypeString = st.nextToken();
                availableDaysString = st.nextToken();
                isCurrentlyAvailable = true;
            }
            else if(st.countTokens() == 7){
                tutorName = st.nextToken();
                grade = Integer.parseInt(st.nextToken());
                subjectsString = st.nextToken();
                tutorID = st.nextToken().substring(0, 5);
                tutorTypeString = st.nextToken();
                availableDaysString = st.nextToken();
                isMSOnCall = true;
                isCurrentlyAvailable = true;

            }
            else{

            }
            String[] subjectsTemp = subjectsString.split("/");
            String[] tutorTypeTemp = tutorTypeString.split("/");

            for(int k =0; k< subjectsTemp.length; k++){
                subjects[k] = subjectsTemp[k];
            }
            for(int k =0; k< tutorTypeTemp.length; k++){
                tutorType[k] = tutorTypeTemp[k];
            }
            tutors[tutorCounter] = new Tutor(tutorName, tutorID, grad
            tutorCounter++;
            System.out.println(tutorType[0]);
        }

    }

    return tutors;
}
```

Initialization of variables

Use of scanner to split the file by lines
Initialization of local variables

Tokens in the line are counted and only lines with token number 9, 6, 7 are considered as tutors

9 is the case where the tutor and tutees are paired, hence tutor instance is created and tutor is marked unavailable

6, 7 is the case where only a tutor exists and they are available because they do not have a tutee. attributed of tutor is assigned for both if/else blocks although the existence of the isMSonCall token causes the difference in token count

Intermediary attributes are used to convert all variables to the appropriate attribute data type of tutors

Instance of tutor is created using the obtained attributes

3. Pair Tab

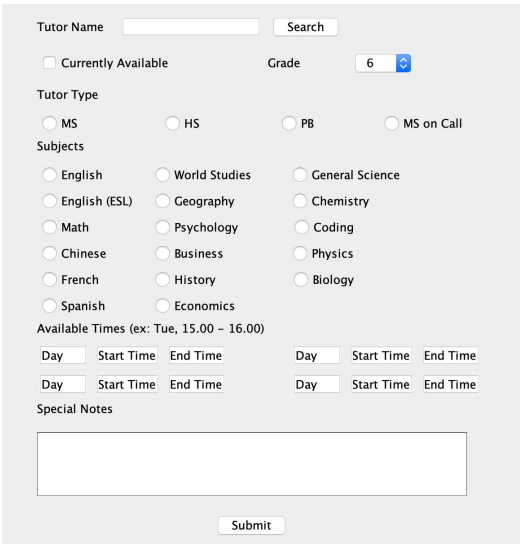
When generating pairs for the tutee input, the tutee is searched and all the attributes of the tutee is loaded to the table. Then, the application loops through subjects of all the tutors and loads available tutors with the same subject as the tutee and these tutors are then displayed to the results table.


```
private void PairTuteeNameSearchButtonMouseReleased(java.awt.event.MouseEvent evt) {
    String searchInput = PairTuteeNameSearchTextField.getText();
    SortAndSearchStudents searchTuteeClass = new SortAndSearchStudents();
    searchedTutee = tutees[searchTuteeClass.TuteeSequentialSearch(tutees, searchInput)];
    System.out.println(searchedTutee.getName());

    ShowTuteeSubjectData(searchedTutee.getSubjects());
    TimeTable timeTableArray[] = searchedTutee.getTimeTable();
    System.out.println("Hi" + searchedTutee.getName());
    PairTuteeSearchResultsTable.setValueAt(searchedTutee.getName(), 0, 0);
    PairTuteeSearchResultsTable.setValueAt(searchedTutee.getGender(), 0, 1);
    PairTuteeSearchResultsTable.setValueAt(searchedTutee.getGrade(), 0, 2);
    PairTuteeSearchResultsTable.setValueAt(searchedTutee.ToStringSubjects(), 0, 3);
    PairTuteeSearchResultsTable.setValueAt(searchedTutee.ToStringTimeTable(), 0, 4);
    PairTuteeSearchResultsTable.setValueAt(searchedTutee.getSpecialNotes(), 0, 5);
    Tutor[] availableTutors = new Tutor[50];
    for(int i = 0; i < availableTutors.length; i++){
        availableTutors[i] = new Tutor();
    }

    int a = 0;
    for(int i = 0; i < tutors.length; i++){
        for(int j = 0; j < tutors[i].getSubjects().length; j++){
            for(int k = 0; k < searchedTutee.getSubjects().length; k++){
                if(tutors[i].getSubjects()[j].equals(searchedTutee.getSubjects()[k]) &&
                  !tutors[i].getSubjects()[j].equals("not set yet") && tutors[i].getIsCurrentlyAvailable()){
                    availableTutors[a] = tutors[i];
                    a++;
                }
            }
        }
    }
    for(int row = 0; row < availableTutors.length; row++){
        if(!availableTutors[row].getName().equals("not set yet")){
            TutorSearchResultsTable.setValueAt(availableTutors[row].getName(), row, 0);
            TutorSearchResultsTable.setValueAt(availableTutors[row].getGender(), row, 1);
            TutorSearchResultsTable.setValueAt(availableTutors[row].getGrade(), row, 2);
            TutorSearchResultsTable.setValueAt(availableTutors[row].ToStringSubjects(), row, 3);
            TutorSearchResultsTable.setValueAt(availableTutors[row].ToStringTimeTable(), row, 4);
            TutorSearchResultsTable.setValueAt(availableTutors[row].getSpecialNotes(), row, 5);
        }
    }

    tutors[0].ToStringSubjects();
```

Word Count: 464

Considerations for User Interface

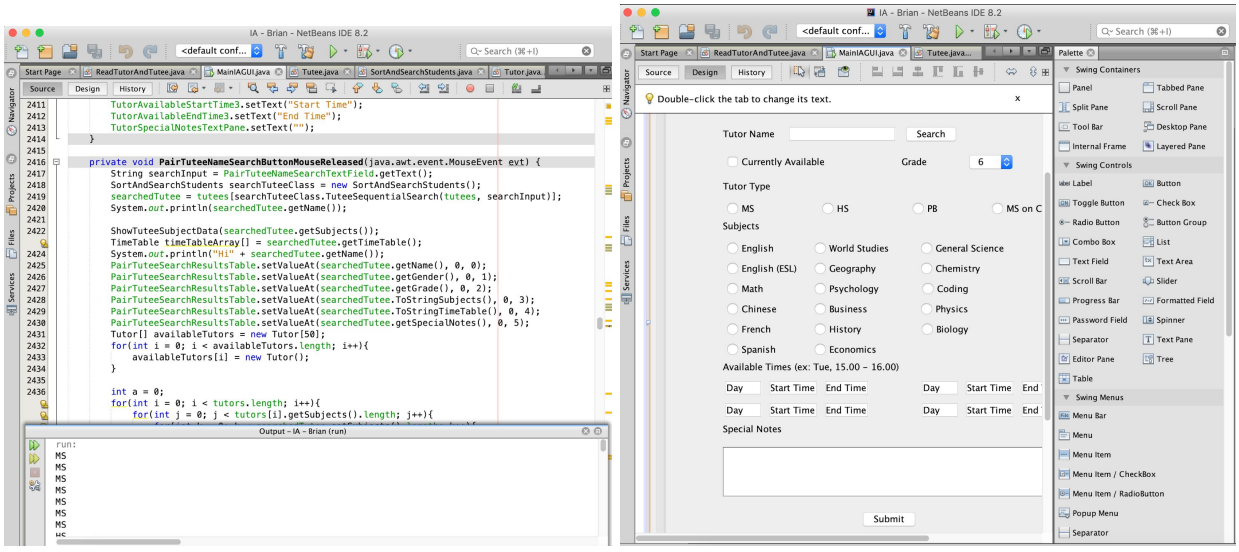
Overall, GUI was used to make the program easy to use for the client. For each textfield, text was either embedded in to make it clear what had to be input and each swing controls were labelled with terminology which the client was familiar with.



Word Count: 45

Software Tool Used

To create the program Netbeans, an IDE for Java was selected due to its code libraries, OOP support and its powerful GUI builder which supports a Swing Application Framework.



Word Count: 29
Criterion C Word Count: 1017