Criterion C → Development

The “Familinage” program integrate all the ToDo from family member, and manage, suggest who need to do what.

Introduction

For my “Familnage” program, I have used a Netbeans software, which is one of most popular software which provides Integrated Development Environment (IDE) for developing the Graphical User Interface (GUI) by providing Java Swing tools with graphical, and convenient access. This attribute enable us to construct and preview how program actually looks like, which allows me to comfortably construct the program with clear vision and idea. Thus, I used the Netbeans.

Summary List of All Techniques

- Static Array
- For loop
- While loop
- Nested loops
- Arrays
- If else statements
- OOP structure
- GUI tabs
- GUI popup menus
- Use of flag value
- Linear searching (technique used)
- Sorting
- Option pane generation for communicating with user
- ArrayList
- Simple and compound selection
- Set and Get
- Parsing values
- Text box
- Combo box
- Swing components
    - Tabs
    - Panel
    - Table

Structure of program

The program has GUI class as a main class of it. In the GUI class, there are two tabs which are for ToDo management and statistics of ToDo. Each of them has own table to show the output. For each table, in order to show several results, two arrays were required. Thus for each arrays, due to Object Oriented Programming (OOP) structure, required two other subclasses. Each subclasses are interacting with the

GUI class. This enables to store the data in array which are necessary for the program. The data itself is inputted via GUI components in the GUI class and stored/compiled in the subclasses, and subclasses returns back the stored data whenever the GUI class ordered and those data displayed through the table or text boxes.

Reflecting back to the main purpose, ultimately program tries to show the ToDo list within the family scale with several convenient functions and output the recommendation of task for specific person and vise versa. In order to complete the followed series of actions, amount of data should be traded and passing by. Initial thought was to develop the system within one main GUI class. However, there were several difficulties and problems related to the data handling, such as lack of place for storing/holding the data, confuseness of location of data. Thus, by creating other classes which connect with main GUI class in OOP structure, it enabled to clarify the location of data, and increased the efficiency of data passing between the system.

Data structure used

- Arrays
- Arrays of Object

Each arrays has own constructor in a different class, due to the OOP structure. There are mainly two types of arrays. One of them is the main array. This array is only used for *Task* array. This consist main data from user inputs, and function as a database. In order to have interchangeability, String was a best datatype for parsing, so it consist only String. Second of them is the subarray. In other word, this array consist the data required during specific process. In this program, *Stats* and *Rank array* are classified to subarray. Both of them are used for statistics, which requires two times of sorting with different pair of data set, String and int.

Each arrays are static arrays and not dynamic such as ArrayList. The reason why is because the client has definite limit to manage the tasks. This is more of humanity object, but because user's life is always indefinite, there are no ToDos after long time such as a year. Also, ToDos are erased after it is done, so it does not require a lot of container to show ToDo list. Static array enables faster access to data itself, and not require amount of memory compared to other dynamic data structure. Therefore, we concluded to have static array as a better option.

Main unique algorithm

Statistic system for recommending task to specific member

This algorithm does not use mathematical methods but takes logical method to find out either "person" or "tasks" to recommend. In this part, it finds out "person" to do the selected task. It includes array, linear search, sort, for/while/nested loops, and if statement.

- Take in the combo box component do linear search for *Task array*.
- Result of linear search stored into *stat array*, and sort *stat array*.
- Repeat following operation until it reaches to the limit.
    - Get name from combo box and linear search with the *stat array*. As the search matches, *countPerson* counter increase.
    - Both name from combo box and *countPerson* number stored into *subStat array*.

```
for(int i = 0; i < stat.length; i++){
    rankT.setValueAt(" ", i, 0);
    rankT.setValueAt(" ", i, 1);
}

int counting = 0;

for(int i = 0; i < taskArray.length; i++){
    if(ctaskCB.getSelectedItem().equals(taskArray[i].getTask())){
        stat[counting].setName(taskArray[i].getPersonName());
        counting++;
    }
}

StatSort sort = new StatSort();
sort.sortForStat(stat);

int countPerson = 1;
int componentNum = 1;
int countRank = 0;

while(componentNum < cnameCB.getItemCount()){
    for(int j = 1; j < stat.length; j++){
        if(cnameCB.getItemAt(componentNum).equals(stat[j].getName())){
            countPerson++;
        }
    }
    subStat[countRank] = new Rank(cnameCB.getItemAt(componentNum), countPerson);
    componentNum++;
    countPerson = 0;
    countRank++;
}
```

- After operation completed, the *subStat array* sorted.
- Show result in the table. First column shows ranking number, and second column shows the person name.
- As it finish, the recommendation sentence; **“Task” should be done by “1st ranked person”** in *recomTF*.

```
StatSort secondSort = new StatSort();
secondSort.sortRank(subStat, cnameCB.getItemCount());

int tableLength = 1;
for(int k = 0; k < cnameCB.getItemCount() - 1; k++){
    rankT.setValueAt(tableLength, k, 0);
    rankT.setValueAt(subStat[k].getRankObj(), k, 1);
    tableLength++;
}
//System.out.println(subStat[0].getRankObj());
recomTF.setText(ctaskCB.getSelectedItem() + " should be done by " + subStat[0].getRankObj());
```

By using two arrays, it divided up the extraction process and numbering process. This division of work avoided the confuseness and efficiently process the data in correct way.

User Interface / GUI work

The main GUI is consisted by several types of GUI components which is called java swing components. Especially combo box owns highest proportion in this system. For example, name, task, and dates are inputted into table (or *Task array*) via combo box. Also, components in combo box are inputted beforehand by user via text field in popup. Additionally, popups used for main information input to save area of pane and avoid confuse.

The basic reason for using combo box at most is it avoids repeated works. In the example above, user should input every information by typing everytime. Due to the characteristic of ToDos which frequently repeats the thing, that operation is painful for users and result with low user friendliness. Also, it works as simple list to store the data, so it can be used to recall data in some operation such as statistics part. Thus, the frequently used data in system is compiled by combo box.

Main reason for using GUI is that provides high accessibility and good visualization for user. Especially ToDos should be listed, the best option to show and interact with it was to use GUI.

Word count: 827