COMMAND LINE TO DO LIST WITH JAVA

OBJECT ORIENTED PROGRAMMING (COMP6175)

1. VALERIUS OWEN (2301868371)

2. FERNANDO ASIKIN (2301870142)

3. CHANDRA WIJAYA (2301872596)

4. KELVIN PRASETIO (2301870703)

5. EVAN CHRISTIAN GUNARTO (2301871095)

6. NADHIF NAKULA BARCELI (2301885705)

FAKULTAS TEKNIK INFORMATIKA

UNIVERSITAS BINA NUSANTARA

BINUSIAN 2023

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

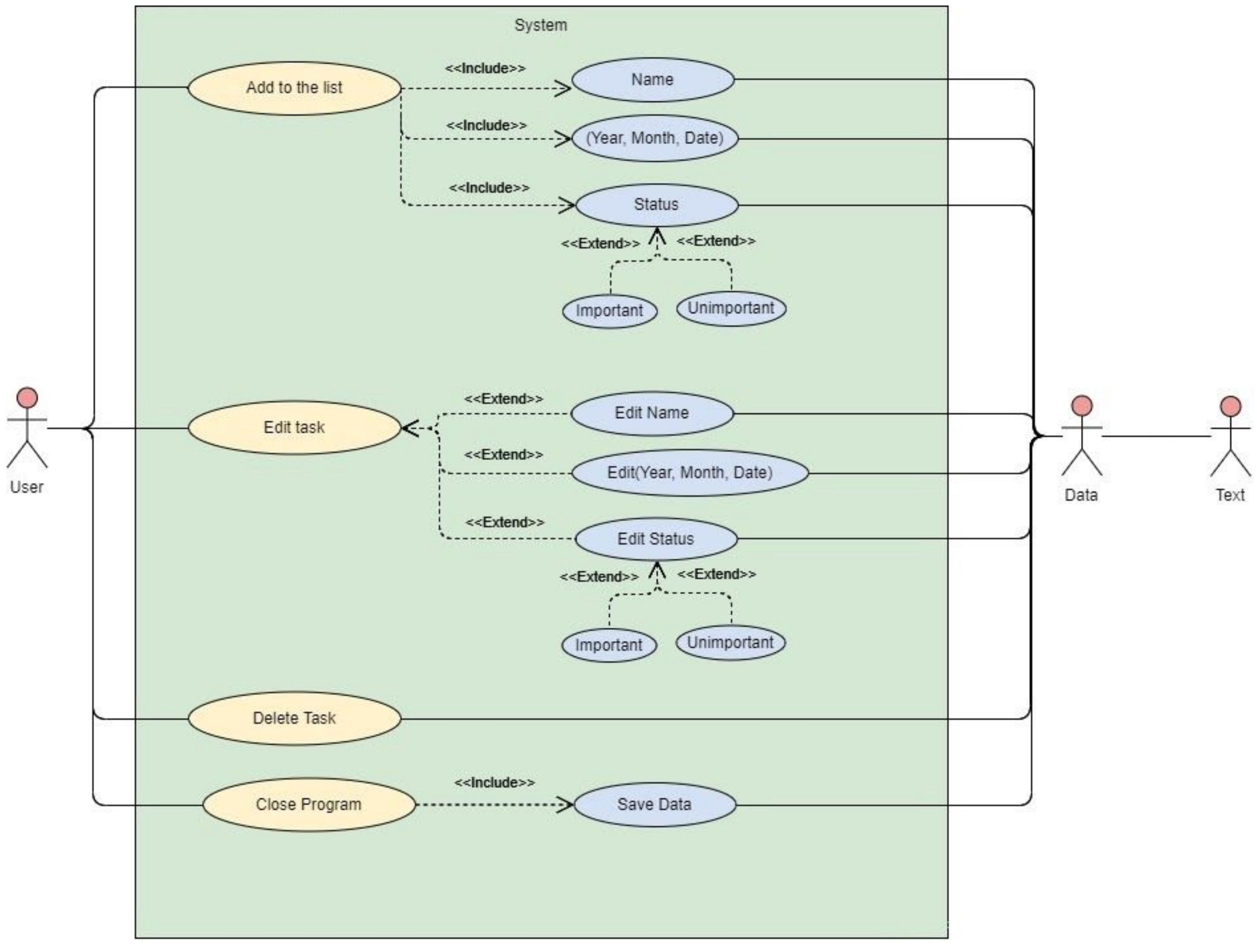
For the class final project, we have to make a complete java application that utilizes multithreading, abstract class, and interface. With these requirements, our group has discussed to program a to-do list that uses a command-line interface. We create the program using a command-line interface because it is a lot lighter in terms of memory and CPU utilization and can perform faster than using a Graphical User Interface. We decided on a to-do list program since many people, in this quarantine, where they are forced to work from home would have problems organizing and finishing their tasks on time. A to-do list program is very important to list and organize our tasks where we can easily review the list and prioritize the most important tasks to be on top to finish first. Organizing the tasks with a list can make everything much more manageable and neat. With the To-do list app, we hope that we can help people to organize their tasks, whether it may be their work or house chores to be finished on time with less pressure and create a new, more productive lifestyle.

This app enables the user to be able to list their tasks and sort it based on importance and due dates, where the task that has a closer due date will be prioritized first but they will be displayed below tasks that are labeled as important tasks. There will also be an option to edit the due date, task name, the importance, and another option to delete the tasks that are rendered useless, obsolete, or completed.

CHAPTER 2

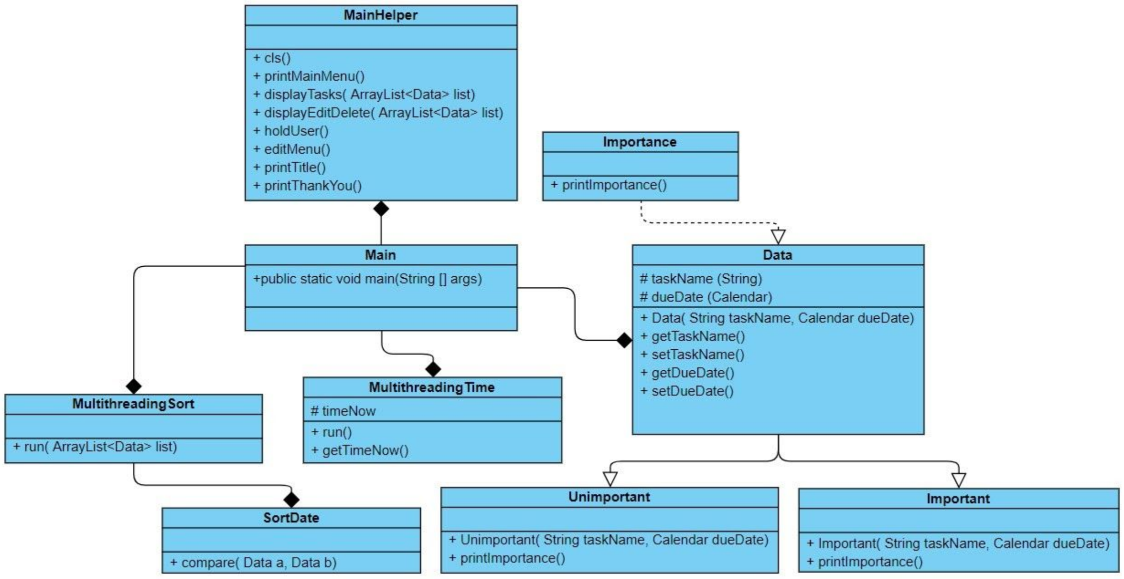
ANALYSIS AND DESIGN

Use Case Diagram



From the use case diagram above, there are 3 actors who contribute to the program, user, data, and text. Users have access to 4 functions in the program; add to the list, edit task, delete task, and close the program. In the function add to the list, the user can add a task name, year, month, and date and also can set the status of the task to important or unimportant. After that, it will be saved to the data and sorted based on the importance of the task. Next in the edit menu, the program will output the name, due date, and importance of the task and the user can then edit the name of the task in the edit name function. Users can also edit the due date of the task in the edit due date function. Finally, the user can edit the status of the task from important to unimportant or vice versa in the edit importance function. Next in Delete from list, users can choose to delete tasks that are already saved in the data. The last one, the user can close the program and the task data will be saved to the data then to the text.

Class Diagram with Class Relationships



From the class relationship above, there are 9 classes which consist of Main, MainHelper, MultithreadingSort, MultithreadingTime, Data, SortDate, SortImportance, Unimportant, Important and an interface which is named Importance that each has its own relationships to one another. The main class has a composition relationship with Mainhelper, MultithreadingSort, MultithreadingTime, and Data because Main depends on those classes in order for the program to work properly. And then there are also some other relationships like inheritance that is owned by Data to Important and data to Unimportant. The Unimportant and Important class inherits attributes and methods from the Data class. There is also a realization relationship between Importance class, which is an interface, and Data class because Data class is implementing Importance class as an interface.

CHAPTER 3

IMPLEMENTATION

BASIC EXPLANATION

The program that our group created is a program to regulate what people have to do. Basically, the program that we made will sort what people have to do from the closest order of time. But if there is a priority sign given, then the program will make the order in the upper order and also sorted according to the closest time sequence.

In our program, when users enter our program, they can see the status of whether files are stored or not. If it is saved it will say 'Save file exists'. However, if it does not exist, there will be no words 'Save file exists'. After the writing, our program will display the logo of our program and after that will display today's date and what tasks the user has saved in this app. If you don't have it then there will be the words "You don't have anything to do". Whereas if there are already then the order of tasks will be displayed. If the order in which the tasks are displayed is an important task, there will be exclamation marks in front of and behind the task. Meanwhile, if the task is not an important task then there will be no exclamation mark in front of or behind the task.

The Task itself is stored and categorized as a SubClass of the SuperClass Data. The Date SuperClass implements an interface of Importance. The data of the tasks will be all stored in the SubClasses of Data. The Data Class itself has 2 of the variables needed to store the task name and task dueDate, 4 methods (2 sets of setter and getter for the 2 variables) and a constructor. The difference between the two child/sub classes are in their class name which is used to determine some actions in the main menu and the interface method of printImportance(). The difference for these 2 sub classes are to differentiate the Important and Unimportant tasks to be listed in the arraylist in the Main class.

After all the sequence of tasks is displayed, the program will issue a main menu that contains commands to add a new task, edit the task (if it already exists), delete the task (if it already exists) and exit the program.

In the add new task menu, the program will ask the user to enter the name of the task they want to add. After they have entered the assignment name, they must enter the year. Input year here cannot be smaller than the current year. If this happens the user must enter another year. After that the user must enter the month. For this month's input is also the same, if the user input the year is this year and entered a month smaller than the current month or entered a month number smaller than 1 or greater than 12. Then the program will ask the user to re-enter the correct month. The same thing happened with the date input. If the user enters the same year and month but the date is smaller than the current date, the program will ask the user to re-enter the correct date. This can also happen if the user inputs an incorrect date in the month. For example, as in 2021 there is no 29th in the second month. If the user enters the 29th, the program will ask the user to re-enter the exact date. After that the user must enter the status of their assignment is important / not. If it is important then they will enter yes if not they will enter no. After that the user must press enter and will return to the main menu.

For the second menu, edit task. Here the user will be given the option to choose the task you want to edit and can also go back to the main menu. If the user has selected the task you want to edit. Then the program will display the task name, due date and status of the task. After that they can choose whether they want to change the name of the task or the date or change the status of their task or if they change their mind they can also cancel the edit. If they want to change the name the program will ask the user to re-enter the new name. If the user wants to change the date, the program will ask the user to input back starting from the new year, month and date. If the user wants to change their status, all they have to do is select the menu and the program will immediately change their status.

After that there is also a delete menu. Here after we select the delete menu the program will display the task prompt and the menu to return to the main menu. If we have selected the task sequence, the program will immediately delete the task.

After that the last is the program close menu. Every data that is inputted will be written every time we display it on the program, this will be very useful so that we can access our data in the future, this also means that every time you add a data , you can close the program without having to exit the program and the data will still be there.

<u>CODE EXPLANATION</u>

As we can see from the Main class, inside the public static void main, we have a new instance of MainHelper, that we call help that is used to hold functions that will be used throughout the Main class as well as an ArrayList based on Data that we call list, like function to display the menu, function to display the edit menu, function to display the tasks that also doubles as a function to write the tasks to the file, function to display the titles and thank you, function to create the txt file so the program can write the tasks that has been added by getting the data from the Arraylist, function to read the tasks from the txt file and then add it to the Arraylist so that it can be displayed in the program, we use delimiter to read the data that are placed between "#" sign, and there is also a function to write the task's arraylist to the file so that it can be displayed when we open the program later, in here we use "FileWriter" and "BufferedWriter" library to write the data.

After all of that Main helper finished, we prepare an integer called choice and set it as -1 as well as if there is a file in the same project file, the program will output "Save file exists", however, if the save file does not exist, the program will create the txt file called TaskList.txt and output "File created: TaskList.txt". Once the program has validated if there is TaskList.txt in the project file, the program will read the text file and show the list of the tasks that we have already created.

After that is finished, the program will print the title of the app "TO DO LIST" by using help.printTitle from the MainHelper class. After that the program will show the current date because there is the use of "MultithreadingTime updateDate = new MultithreadingTime();" and "updateDate.run();". After that the list of the task will also be sorted again because there is the use of "MultithreadingSort sortingThread = new MultithreadingSort();" and "sortingThread.run(list);". The Classes of MultithreadingSort and MultithreadingTime are created and used to have 2 separate threads at the same time.

Within the MultithreadingTime Class is the protected variable of Calendar named timeNow and two methods run() and a getter for the timeNow. The run() method stores the current date in the timeNow where every time the thread runs, the thread will get update the current time so that it is the current time. The getter will return the variable of timeNow.

Within the MultithreadingSort Class is the code that is used to sort the task list by its date and importance using:

```
Collections.sort( list, new SortDate() );
Collections.sort( list, new SortImportance() );
```

New SortDate() and new SortImporance() are comparators used in the sorting procedure.

The Comparator SortDate is a class that implements Comparator<Date> with the required method for a Comparator using two Date objects. The method getDueDate using getter in the Date superclass and stores them in two variables named x and y, the method returns x.compareTo(y) value of the two dueDates.

The Comparator SortImportance is also a class that implements Comparator<Date>. The compare method in the SortImportance is using two int variables to represent the importance of the two objects. If the object is instanceof Important then the int value is 1, otherwise 0. The method returns the deviation of the two int variables.

After that there is also “Calendar timeNow = updateDate.getTimeNow();” which will get today's date, month and year. And after that there is “int dayNow = timeNow.get(Calendar.DATE);”, “int monthNow = timeNow.get(Calendar.MONTH);” and “int yearNow = timeNow.get(Calendar.YEAR);”. We make them specific because we want to make our own format of the date today. The month has “++monthNow;” because month is started in java from 0, because of that we must add with ‘++’. After all of that finishes we print out the format with “System.out.println("Current Date: " + dayNow + "/" + monthNow + "/" + yearNow);”. After all of that print out, the program will display the list of the task using the help.read(list) preceded by calling the create function on the MainHelper class to check if there is a save file or not and will create a new txt file called “TaskList” if there is no such file.

After that it will show the Main menu, including the add task, edit task, delete task and close program. In the add task menu we will be having a String taskname. This String is used to collect the task name that is added by the user. After that there is also an int date that is set to 0, year set to 0 and month that set to -1. The last variable in the add task menu is the String to hold whether the task that is being added is categorized as important or unimportant. Next in the add menu is the part in which the user inputs the data for the new task. The String name cannot be longer than 25 characters, if else the program will ask the user to input again. Next is the input for the dates, the user is requested to input the data in separate time sections starting with year, then month and then date. The purpose of this is to make sure that the data for the task is not in the past. The calendar variable timeNow is updated after each input to ensure that the comparison used to check whether the date is set in the past is accurate. In the comparison itself we are using the method .getActualMaximum() to get the accurate value for a given year/month/date to check if the data is beyond the maximum (ex. If the date is 32, month 13, ect) and also compare the data with timeNow. After inputting the date, the dueDate for the task is complete and next we ask the user to determine if the task is important or unimportant with a yes or no question. If the answer is yes, the task object is created as an Important class object and added to the ArrayList, otherwise the task object will be an Unimportant class object. The final action in the add menu is to display that the task is successfully added and the user will be thrown back to the main menu.

In the edit task, help.cls() is called to create a space to see the menus and task list more clearly, help.displayTask(list) to print out the task list, first if the list is empty then there is nothing to edit and the user is back in the main menu. If the present list is not empty then the user is asked to input the index of the task in which it needs to be edited, the task index is stored within the int variable which Index and of course the action is repeated if the user inputted index which is not in the list.

The changes of the Task will be stored in 3 variables (Date x, String taskName and Calendar dueDate) using the selected index’s data through the use of the getter method in the Date Class.

The selected task is then displayed with all the information on its name dueDate and importance. The menu containing the edit options is also displayed using help.editMenu(), the variable int which stores the option chosen by the user. The options are 1. Edit Task Name, 2. Edit Task Due Date, 3. Edit Task Importance, and 4. Cancel. The process of editing the Task data is similar to the addtask. 1. Edit Task Name, is identical to the add task way of inputting name, the new taskName is stored in the String tempName and is set to be the new taskName using the line x.setTaskName(tempName). 2. Edit Task Due Date is also very similar to the add task input dueDate, the user is still requested to input the new edited dueDate separately, updating the timeNow after each input, in the last step the taskDue is set to the new user inputted date and the new dueDate is set using x.setDueDate(taskDue). 3. Edit Importance, first removes the selected index, creates the same object with the taskName and dueDate in a different class opposite of the starting class (Important -> unimportant and vice versa) and adds it to the list. 4. Cancel is self-explanatory, the edit task ends and the user is back to the main menu.

The Third task is the Delete Task, this action is very simple, first as in the edit task the list is first displayed and the user is asked to input the index to be deleted, then the index is removed using the list.remove(whichIndex).

After that, if the user input 4 in the menu the program will be finished and will print thank you because it is using “help.printThankYou();”.

REFERENCE

Visual Paradigm Online - Suite of Powerful Tools. (n.d.). Retrieved May 21, 2020, from https://online.visual-paradigm.com/

Java Create and Write To Files. (n.d.). Retrieved May 31, 2020, from https://www.w3schools.com/java/java_files_create.asp

Java Read Files. (n.d.). Retrieved May 31, 2020, from https://www.w3schools.com/java/java_files_read.asp

Java Delete Files. (n.d.). Retrieved May 31, 2020, from https://www.w3schools.com/java/java_files_delete.asp

Collections.sort() in Java with Examples. (2018, December 7). Retrieved May 28, 2020, from https://www.geeksforgeeks.org/collections-sort-java-examples/

Calendar Class in Java with examples. (2018, August 28). Retrieved May 26, 2020, from https://www.geeksforgeeks.org/calendar-class-in-java-with-examples/

Calendar set() Method in Java with Examples. (2019, February 21). Retrieved May 26, 2020, from https://www.geeksforgeeks.org/calendar-set-method-in-java-with-examples/?ref=rp

Calendar getActualMaximum() Method in Java with Examples. (2019, February 14). Retrieved May 27, 2020, from https://www.geeksforgeeks.org/calendar-getactualmaximum-method-in-java-with-examples/?ref=rp