**Integrative Task#1 Computation and Discrete Structures 1**

**By: Juan Pablo Parra, Stick Martinez and Daniel Gonzalez**

**Engineering Method:**

1. **Identification of the problem:**

**Context:**

It is necessary to design a system for managing tasks and reminders which allows you to do three main things: add, organize and manage your to-do's and reminders. When adding tasks and reminders it will be necessary to store both elements. Since we must work with priorities, the tasks will have two categories: "Priority" and "Non-priority". For priority tasks, when adding a new task it will be registered in the task queue as the most important tasks are displayed first. Non-priority tasks are managed on a first-come, first-served basis. Finally, you need to revert the actions performed by the user, for this it is necessary to record every action you make, whether you added a task, modified it or not.

**Needs and Symptoms:**

* Add to-to´s and reminders

For this we are going to use a hash table which has an identifier i.e. a key in terms of hash and the task information is its value in terms of hash.

* Organize to-to´s and reminders
* Manage to-to´s and reminders
* For priority task create two categories "Priority" and "Non-priority"

This will require the use of a priority queue.

Priority tasks need to be inserted into the priority queue according to the level of importance allowing important tasks to be handled first.

On the other hand, non-priority tasks will be performed on a first-come, first-served (FIFO) basis.

* Undo Method

This will require the use of a queue that keeps track of the actions performed by the user. These can be: adding, modifying or deleting a task.

* For the project to be successful it is also necessary the presence of some indirect aspects such as the good documentation of the project and the efficiency of the algorithms used if you work with large tasks or reminders.

**Problem definition:**

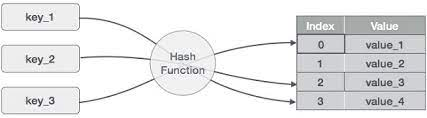
A system needs to be created that allows to add, organize and manage tasks and reminders by organization standards such as deadline or user specified priority.

**Specification of requirements:**

**\*\*Its in another document in the folder docs\*\***

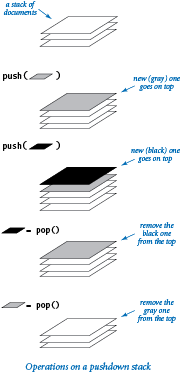
1. **Compilation of information**

**Hash tables:** A hash table is a type of data structure that allows us to store keys and values just like a dictionary. Using the hash function, each key is processed so that it can be taken to the place where the information is to be stored. Following this line, it is normal that the items collide when they are being stored, that is why techniques such as "Separated chaining" and "Open addressing" are created, where the positions closest to where the key was hashed are used to store them there. In the following image we can see in more detail how the key is processed and the sorting of the information in a hash table is carried out.



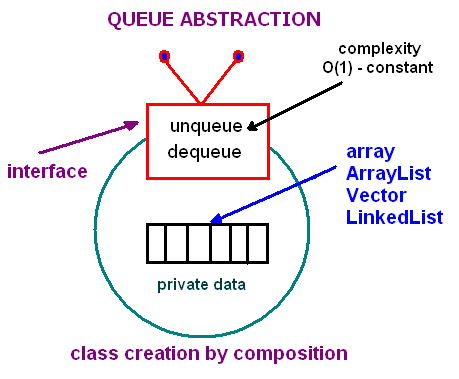
Source: <http://bitybyte.github.io/Hashtables/>

**Stacks:** A stack is a collection of data where the information entered is sorted by the "Last in First out" method, i.e. the first item in is the last one out, and the last one in is the first one out. The stacks enter their items utilizing the push() function and remove them using the pop() function. The following image shows what a stack looks like.



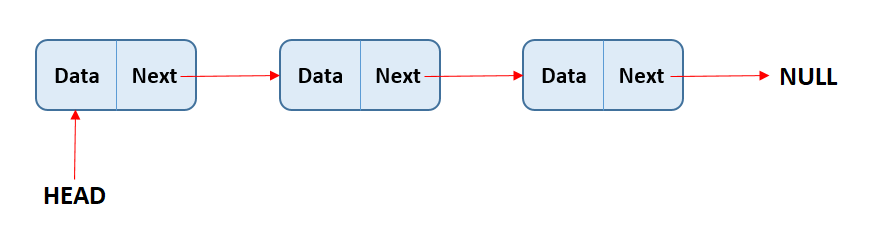
Source: <https://introcs.cs.princeton.edu/python/43stack/>

Queues: A queue is a container of items, in which these items are entered by means of the "First in First Out" method, that is to say, in the queues the first item that enters the queue is the first one that leaves. To do this, the functions offer() to add and remove() to remove are used.



Source: <https://www.andrew.cmu.edu/course/15-121/lectures/Stacks%20and%20Queues/Stacks%20and%20Queues.html>

**Linked List:** A linked list is a type of data structure in which each position in the list is a node. The lists are divided into simple linked, doubly linked and circular lists, the main difference of these is that the simple list only knows its next one while the doubly linked list knows its previous and next one as well as the circular list. The following image shows the operation and behavior of a linked list.



Source: <https://www.alphacodingskills.com/ds/doubly-linked-list.php>

1. **Search for solutions:** During the use of tasks and reminders, the database, dates and priorities will be used to meet the needs of the system.

**Save & Load**:

* Txt type file.
* Comma Separated Values type file
* Json/Gson type file
* XML type file

**Date:**

* Date class
* Java Calendar
* Date format

**Priority:**

* Collect sorts
* Classify sorts
* Insert sorts
* Bubble sorts

**Data save:**

* Hash Table
* Linked lists
* Arrays
* Binary Tree Search
* Stacks
* Queues

1. **Preliminary Designs:**

Here is it the discarded options:

**Save & Load:**

* XML type file: Due to the complexity of the XML file, this option is discarded.

**Date:**

* All options are necessary in this step,

**Priority:**

* All options are necessary in this step, because they are necessary when implementing priorities and organizing tasks.

**Data save:**

* Binary Tree Search: Due to the fact that we are sorting the elements in regards of priority and entry order.
* Array: We discard this option, because the alternative to using arrays is a linked list, besides being more practical this option.

1. **Testing and Solution Selection:**

Criteria must be defined to evaluate the solution alternatives and, based on this result, choose the solution that best meets the needs of the problem. The solution that best meets the needs of the problem posed. The objective is to establish a weight that indicates which of the possible values of each criterion have more weight.

**Parameter A.** Accuracy of the solution.

[2] Exact (an exact solution is preferred).

[1] Approximate

**Parameter B.** Efficiency. A solution with better efficiency than the others considered is preferred.

[4] Constant

[3] Greater than constant

[2] Logarithmic

[1] Linear

**Parameter C**. Completeness. A solution that finds all solutions is preferred.

[3] All

[2] More than one if any, but not all.

[1] Only one or none

**Parameter D**. Ease of algorithmic implementation:

[2] Compatible with the basic arithmetic operations of modern computer hardware.

[1] Not fully compatible with the basic arithmetic operations of a modern computer.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Save and Load | **Parameter A** | **Parameter B** | **Parameter C** | **Parameter D** | **Total** |
| Txt type file | 1 | 4 | 2 | 1 | 8 |
| Comma Separated Values type file | 1 | 3 | 2 | 1 | 8 |
| Json/Gson type file | 2 | 4 | 3 | 2 | 11 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | **Parameter A** | **Parameter B** | **Parameter C** | **Parameter D** | **Total** |
| Java Calendar | 2 | 4 | 3 | 2 | 11 |
| Date class | 2 | 4 | 3 | 2 | 11 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date and Save | **Parameter A** | **Parameter B** | **Parameter C** | **Parameter D** | **Total** |
| Hash Table | 2 | 4/1 | 3 | 2 | 11/8 |
| Linked lists | 2 | 1 | 3 | 2 | 9 |
| Stacks | 2 | 4 | 3 | 2 | 11 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Priority | **Parameter A** | **Parameter B** | **Parameter C** | **Parameter D** | **Total** |
| Collect sorts | 2 | 1 | 3 | 2 | 8 |
| Classify sorts | 2 | 1 | 3 | 1 | 7 |
| Insert sorts | 2 | 4 | 1 | 2 | 9 |
| Bubble sorts | 1 | 3 | 2 | 1 | 7 |

1. **Reports and specifications**
2. **Implementation**