Criterion C: Development

Word Count: 614

Introduction

I programmed an application which is utilized to input, search for and organize information about a user's miscellaneous tasks as an individual who works from home in collaboration with multiple business partners. The Netbeans IDE and a Java OOP approach is used to make a GUI interface for the user to most easily work with. The program works with a linked list as the primary data structure, allowing the user to input and remove data dynamically.

Summary of Programming Techniques

List of Techniques:

- Flag Values (Such as "Not set yet")
- Nested Loops
- Encapsulation
- Multi Conditional if/else statements
- User defined objects made from an OOP "Task" template class
- Sorting - bubble sort to sort particular data based on different attributes
- GUI tabs
- Searching: Linear/sequential search
- Default table Model

Data Structure Used:

- Linked List

Summary of 'Task' Class

In this class, initial variables were set, user objects were made, and encapsulation - .get methods specifically - allowed the extendability of these variables into the Main GUI class.

1. Flag Values

```
private String taskName = "Not set yet";
private String taskLocation = "Not set yet";
private String taskUrgency = "Not set yet";
private String contactedGroupName = "Not set yet";
private String daysRemaining = "Not set yet";
private String prepTime = "Not set yet";
```

2. User Defined Objects from a Template Class; e.g. "Task"
3. Parameter Passing - local variables within the program accessed in sub-programmes without the need for global variables.
4. Default Constructors

```
public Task(String taskName, String taskLocation, String taskUrgency, String contactedGroupName, String daysRemaining, String prepTime){
    this.taskName = taskName;
    this.taskLocation = taskLocation;
    this.taskUrgency = taskUrgency;
    this.contactedGroupName = contactedGroupName;
    this.daysRemaining = daysRemaining;
    this.prepTime = prepTime;
}
```

5. Encapsulation: Using the accessor 'get' methods to retrieve attributes of private variables.

```
public String getTaskName(){
    return taskName;
}
```

'SortAndSearch' Class To Re-Organize Tasks

In this class, I created a bubble sort for daysRemaining, taskUrgency and prepTime in order to allow the user to customize the display of their inputted data in the table.

6: Bubble Sort (Used similarly for daysRemaining and prepTime):

- i: int n acts as an initial size of the loop which will reduce through n--.
- ii: 'boolean sorted = false;' assumes that the tasks are not sorted, and acts as the condition for the sort to continue - 'while(!sorted)' → while not sorted.
- iii: for loop to iterate through the list for a designated number of iterations

```
*/
public class SortAndSearch {
    //DO in action performed
    public void sortByUrgency(LinkedList<Task> tasksList){
        int n = tasksList.size();
        boolean sorted = false;
        while(!sorted){
        n--;
        sorted = true;
        for(int i = 0; i < n; i++){
                if(tasksList.get(i).getTaskUrgency().charAt(0) < tasksList.get(i+1).getTaskUrgency().charAt(0)){
                    //swaps the 2 elements if the first letter is less than the second letter
                     Task temp = tasksList.get(i);
                     tasksList.set(i, tasksList.get(i+1));
                     tasksList.set(i+1, temp);
                     sorted = false;
           }
        }
      }
    }
```

Summary of GUI/Main class programming

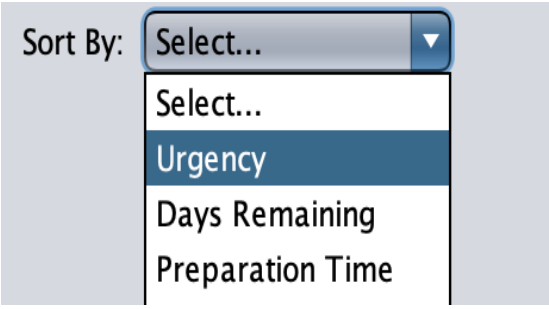
7: Linear/Sequential Search: Method for finding a key element (NameForSearchTF.getText()), within a list, checking each element sequentially for a match.

- i: For loop used to iterate through the list for a designated number of iterations
- ii: if statement used to provide a condition in which the search is complete.

```
//Linear search used to set assigned values for each variable in accordance to it's task name.
for(int i = 0; i < tasksList.size(); i++){
    //for loop used to loop through the tasksList - tasksList.size tells the computer to loop through as many times as the
    //list has elements.
    if(tasksList.get(i).getTaskName().equals(NameForSearchTF.getText())){
        //if the name retrieved from the tasksList is the same as the name in the 'NameForSearchTF'...
        LocationForSearchTF.setText(tasksList.get(i).getTaskLocation());
        ContactForSearchTF.setText(tasksList.get(i).getContactName());
        UrgencyForSearchTF.setText(tasksList.get(i).getTaskUrgency());
        DaysRemainingForSearchTF.setText(tasksList.get(i).getDaysRemaining() + "");
        PrepTimeForSearchTF.setText(tasksList.get(i).getPrepTime() + "");
        //each of these elements in the search menu will be assigned through .setText, to be the same as their inputted element.
        //for instance, locationForSearchTF - the location to search for, will be assigned the same location as inputted.
    }
}
```

8: Multi-conditional if/else statements: Use of 'if' and 'else if'

- Jdropdown menu has different actions performed as the different options were selected.




```
if(selectedIndex == 1){
    //calling upon the urgency sort made in the SortAndSearch class
    SortAndSearch urgencySort = new SortAndSearch();
    urgencySort.sortByUrgency(tasksList);
    for(int row = 0; row < tasksList.size(); row++){
        DataTableTF.setValueAt(tasksList.get(row).getTaskName(), row, 0);
        DataTableTF.setValueAt(tasksList.get(row).getTaskLocation(), row, 1);
        DataTableTF.setValueAt(tasksList.get(row).getContactName(), row, 2);
        DataTableTF.setValueAt(tasksList.get(row).getTaskUrgency(), row, 3);
        DataTableTF.setValueAt(tasksList.get(row).getDaysRemaining(), row, 4);
        DataTableTF.setValueAt(tasksList.get(row).getPrepTime(), row, 5);
    }
    JOptionPane.showMessageDialog(this,"Tasks Organized By Urgency");
}else if(selectedIndex == 2){
    SortAndSearch daysRemainingSort = new SortAndSearch();
    daysRemainingSort.sortByDaysRemaining(tasksList);
    for(int row = 0; row < tasksList.size(); row++){
        DataTableTF.setValueAt(tasksList.get(row).getTaskName(), row, 0);
        DataTableTF.setValueAt(tasksList.get(row).getTaskLocation(), row, 1);
        DataTableTF.setValueAt(tasksList.get(row).getContactName(), row, 2);
        DataTableTF.setValueAt(tasksList.get(row).getTaskUrgency(), row, 3);
        DataTableTF.setValueAt(tasksList.get(row).getDaysRemaining(), row, 4);
        DataTableTF.setValueAt(tasksList.get(row).getPrepTime(), row, 5);
    }
    JOptionPane.showMessageDialog(this,"Tasks Organized By Days Remaining");
}else if(selectedIndex == 3){
    SortAndSearch prepTimeSort = new SortAndSearch();
    prepTimeSort.sortByPrepTime(tasksList);
    for(int row = 0; row < tasksList.size(); row++){
        DataTableTF.setValueAt(tasksList.get(row).getTaskName(), row, 0);
        DataTableTF.setValueAt(tasksList.get(row).getTaskLocation(), row, 1);
        DataTableTF.setValueAt(tasksList.get(row).getContactName(), row, 2);
        DataTableTF.setValueAt(tasksList.get(row).getTaskUrgency(), row, 3);
        DataTableTF.setValueAt(tasksList.get(row).getDaysRemaining(), row, 4);
        DataTableTF.setValueAt(tasksList.get(row).getPrepTime(), row, 5);
    }
    JOptionPane.showMessageDialog(this,"Tasks Organized By Urgency");
```

In this code, the multi-conditional if/else if statements check whether the option selected form the JDropdown menu was 1/'Urgency', 2/'Days Remaining' or 3/'Preparation Time', in order to determine which attribute was the one to be sorted.

Upon which, the given sort - urgencySort, daysRemainingSort or prepTimeSort is done through the MainGUI_ClientDB class **using a** SortAndSearch called from the SortAndSearch class.

9: Use of a Default Table Model: an implementation of TableModel that allows actions such as removing elements to be much easier done.

- Commands such as removeRow.

```
DefaultTableModel model = (DefaultTableModel) this.DataTableTF.getModel();
int[] rows = DataTableTF.getSelectedRows();
for(int i=0;i<rows.length;i++){
    model.removeRow(rows[i]-i);
}
//removes the task from the original tasksList by adding it to a new empty list.
tasksList = new LinkedList<Task>();
JOptionPane.showMessageDialog(this,"Removed Task Successfully");
```

- For loop iterating rows.length times.
- model.removeRow simply removes the row(s) selected.

"How Do You Remove Selected Rows from a JTable?" *Stack Overflow*, stackoverflow.com/questions/655325/how-do-you-remove-selected-rows-from-a-jtable.

10: Nested loops: used when working with two-dimensional tables holding columns and rows, to iterate through the data in the table's rows and columns.

- Clears table by setting each index's text field to "" - empty String.

```
for (int i = 0; i < DataTableTF.getRowCount(); i++) {
    for(int j = 0; j < DataTableTF.getColumnCount(); j++) {
        DataTableTF.setValueAt("", i, j);
    }
}
```

"Java - How to Clear Contents of a JTable ?" *Stack Overflow*, stackoverflow.com/questions/3879610/how-to-clear-contents-of-a-jtable. Accessed 8 Apr. 2022.

11: Initialization of a new task using encapsulated accessor methods '.get'

```
Task task = new Task(TaskNameTF.getText(), TaskLocationTF.getText(), UrgencyTF.getText(),
ContactGroupTF.getText(), DaysRemainingTF.getText(),  TimeToWorkTF.getText());
```

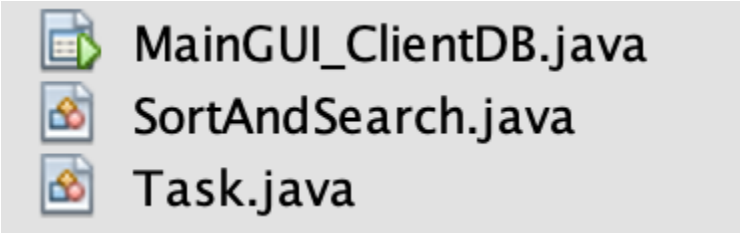
The .getText() call simply retrieves the text inputted by the user in the 6 text fields on the 'Enter Tasks' page (See 'User Interface/GUI Work'), and adds them to the Linked List as a new task.

Data Structure Used:

I used a Linked List in this program due to the fact that data in this program must be able to added and removed dynamically, thus prompting the use of a dynamic data structure such as a Linked List. This allows the user full flexibility in terms of adding or removing data to their preference. Furthermore, the speed of sorting and searching is not pivotal to the user, since this program is mainly focused on keeping track of tasks, thus lessening the extent to which the Linked List is disadvantageous in comparison to an Array List for example. The Linked List is privately declared as it is only worked with in the MainGUI class, and therefore making it public would be redundant.

```
private LinkedList<Task> tasksList = new LinkedList<Task>();
```

Structure Of The Program



This program uses 3 classes. The MainGUI_ClientDB class is the central feature of the program, using OOP. It is dependent on and thus uses/collaborates with the other classes 'Task' and 'SortAndSearch'. It also contains Java Swing tools allowing the design interface to function properly.

OOP Elements

Relationships:

- Has a (aggregation)
- Uses a (dependency)
- MainGUI_ClientDB has a Task
- MainGUI_ClientDB uses a SortAndSearch

MainGUI_ClientDB **uses a** SortAndSearch method

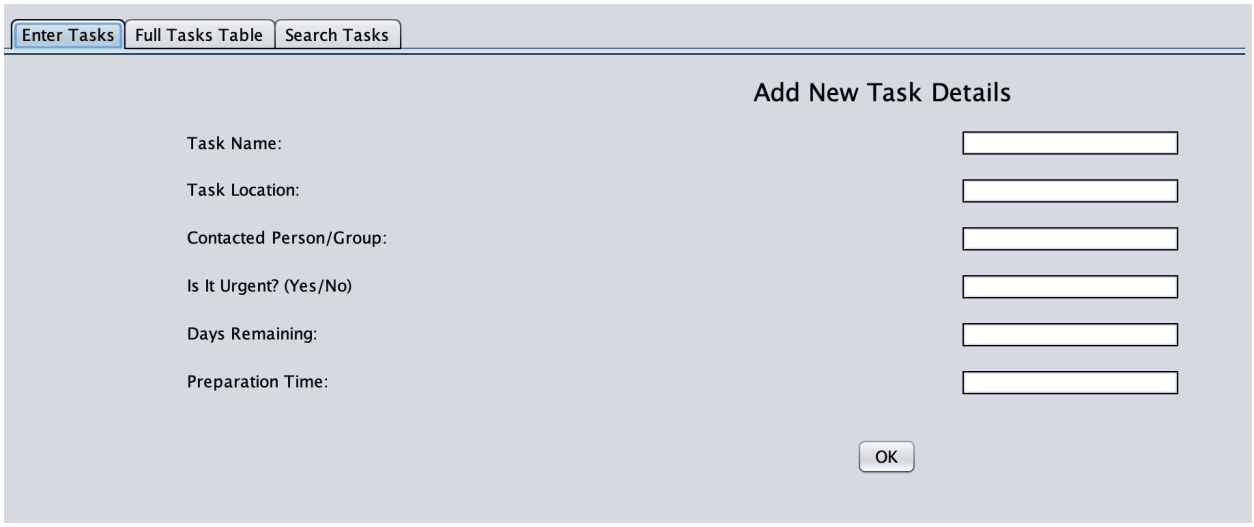
```
SortAndSearch urgencySort = new SortAndSearch();
```

MainGUI_ClientCB **has a** Task

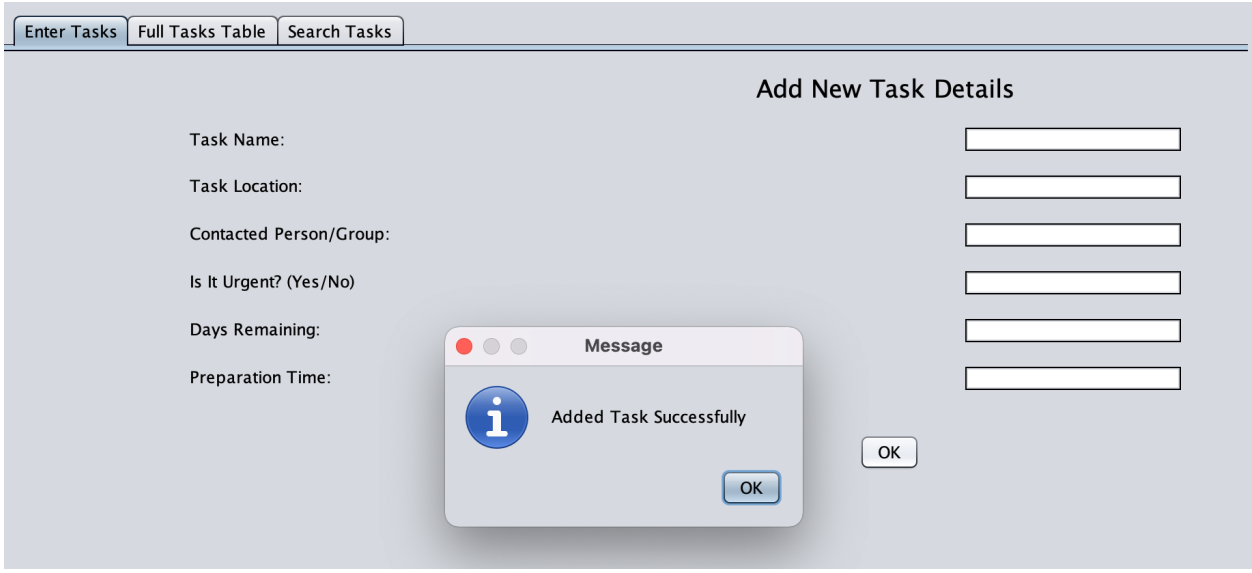
```
Task task = new Task(TaskNameTF.getText(), TaskLocationTF.getText(), UrgencyTF.getText(),
ContactGroupTF.getText(), DaysRemainingTF.getText(),  TimeToWorkTF.getText());
```

The overall use of OOP allows the program to be more easily debugged, reused, managed, and extended, as the classes and their relationships occur in a logical manner.

User Interface/GUI Work

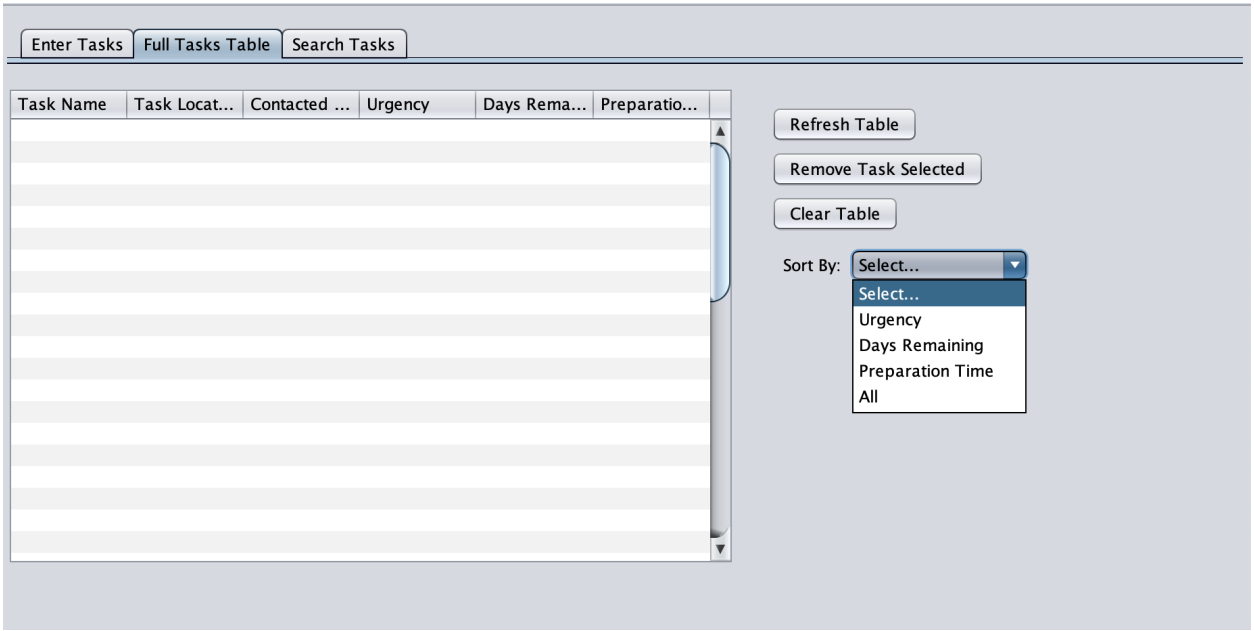


Common features of the "Enter Tasks" page include textfield and buttons. Textfield is used to input necessary information for the user to later work with. "OK" button allows the user to input data, which can later be searched for in "Search Tasks" or displayed categorically in "Full Tasks Table".

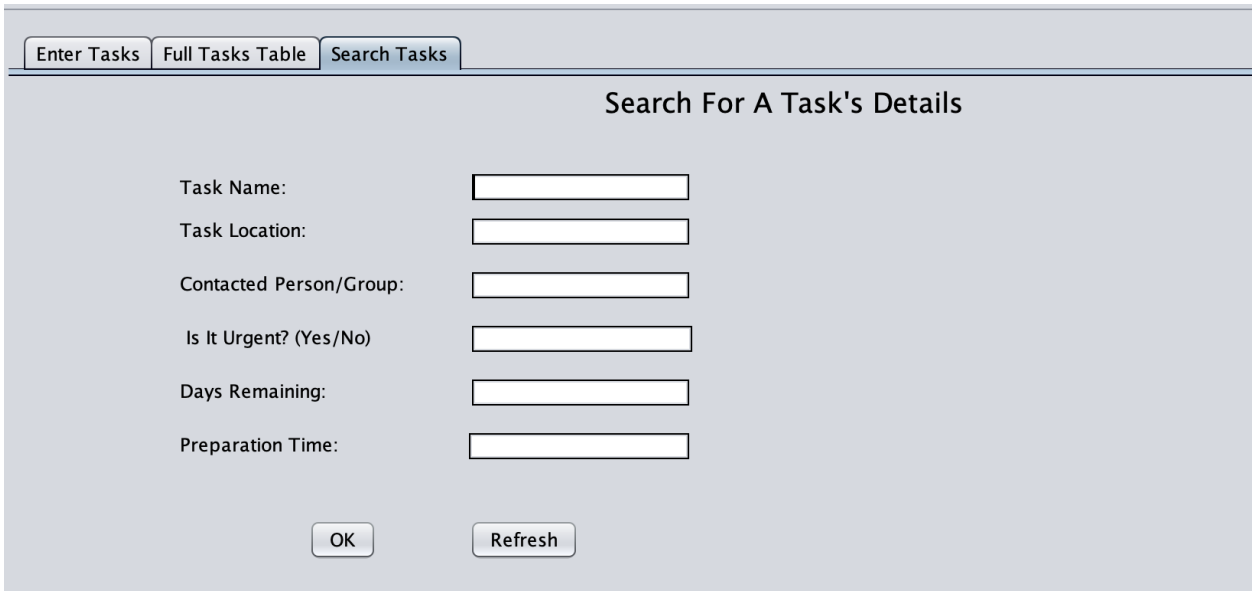


Simultaneously, the "OK" button refreshes "Enter Tasks" page, displaying a confirmation that the task has been successfully added to the program. This is done through the Joptionpane code.

```
JOptionPane.showMessageDialog(this,"Added Task Successfully");
```



"Full Tasks Table" page presents all the information needed by the client in a JTable, where the user can access inputted data through the "Refresh Table" button. "Clear Table" button allows all data to be removed at once to the client's liking, whereas the "Remove Task Selected" allows them to remove individual tasks if they would rather do that. The ComboBox "Sort By:" allows the user to customize how they want to organize the presentation on their tasks - by urgency, days remaining or preparation time.



"Search Tasks" page allows the user to input their task name in order to view it's associated attributes, called upon with the "OK" button.