Criterion C Development

The “Rx” program I developed is aimed to be used by doctors and patients to keep track of medicine prescription.

Software Tool Used

The software that I used to develop this program is Netbeans. Netbeans is one of the most popular Integrated Development Environment (IDE) used by programming professionals. The program provided me a working environment where I can see the output and also preview the GUI application and it support object oriented programming. Since my program will be a GUI program, Netbeans is the right choice for developing my program.

Techniques Used

- Static Array
- Constructor
- For loops
- Nested loops
- If else statements
- Switch case
- Mathematical operations
- Linear searching
- Error handling
- GUI tabs
- GUI popup menus
- Text boxe
- Combo box
- Parsing values
- Set and get
- Swing controls
- Swing Containers
    - Panels
    - tabs

Structure of the Program

The program have an GUI as its main class where the GUI is separated into three pages: one for medicine schedule, one of patient search and data, and one for medicine data and prescription. Therefore, I have in total of four arrays, which caused me to have 4 extra classes for the constructors of those arrays. The main GUI and the other four constructor classes interact with each other to be able to create arrays that can store data needed to be used in the program. One constructor class will be connect to one array used in the main GUI. The data can be added from the GUI to the array and information from the array can be displayed and utilized in the GUI. Data added on one tab will be displayed on other tabs' element through accessing the arrays.

At first, I decided to make each tabs to be have its own GUI, however, I thought that this would cause confusions in users and also increase the risk of bugs since more values and data will be passing between GUI. In my new design, I minimized and fit everything into one GUI. This is when I utilized the panel tabs since it gives the sense of separation between tabs like if I was to without having to create more GUI classes. This will be more efficient since less classes are there to be passing values between each other are and it would be less complicated for the users since they can access all needed components in one GUI. This would resulted in less bug since values can be mixed up when passing between classes.

Data Structures used

- Arrays
- File
- Array of objects

Each array has its own constructor in a different class. Each array will have two datatype, String and Double. The main arrays are *medicineList*, *patientList* and *rxList*. *medicineList* array consists of two attributes, the name (String) and the dose (Double). *patientList* array consists of four String attributes, suffix, first name, last name and the disease. *rxList* array consists of three attributes, patient name (String), time (String) and dosage (double).

I chose to use normal static array instead of linked list in my program. The reason is that the capacity of the clinic is definite since they will not expand the size of the clinic. Therefore, knowing a definite size allows me to construct a more efficient data structure. Static array is faster to access and uses less memory compared to linked list. Since I know the size, static array would be more suitable because linked list size is dynamic.

Unique Algorithm

Calculating total dosage per day

This code is to calculate the dosage per day for each patient for each medicine in the prescription. The algorithm includes array, linear search, loops, if statements, and arithmetic operations.

- initialized variables essential for the class. See the picture below.
- The first loop runs through the *rxList* array and sorted out whether that array position matches the patient selected.
- If condition is true, it proceed to another loop in *medicineList* to check if the medicine listed in the *rxList* exists. If true, it proceeds. See image below.

```
private void totalDoseBTNActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //fix this thing
    String patientToShow = "";
    totalDoseTA.setText("");
    patientToShow = (String) patientNameCB.getSelectedItem();
    double currentDose = 0.0;
    double totalDose = 0.0;
    double doseToAdd = 0.0;
    String medicine = "";
    System.out.println("Start calculating dose");

    for (int i = 0; i < RxCounter; i++) {
        if (RxList[i].getSelectedPatient().equalsIgnoreCase(patientToShow)) {
            for (int j = 0; j < medCounter; j++) {
                if (RxList[i].getSelectedMed().equalsIgnoreCase(medicineList[j].getmedName())) {
                    System.out.println("found medicine in database");
```

- Next loop runs through the *doseTotalList* array that stores the total dosage of each medicine for each person.
- The two if statements in the loop determines whether the medicine already exists in the array and exists under that patient's name. This allows the program to know whether to create new position in array or update an existing position.
    - If it already exists, then the newly calculated dose will be added to current dose which is obtained by *get*. Then the new value will be *set* as the new total dosage.
    - If it does not exist, then it will establish new index in the array. See image below.

```
for (int k = 0; k < TDCounter; k++) {
    if (doseTotalList[k].getMedicine().equalsIgnoreCase(RxList[i].getSelectedMed())) {
        //already exist   "add on to existing value in the array"
        System.out.println("medicine already added -> update value");
        currentDose = doseTotalList[k].getTotalDose();
        doseToAdd = RxList[i].getSelectedDose() * medicineList[j].getmedDose();
        totalDose = currentDose + doseToAdd;
        System.out.println("new total= "+totalDose);
        doseTotalList[k].setTotalDose(totalDose);

        System.out.println("new dose total for " + doseTotalList[k].getMedicine() + " : "
                + doseTotalList[k].getTotalDose());


    }
    if (TDCounter ==  0 || !doseTotalList[k].getMedicine().equalsIgnoreCase(RxList[i].getSelectedMed())
            ) {

        //new medicine  "add new value and medicine into the array"
        System.out.println("not added yet -> create new index");
        currentDose = RxList[i].getSelectedDose() * medicineList[j].getmedDose();
        System.out.println("Med name: " + RxList[i].getSelectedMed());
        System.out.println("Dose: " + RxList[i].getSelectedDose() + "*" + medicineList[j].getmedDose());
        doseTotalList[TDCounter] = new doseTotalData(patientToShow, medicine, currentDose);
        TDCounter++;
        System.out.println("TD counter = "+TDCounter);
    }
    else{
        System.out.println("Does not match any condition");
    }
```

Adding to schedule

A code to determine, from the prescription, where each medicine goes to. This will consider which time block and which patient. The algorithm includes array search, linear search, loops, and switch case.

- I started the algorithm by initializing more variables just for this class and reset the text areas, see the picture below

```
//  Change to TA
String patientToShow  = "";
String medicineToShow = "";
String timeToShow     = "";
Double doseToShow     = 0.0;

   patientToShow = (String) patientNameCB.getSelectedItem();

   //reset
   t5amTA.setText("5am");
   t10amTA.setText("10am");
   t11amTA.setText("11am");
   t12pmTA.setText("12pm");
   t1pmTA.setText("1pm");
   t2pmTA.setText("2pm");
   t3pmTA.setText("3pm");
   t4pmTA.setText("4pm");
   t5amTA.setText("5am");
   t5pmTA.setText("5pm");
   t6amTA.setText("6am");
   t6pmTA.setText("6pm");
   t7amTA.setText("7am");
   t7pmTA.setText("7pm");
   t8amTA.setText("8am");
   t8pmTA.setText("8pm");
   t9amTA.setText("9am");
```

```
   for(int i=0; i<RxCounter; i++){
if  ( RxList[i].getSelectedPatient().equalsIgnoreCase(patientToShow)){


   System.out.println("Patient Found");
   System.out.println("medicine: "+RxList[i].getSelectedMed());
   System.out.println("    Dose: "+RxList[i].getSelectedDose());
   System.out.println("    Time: "+RxList[i].getSelectedTime());


   medicineToShow = RxList[i].getSelectedMed();
   timeToShow     = RxList[i].getSelectedTime();
   doseToShow     = RxList[i].getSelectedDose();
```

- After that, I utilized linear searching where it is a for loop through the RxList array and find a position where *getPatientName* matches the *patientToShow*.
- When it is found, the datas in that array position will be set to the initialized variables.
- Then, the switch case is implemented with the condition being the time. By considering the time, a string like 5am, the switch case will determine which block it will go to, see the image below.

```
switch(timeToShow){

    case("5am"):
          System.out.println(medicineToShow+" "+doseToShow+" added to 5am block");
          t5amTA.setText(t5amTA.getText()+"\n"+medicineToShow+" "+doseToShow);
          break;

    case("6am"):
           System.out.println(medicineToShow+" "+doseToShow+" added to 6am block");
           t6amTA.setText(t6amTA.getText()+"\n"+medicineToShow+" "+doseToShow);
           break;

    case("7am"):
           System.out.println(medicineToShow+" "+doseToShow+" added to 7am block");
           t7amTA.setText(t7amTA.getText()+"\n"+medicineToShow+" "+doseToShow);
           break;

    case("8am"):
           System.out.println(medicineToShow+" "+doseToShow+" added to 8am block");
            t8amTA.setText(t8amTA.getText()+"\n"+medicineToShow+" "+doseToShow);
           break;

    case("9am"):
           System.out.println(medicineToShow+" "+doseToShow+" added to 9am block");
           t9amTA.setText(t9amTA.getText()+"\n"+medicineToShow+" "+doseToShow);
           break;

    case("10am"):
           System.out.println(medicineToShow+" "+doseToShow+" added to 10am block");
           t10amTA.setText(t10amTA.getText()+"\n"+medicineToShow+" "+doseToShow);
           break;
```

User Interface / GUI work

My main class is a GUI where clients will be on it. The application interface consists of 3 tabs where different tabs have different Java Swing Components to suit each of its purposes. The patient's' schedule tab have text area for each time block from 5 AM to 8 PM, a combo box that allows the client to select a patient name to display that person's prescription, a text area that shows the total dosage, and all action are performed after clicking the button designated. Combo box limited the input which means that there will be less chance of input errors.

The patient tab have text fields for inputting the patient's suffix, first name, last name, disease, and also a table to display list of patients. The medicine tab have text field to input the medicine brand and the dosage (mg), a text fields to select prescription information (medicine, time, and dose) to a selected patient, a text area and table to display recent prescriptions and list of medicines added to the database.

I decided to implement GUI because it would be easier for the clients to use the application. A schedule, for example, would be hard to render if it's only display on the system's output. Java Swing components allow me to create a more interactive and simplistic application for the clients.