**ENGINEERING METHOD - TI 1**

**IDENTIFICATION OF THE PROBLEM:**

In this project we are asked, by a recognized Health Care Provider Institution, to make a first version of a system to manage the admission and discharge of patients in a Clinical Laboratory. This program will be executed by the staff located at the reception of the center who will be responsible for performing the admission process and then direct the person to one of the two current units of the laboratory (Hematology and General Purpose) with an assigned turn of attention. When a patient arrives at the laboratory they must be searched for, or entered if they are not in the system. The system should have an "undo" option each time an admission or discharge action is performed, to make it possible to correct these possible mistakes. Finally, the system must have a panel that allows monitoring the list of people currently in the laboratory at all times.

R1. The system should allow searching for a patient from the system database.

R2. The system must allow to register a patient in the system database. The patient's id, name, gender, age and priority will be saved.

R3. The system must allow to enter a patient in the queue of one of the two sections, hematology or general purpose.

R4. The system must allow to remove a patient from the queue of one of the two sections, hematology or general purpose.

R5. The system must allow undoing the queue entry or exit.

R6. The system must allow to review either of the two section queues.

**- GATHERING THE NECESSARY INFORMATION:**

To solve this problem, we will need to make use of data structures that allow us to store patient information in the system and also create a queue that is determined by the priority value.  Therefore we will make use of two abstract data structures, one is Priority Queue and the other will be Hash table. Priority queue is a type of queue in which each element has an associated priority value. This priority value is used to know the order in which the elements will be removed from the list. If two elements have the same priority value, they are removed as they entered the queue. In this project, this abstract data type will be used to simulate the queue per shift that we are asked to queue patients in each section. The priority value will be the priority that the patient has in terms of attention. For the operation of the Priority queue we will use Heap, an IPriorityQueue interface and a Node class.  On the other hand, the Hash table is a data structure that stores data associatively. In a hash table, data is stored in an array format, where each data value has its own unique key. Access to the data is very fast if we know the key of the desired data. In this data structure we will store the patients that are registered in the system, these will be the ones loaded in the database and the new ones that are registered directly from the program. For the operation of the Hash table we will use Chain hash table, an IHashTable interface and a Node class. The methods of both structures will allow us to develop the project more easily.

In addition to this, we will use a database made by us. This will be loaded with patients for the execution of the program. Patients that are registered directly in the program will also be saved in this database.

<https://www.geeksforgeeks.org/priority-queue-set-1-introduction/>

<https://www.geeksforgeeks.org/hashtable-in-java/?ref=gcse>

<https://www.geeksforgeeks.org/hashing-data-structure/?ref=gcse>

<https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm>

**- SEARCH FOR CREATIVE SOLUTIONS:**

 For this project, different data structures presenting different strategies, with different spatial and temporal complexities, are presented as a possibility. Therefore, in search of an efficient solution, the data structures found are the following:

***Alternative 1. Simple linked list***

A simple linked list is a data structure in which each element points to the next. In this way, having the reference of the beginning of the list we can access all the elements of the list.

***Alternative 2. Double linked list***

A doubly linked list has an additional pointer known as the previous pointer at its node, in addition to the data part and the next pointer as in the simple linked list.

***Alternative 3. Array***

An array is a data structure that allows us to store a large amount of data of the same type. The size of arrays is declared first and cannot be changed at runtime.

***Alternative 4. ArrayList***

It is a class that allows to store data in memory in a similar way to the Arrays, with the advantage that the number of elements that it stores is done dynamically, that is to say, it is not necessary to declare its size as it happens with the Arrays.

***Alternative 5. Hash Table***

A hash table or hash map is a data structure that associates keys or keys with values. The main operation that it supports efficiently is the search.

***Alternative 6. Queue***

A queue is a linear data structure or collection in Java that stores elements in FIFO (first in, first out) order.

***Alternative 7. Stack***

A stack is a linear data structure that has a single fixed access point through which elements are added, removed, or queried. The access mode to the elements is LIFO (Last In First Out).

***Alternative 8. Heap***

A heap is a tree-like data structure with information belonging to an ordered set. Maximum heaps have the characteristic that each parent node has a value greater than that of any of its child nodes, while in minimum heaps, the value of the parent node is always less than that of its child nodes.

***Alternative 9. Priority queue***

A priority queue is an abstract data type similar to a queue in which elements are additionally assigned a priority. In a priority queue, an element with a higher priority will be removed from the queue before an element with a lower priority.

**- GO FROM THE MAIN IDEA TO THE PRELIMINARY DESIGN:**

We have 9 data structure alternatives to store information in these, however, there are some structures that can be easily discarded, for example:

Array is a linear array that any operation of this structure carries a time complexity of O(n), moreover, it has a limit of available spaces to store data. The ArrayList structure does not present this last space problem as in Array, but its operations have the same time complexity, i.e. O(n). For data insertion, the Nodes option is a good choice, however, the doubly linked list is not functional in this type of exercise. Finally, the Stack complies with the LIFO modality, but for this exercise it must be handled with an additional attribute, the priority. Therefore, the Stack does not evaluate this priority and in that case is not functional for this problem context.

A careful review of the other alternatives leads us to the following:

***Alternative 1. Simple linked list***

* With them you can represent different types of data structures.

***Alternative 5. Hash Table***

* The major advantage of using a hash table is the speed of search.

***Alternative 6. Queue***

* For a shift implementation it is the ideal data structure, because it handles First In - First Out (FIFO) philosophy.

***Alternative 8. Heap***

* In a priority heap, the largest element (or the smallest, depending on the chosen ordering relation) is always at the root node. For this reason, heaps are useful for implementing priority queues.
* The efficiency of operations on heaps is crucial in various traversal algorithms.

***Alternative 9. Priority queue***

A priority queue in which elements are inserted in monotonically increasing order; and therefore the last inserted element is always the first to be retrieved (since it will have the highest priority so far). In a queue, the priority of each inserted element is monotonically decreasing; and therefore the first inserted element is always the first to be retrieved (as all subsequent elements will have lower priorities).

**- SOLUTION EVALUATION AND SELECTION:**

Criteria:

Criteria must be defined that will allow us to evaluate the solution alternatives and based on this result choose the solution that best meets the needs of the problem posed. The criteria we chose in this case are listed below.

***Criterion A. Algorithmic Complexity:***

* [1] High
* [2] Medium
* [3] Low

***Criterion B. Functionality for Exercise:***

* [1] Non-functional
* [2] Functional

***Criterion C. Complexity of Implementation:***

* [1] High
* [2] Medium
* [3] Low

***Criterion D. Compatibility with other Data Structures:***

* [1] Poor
* [2] Good

***Evaluación:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Name/Criteria | CriteriaA | CriteriaB | CriteriaC | CriteriaD | Total |
| ***Simple linked list*** | 2 | 2 | 3 | 2 | 9 |
| ***Hash Table*** | 1 | 2 | 2 | 2 | 7 |
| ***Queue*** | 2 | 1 | 2 | 1 | 6 |
| ***Heap*** | 3 | 2 | 3 | 2 | 10 |
| ***Priority queue*** | 2 | 2 | 2 | 2 | 8 |

***Selection:***

According to the above, the one with the lowest score, i.e. Queue, should be discarded. In addition, due to the compatibility among the remaining data structures, also, thanks to their high weight, the 4 data structures out of 9 preliminary ones, which are the following, should be finally selected:

***Simple linked list***

For the use of the Nodes in other methods such as the Heap and the Hashtable in case a collision occurs and a chain method must be performed.

***Hash Table***

To store the information of the Patients, and to have a fast and efficient search of the data of the respective person.

***Heap***

To organize with certain criteria (Maximum or Minimum), the priority values that will be passed to the Priority Queue for taking shifts.

***Priority queue***

To perform the shift call in the IPS, in an efficient way with simple implementation.