**Engineering Design Method**

**Problem identification:**

The specific needs of the context of the problem, along with its resolution conditions are these ones:

* The reception staff of a Healthcare Provider Institution needs to manage the entry and exit of patients to the Clinical Laboratory.
* There is no system, within the Health Provider Institution, that allows the administration of the entry and exit of patients to the Clinical Laboratory.
* There are people with priority and people without priority.
* People with higher priority should enter before people with lower priority.
* The solution must ensure efficiency in patient registration.
* The solution must guarantee efficiency in the search for patients.
* The solution must allow you to undo check-in and check-out actions.

Problem definition:

A recognized Health Provider Institution requires a system that allows managing the entry and exit of patients in a clinical laboratory; this should allow patients to be organized between general and priority access to be treated respectively.

**Information gathering:**

* Stack:

It is a list or data structure that allows data to be stored and retrieved. Its main feature is its mode of access to elements, which is the L.I.F.O type (Last in, first out).

* Queue:

It is a data structure, which is used to store elements. Its main feature is its mode of access to elements, which is the F.I.F.O type (First in, First Out).

* Priority queue:

It is a data structure, which is used to store elements, and works just like a normal queue does, but includes the difference that, in it, each stored element has an assigned priority. Its main characteristic is that the elements with higher priority come out of the queue before those with lower priority. While the elements with the same priority leave the queue, following the order of the queue. In addition to this, it is important to add that its implementation is done through heaps.

* Heap:

It is a data structure, which is used to store elements, which is usually an array that can be seen as an ABB (Binary Search Tree). In this structure, each ABB node is an element within the array.

It is important to say that it is not really an ABB, it is just an arrangement of elements that we imagine as an ABB. In addition to this, we see it as an almost full ABB since it may not be full at its lowest level. If it is not full at its lowest level, it will always be full, from the left to its lowest level.

* Hash table:

It is a data structure, which is used to store elements, and which implements another data structure called Dictionary. This structure associate’s keys with values. One of its main characteristics is the average time of its operations is *O*(1), especially in the search, which makes it a very viable option when looking for efficiency in it. It works by transforming the key with a hash function into a hash, a number that identifies the position where the hash table locates the desired value.

* Linked-List:

It is a linear collection of data elements where each element points to a next element and, optionally, to a previous element. It is, in other words, a collection of nodes that, when put together, constitute a sequence.

This structure can be used to implement other types of structures such as hash tables, stacks, queues, etc.

**Search for solutions:**

Alternative 1:

Make use of priority queues to assign shifts to patients with higher priority, queues to assign shifts to people who do not have any priority, then use hash tables to group them all. Also make use of the Stacks or stacks to implement the option to undo the actions performed.

Alternative 2:

Make use of a single priority queue to enter all patients and assign them their shift, and a hash table to save all patients. Also make use of the Stacks or stacks to implement the option to undo the last action performed.

Alternative 3:

Make use of priority queues to assign patients with higher priority their turn and likewise use simple queues to assign other patients in general care. Also make use of the Stacks to implement the option to undo the last action performed.

**Transition from brainstorming to preliminary designs:**

As a first measure, Alternative number 3 is discarded because we consider it the most incomplete to implement a solid and feasible solution using good practices for a software implementation, since by implementing only two queues, one of priority and a simple one, without making use of a data structure such as the hash table, makes the algorithm much more inefficient in the long run.

Review of the other alternatives:

Alternative 1:

* - It is precise and can have good practices in the process of its implementation. However, it can become lengthy and prone to bad practices if you don't take care of how to implement priority queues correctly.
* This alternative implies implementing its own hash table, as well as the priority queue, the queue, and the stack, for its correct operation.

Alternative 2:

* By making use of a single priority queue, we are delegating many functions to a single part of the system, which would cause us to encounter bad practices, violating software design criteria such as the partitioning of duties.

**Evaluation and selection of the best solution:**

Evaluation criteria:

* Criterion A.

Compliance with design criteria.

* Criterion B.

Flexibility. Flexible solution and future scalable solution.

* Criterion C.

Completeness. It makes use of everything necessary to be functional and sustainable in the future.

Evaluation mode for each:

Numerical, 0 being the one that least meets the criteria and 5 being the one that fully meets the criteria.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Criterion A | Criterion B | Criterion C | Total |
| Alternative 1 | 4 | 4 | 5 | 13 |
| Alternative 2 | 2 | 3 | 2 | 7 |

Selection:

According to the previous evaluation, alternative 1 should be selected, since it obtained the highest score according to the defined criteria.

**Specification of requirements:**

**R1**: The program should allow reception staff to enter the patient with their complete information in a database.

**R2**: The program should allow the reception staff to record the patient's admission in a database.

**R3**: The program should allow reception staff to direct the patient to one of the two laboratory units (Hematology and General Purpose).

**R4**: The program should allow reception staff to search for a patient in the database.

**R5**: The program should allow reception staff to assign a priority to a patient from the database.

**R6**: The program must allow the reception staff to enter priority patients with priority (those who have some important underlying disease, are older adults, are pregnant, etc. This information is extracted from the laboratory's database).

**R7**: The program shall allow reception staff to admit general access patients on a first-come, first-served basis.

**R8**: The program should allow reception staff to discharge patients who had previously been admitted.

**R9**: The program should allow reception staff to undo the action of entering or exiting patients in the event of a mistake.

**R10**: The program must have a panel that will allow the reception staff to always monitor the list of people currently in the laboratory.

**R11**: The program must have a panel that will allow the reception staff to always monitor the order of attention of the people in each unit.

**R12**: The program must have a panel that will allow the reception staff to always monitor the discharge of patients in order to continue with the care of the following people.

**BONUS:**

**R13**: The program must ensure that in a random time between 1 or 2 minutes a patient is discharged from the units.

**R14**: The program must ensure that the person automatically served by the unit does not appear on the waiting list but must still appear present in the system so that later the reception staff manually discharge the person.