Chaudhary Samir Zafar

AEE363

@00546317

Brief Summary | I HAVE USED JDK 13.0.2

I have chosen to develop an Array List data structure and I have used an abstract data type to help me implement this data structure. I used the data structure to make a console-based application.

DATA STRUCTURES AND ALGORITHMS

**Justification of choice**

The data structure I decided to use was an ArrayList, the main benefits of an ArrayList are:

Array-Lists are able to append dynamically, this means they do not hold a definite allocation of memory unlike traditional arrays. Unnecessary memory usage results in inefficiency and can limit the performance of an application hence why I believe using ArrayList is suitable as they can append upon the runtime of the application. Low memory usage is an advantage for applications and if my application was moved to a mobile device it would perform the same due to the low memory usage. This is an important factor to consider, as in modern society we need to design our applications suited to mobile devices as they are so popular.

My application involves the functionality of searching, adding and removing tasks from a toDoList. This means searching through the list and maybe retrieving a specific element. Hence, I took this into account, and I used an ArrayList as it has O(1) time complexity for getting a element in a certain index which in comparison to a linked list Is faster which has a time complexity of O(n) which means it has to traverse through all elements.

LinkedList do have the benefit of retrieving the first and last element a lot easier, however I believe this wasn’t relative to the application I was developing. I decided I could take advantage of the Array-Lists index-based system by referencing each element by an index which the user could see to aid them in removing any elements.

Another reason I chose Array-Lists is the time complexity relative to adding elements. Unless it has to copy to a new array it is the faster than a linked list. This means in most cases the time complexity for adding an element to an Array-List is O(1) compared to a linked-list which is O(n).

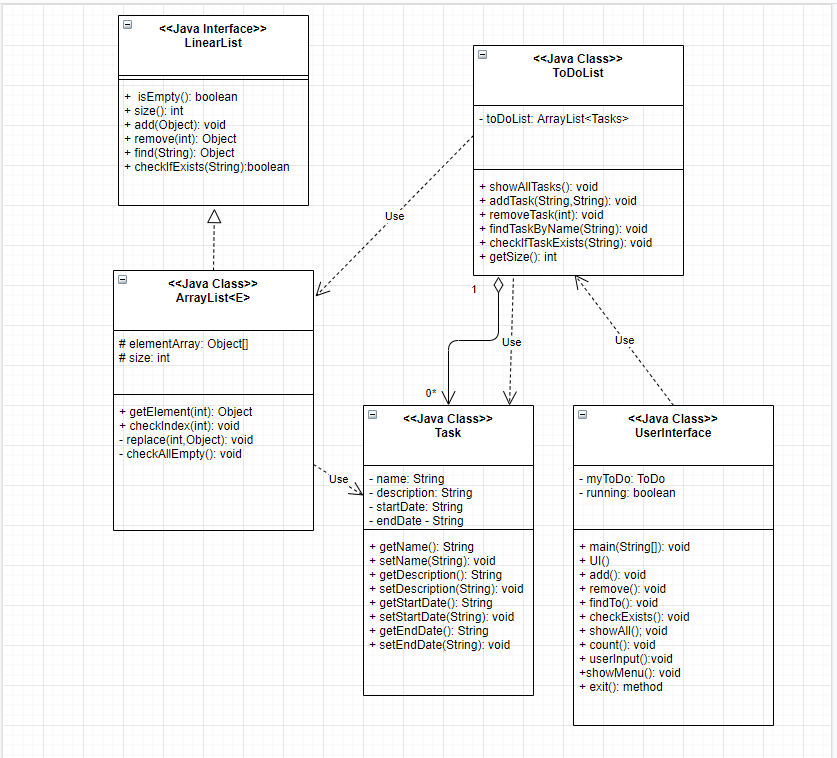
To conclude, I felt as if Array-Lists were the most suitable to carry out the functionalities of the application. Through process of elimination and identifying drawbacks and benefits of each data structure, i realized Array-Lists were the most efficient in terms of time complexity relative to my application.

**NOTE FOR TAREK : I HAVE USED JDK 13.0.2**

Sources : Data Structures and Algorithms in Java

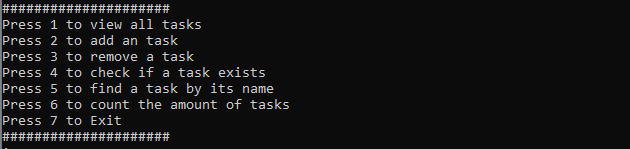
ISBM : 978-1-118-80857-3

Author – Micheal T. Goodrich, Roberto Tamassia, Michael H. Goldwasser

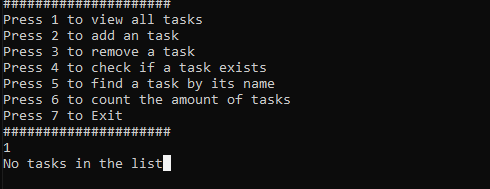
**UML Diagram**

**Screenshots**

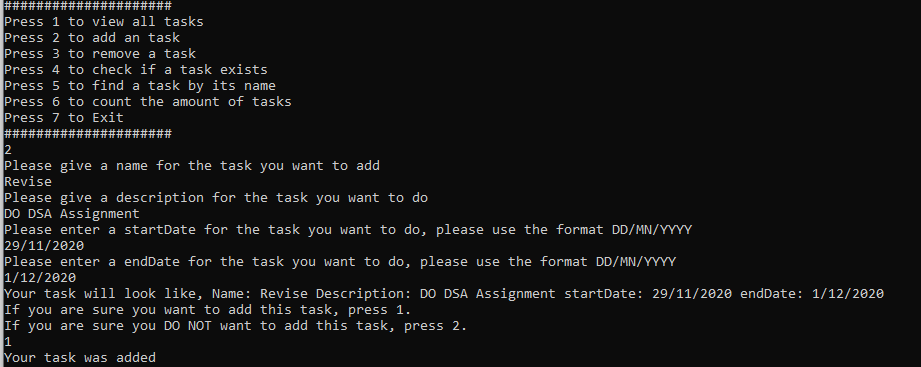
I have sections inputs for the user to interact with the application.



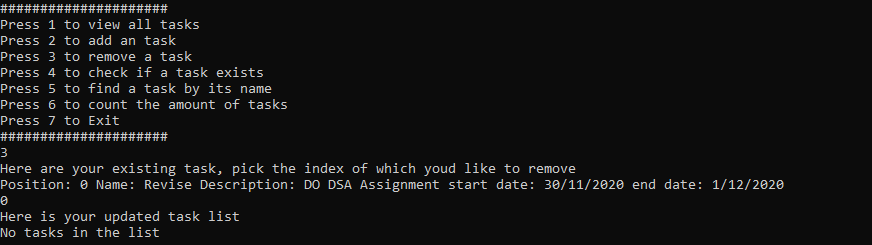
This menu is displayed when the user enters the application and shows the available options for the user and what the outcome of the interaction will be.



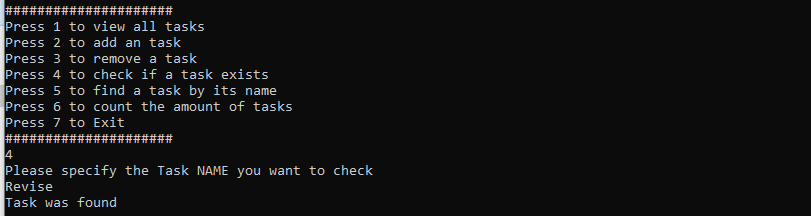
This screenshot display how the user can view all their tasks in their list, for this instance there is no current tasks in the list.



This screenshot shows the application can successfully add items to the list, when the item is added the application outputs a success message and I have added a verification stage for extra choice.



This screenshot shows the procedure for the removing an item. The user is provided with all of the available indexes to choose from and can successfully remove any item.



This screenshot shows the output of the function of checking if a task exists. The task I checked existed so my application output “the task was found” if the task was nonexistent then the output would be “Task was not found”



The screenshot shows the application takes a task name and returns the task name and position, if the name is invalid or not found the user will be notified about it.

If the user presses 6 then the application will display the size of the toDoList.