**Engineering design method**

**Phase 1**

**Context description:**

The management of records in an organization is essential for its proper functioning, the flow of people, the provision of services, as well as the registration of customers are aspects that must be controlled with great precision to ensure an efficient and agile operation without cost overruns; However, this process can be affected by multiple factors, such as the use of the classic paper-based registration model, which leads to inefficiency, is subject to numerous human errors and can lead to disorder and loss of information; or the implementation of a poorly designed registration system that does not meet its objective of speeding up processes and that can’t be trusted to manage an organization’s data. In the healthcare sector, to which the healthcare provider requesting the solution belongs, the importance of this process increases, because the integrity of the clients is at stake, so workers must be effective and accurate with their actions, and they must be helped by a reliable and efficient tool that can manage a large flow of data.

**Symptoms and Needs:**

* The client wants a database simulated with a simple text file.
* The client needs that the method that searches the patients in the database should do it in the most efficient way possible because they deal with a big number of patients.
* The client wants the system to give the patients priority based on their conditions.
* The client wants an undo button in case of human error since it's brand-new software.
* The client wants to have a panel that allows the user to monitor the queues, the patients inside the building, and waiting for checkout.

**Problem Identification:**

A Health Provider Institution requires a software program that allows them to manage the entry and exit of patients in the clinical laboratory.

**Requirements specification:**

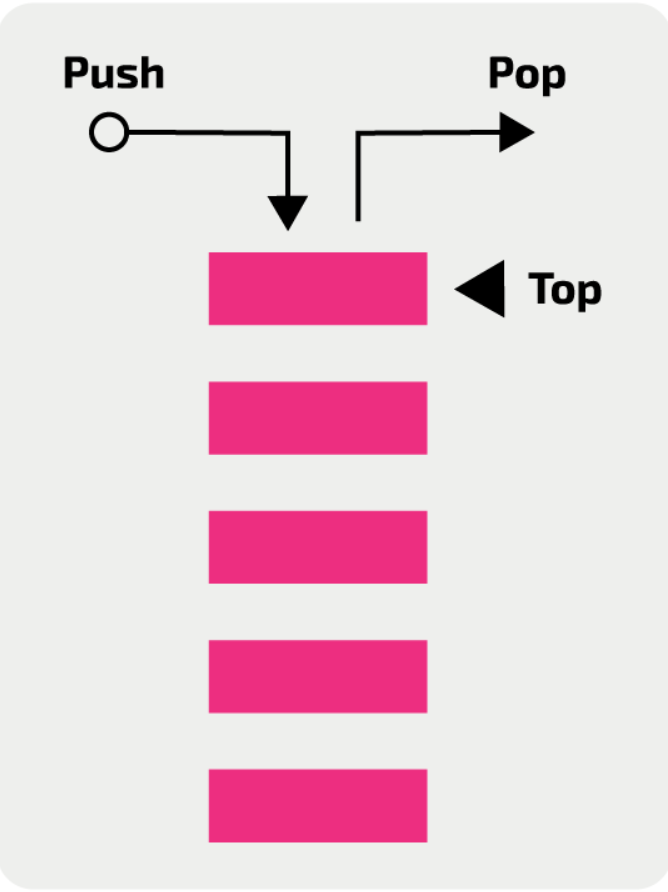
**Note:** The “requirements analysis” file is located in the “doc” folder.

**Phase 2**

**Data structures:**

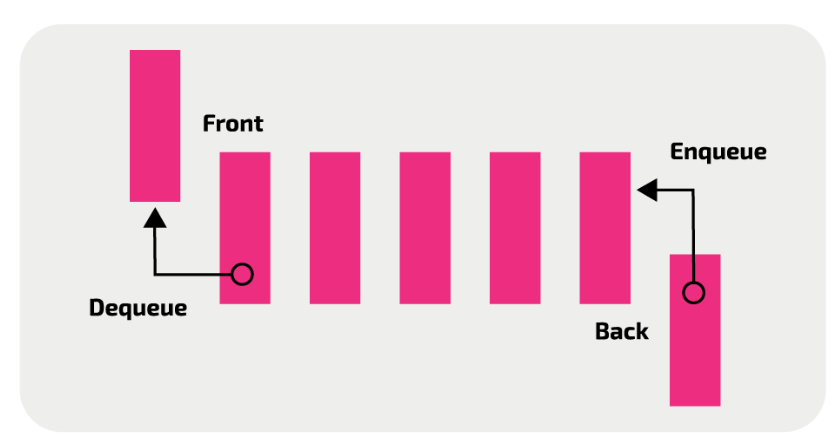
Since the program must manage people, several data structures were considered to store the necessary information in different ways according to the needs of each requirement.

* **Stack:** Is a data structure that holds elements in a linear and ordered sequence. It’s a LIFO (last in, first out) structure, and the main operations are the following:
  + push(e): adds an element at the top of the stack.
  + pop(e): removes the element at the top of the stack.
  + peak(e): returns the element at the top of the stack.
  + isEmpty(e): returns true if the stack is empty or false otherwise.



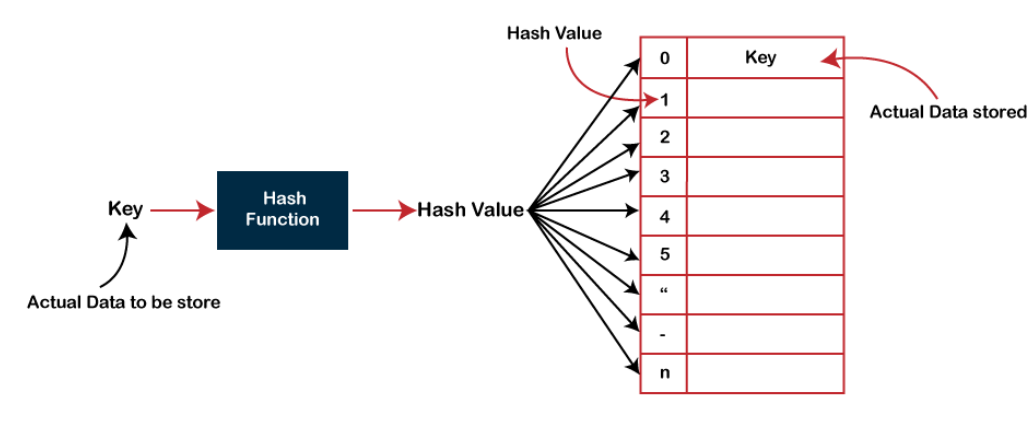
Source:<https://isaaccomputerscience.org/concepts/dsa_datastruct_stack?examBoard=all&stage=all>

* **Queue:** Is a data structure that holds elements in a linear and ordered sequence. It’s a FiFO (first in, first out) structure and the main operations are the following:
  + enqueue(e):adds an element to the back of the queue.
  + dequeue(): removes the element at the front of the queue.
  + peek(): returns element at the front of the queue.
  + isEmpty(e): returns true if the stack is empty or false otherwise.



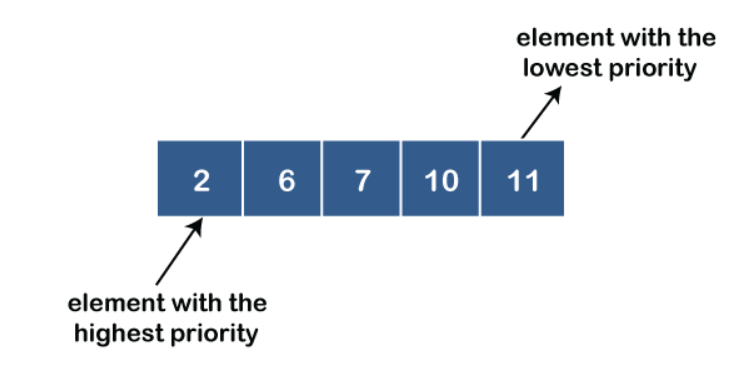
Source:<https://isaaccomputerscience.org/concepts/dsa_datastruct_queue?examBoard=all&stage=all>

* **Hash table:** Is a data structure that is known for its efficiency saving and searching elements, it has a definite number of positions and it works with a hash function that transforms a given key (associated to an object) to an index of the table.The main operations are the following:
  + Insert(e): adds an element to the table in the position given by the hash function.
  + Search(k): returns an element applying the hash function to the given key and searching in that position.
  + Delete(k):removes an element applying the hash function to the given key and searching in that position.



Source: <https://www.javatpoint.com/hash-table>

* **Priority Queue:** is a data structure in which each element has a priority (key), the element with the highest priority comes first at the queue, and the elements are removed according to their priority, from highest to lowest or vice versa depending on the type of priority queue. The main operations are the following:
  + Insert(e): adds an element to the queue according to its priority.
  + Maximum(): return the element with the highest priority.
  + ExtractMax(): removes the element with the highest priority;
  + IncreaseKey(e,k): increases the key (and thus the priority) of a given element.



Source:<https://www.javatpoint.com/ds-priority-queue>

**Automize the client's attention with Timer and TimerTask or with Runnable, ScheduledThreadPoolExecutor, and ScheduledExecutorService :**

To automate the client's attention, we should count the time and find a way to execute the action in the system in an interval of time.

* To establish the activities that will automatically execute, we might use the Timer and TimerTask classes, where the first one will allow us to schedule a task by periods in milliseconds or run it just once, the second consists of a function that should be scheduled.
* Runnable is an interface that a class should implement, where its instances are focused to be executed by a thread. We are going to execute our runnable with the class ScheduledThreadPoolExecutor and the interface ScheduledExecutorService which will allow us to schedule our runnable with a given delay and period in various units of time, and use methods such as cancel to stop or thread.

Sources:

* <https://www.geeksforgeeks.org/java-util-timer-class-java/>
* <https://www.geeksforgeeks.org/runnable-interface-in-java/>
* <https://www.geeksforgeeks.org/scheduledthreadpoolexecutor-class-in-java/#:~:text=ScheduledThreadPoolExecutor%20class%20in%20Java%20is,delay%20for%20some%20future%20time.>
* <https://www.geeksforgeeks.org/scheduledexecutorservice-interface-in-java/>

**Data loading and saving:**

Since there are previous users that must be loaded in the the program to perform the operations a tool for reading and writing text files is needed.

* **Json:**

To read and write files we can use the gson jar developed by google that allows us to serialize and deserialize between Java objects and their representation in JSON notation in addition to the use of FileInputStream and FileOutputStream, the first one gets input bytes from a file in a file system and the other is an output stream to write data in an existing file or create it in case it does not exist.

Source:<https://www.digitalocean.com/community/tutorials/java-json-example>

* **Txt with personalized format:**

The File class is used to find and save the path of the file that is going to be modified. The reading of data can be done from a plain text file with the help of the FileReader and BufferedReader classes, the first one works as a main stream and the second one as an intermediate stream, and it is also possible to write in the text file with FileWriter and BufferedWriter, all this can be handled with a personalized format that determines the way in which the text is read and written.

Source:<https://www.ecodeup.com/como-escribir-y-leer-archivos-de-texto-plano-en-java/>

**Safety measures:**

Measures to ensure the correct functioning of the program are needed in order to minimize the possibility of errors occurring.

* **Exceptions:**

Exception handling in java is an effective way to handle runtime errors so that the application does not crash its execution. When an exception occurs an object is generated containing a name, error description and system status, both default java exceptions and user designed exceptions( to handle specific cases) can be "trapped" to allow the normal flow of the system.

Source:<https://www.geeksforgeeks.org/exceptions-in-java/#:~:text=Exception%20is%20an%20unwanted%20or,method%2C%20it%20creates%20an%20object>.

* **Tests:**

For testing there is JUnit, a Java framework that allows you to generate test scenarios linked to the classes/objects you need to check and use them to perform unit tests with the help of “assertions” that allow you to make assumptions that will be checked when the test is executed.

Source:<https://www.vogella.com/tutorials/JUnit/article.html>

* **generics:**

The use of generics allows "types" to be parameters when defining classes, interfaces and methods. It is used to improve software stability because of the checking of "types" at compile time, thus reducing run-time errors that can be problematic.

Source:<https://docs.oracle.com/javase/tutorial/java/generics/why.html>

**Related solutions:**

* **IPS of the city:** its attention method works in such a way that when entering the place, users find a machine that generates tokens, where the user enters their id, selects the type of attention, the machine prints a paper with their turn and it will be added to the attention queue, and the workers of each box will call people based on the time you booked your appointment and the type of queue that each stall is in charge of, once the user is called to the delivery box your ticket to access care.

Source: implemented by “IPS VIVIR”

* **Qmatic:** it allows users to book an appointment from an electronic device with an Internet connection to the virtual queuing system so customers can wait anywhere while monitoring their progress in real time and receive notifications when it is their turn to be attendent. When customers arrive they check-in through a self-service kiosk, mobile device, or front desk, it also gives customers all the information they need while they wait, like regular updates on wait times via phone or displayed digitally on screens. And it offers a service with online surveys and applications so that customers can rate how the attention went, which provides information to create even better services. It stores the information in the cloud.

Source:<https://www.qmatic.com/blog/queue-management-systems-in-hospitals>

* **virtuaQ:** Allow the client to request an appointment remotely from anywhere, it works under a queue system with an SMS token, and it allows you to send notifications to inform about what kind of prepartion must be done before coming to the appointment. Allows important customers to be moved forward in queue without others knowing. Managers can reduce wait time and also respond to appointment breach. It stores the information in the cloud.

Source:<https://virtuaq.com/healthcare#:~:text=Reduce%20wait%20time%20for%20patients,soon%20as%20it%20is%20done.>

**Phase 3**

For this point we used brainstorming and the construction of scenarios, with the first technique, and with help of the previous research and preliminary ideas, multiple implementation options were obtained for the program; and in the second technique a few hypothetical scenarios that could happen in the program were built as well as the expected response for each case, thus collecting ideas that had been overlooked and that serve to solve these possible scenarios.

**Proposed solutions:**

We decided to divide the solutions into different modules that collect the main functionalities of the program in order to obtain a combination of solutions (one per module) that together provide an answer to the complete problem.

1. **Load and save:**
   1. Use Json notation in a txt file.
   2. Use a simple txt file.
   3. Use an Excel file.
   4. Use a Csv file.
2. **Database implementation:**
   1. Use an arrayList.
   2. Use a BST.
   3. Use a hash table
   4. Use a chained hash table
   5. Use an array.
3. **Undo Actions**
   1. Stack implemented with Linked List.
   2. Stack implemented with ArrayList.
4. **Attention queues**
   1. One priority queue implemented with heap for each unit.
   2. One priority queue and one regular queue for each unit in order of arrival.
   3. One priority queue implemented with bubble sort for each unit.
5. **Automatic attention**
   1. Discard it.
   2. Use Timer and TimerTask.
   3. Use Runnable and ScheduledThreadPoolExecutor.
   4. Use Runnable, ScheduledThreadPoolExecutor and ScheduledExecutorService.
   5. Use timer that it’s executed we certan action is performed.

**Phase 4**

**Discarded solutions:**

1. Load and save:

1.3. We don't have the knowledge to implement this kind of solution

1.4. Since we are working with Strings, having the values separated by commas might cause some problems

1. Database implementation:

2.1.Since the database has to contain a lot of people, the structure used for this has to be efficient, and the operations that implement the ArrayList don’t meet this condition.

2.5. Array’s efficiency is an obstacle for the same reason and in addition is limited to a certain number of spaces.

1. Attention queues:

4.3. Since bubble sort has complexity it just doesn't fit the efficiency part

1. Automatic attention:

5.1. We research how to schedule a task and feel like we were capable of doing in it

5.5. With this option we can keep track of time in order to automatically attend a patient, but the operation won’t be executed immediately

**Possible solutions:**

1. Load and save:

1.1. Json:

* It separates every object between curly brackets where every attribute of the object is separated by a comma. We can also use Gson jar from google to use methods that simplify its use and make it easier since they can automatically turn objects to Json and vice versa.

1.2. txt

* + We have to implement methods to separate each object and its attributes. And to turn the data into objects again.

1. Database implementation:

2.2. BST :

* BST is a way of organizing and containing information through linked nodes.
* The first entry (Root) defines the organization of the tree, on the left side will go the values smaller than the root and on the right side the larger ones. This allows to search for information in O(Log n)

2.3. Hash Table

* It is a form of data organization and containment with a size limit where it can contain certain nodes.
* To insert the data in the Hash table, each element must have a unique numeric identification (key) that goes through a preset function thet gives the index where the element is located.
* The Information search is done in O(1).

2.4. Chained Hash Table

* It is a form of data organization and containment with a fixed size but no limit on its storage capacity.
* The chaining allows the Hash Table to place more than one node in the same position of the array since they will be linked to each other.
* To insert the data each element must have a unique numeric identification (key) that goes through a preset function thet gives the index where the element is located.
* Su búsqueda de información se hace en O(n) para el peor caso, pero en promedio es O(1). Para nuestra implementación es poco probable que se de el peor caso
* The information search is done in O(n) for the worst case, but on average it is O(1). For our implementation, the worst case is unlikely to occur.

1. Undo Actions:

3.1 Stack Linked List

* Implementing Stack as a Linked List allows to perform the data extraction and insertion processes in O(1).
* Linked List for Stack uses linked nodes where each element has a reference to the next, and for this particular case the “head” atrbutte will always be pointing at the last inserted element.
* To change the “head” each time a node is inserted, an auxiliary node is used to maintain the order of the list and thus perform its operations in O(1).
* Since they are linked nodes, they do not occupy too much memory space.

3.2 Stack ArrayList

* Implementing stack as an arraylist allows adding and extracting elements without any order unless these functionalities are limited as is the case of this implementation where only the last inserted data is extracted.
* The data insertion and extraction methods take both O(n) execution time.
* Has additional pre-implemented features that makes easier its use as a stack.

1. Attention queues:

4.1. One priority queue implemented with heap for each unit

* A priority queue, implemented as a heap, for each unit allows the patients to be attended by priority level, placing in the first position the ones with the highest priority so they will be attended first.
* Priority queue can be implemented as a heap thanks to its methods that allow the organization of data according to priority as well as modifying the priorities of all data.
* The priority of all the data is modified so that the list of patients works for those who have a high priority and for those that are going to be attended on a first-come, first-served basis.
* Since there is a priority queue for each unit, the memory space occupied is smaller than the one in the second option and the heap allows to perform operations in O(n log n).

4.2. One priority queue and one regular queue for each unit in order of arrival

* This option allows each priority queue to attend the patients by their priority level and each general queue to attend them on a first-come, first-served basis.
* The extraction and insertion of elements in the queue implemented as an ArrayList takes an execution time of O(n).
* The priority queue sorts the data according to their priorities from highest to lowest and extracts the data with the highest priority.
* The priority queue does not allow an increase on all priorities.

1. Automatic attention:

5.2. Use Timer and TimerTask

* Timer and TimerTask allow to schedule self-executing actions in the code.
* With self-executing actions you can perform the automatic attention of the queues. Timer is the executor of timerTask which is an action previously defined.
* Timer and TimerTask intervals of execution work in milliseconds.
* This option implements a way to stop the timer.

5.3. Use Runnable and ScheduledThreadPoolExecutor

* With Runnable and ScheduledThreadPoolExecutor you can execute actions in the code programmatically based on time.
* It handles the measurement of time in seconds, which are the time intervals in which actions are performed.
* It implements shutdown that allows it to stop the timer in a certain way since it stops receiving new actions to execute, but executes those that are in the queue.

5.4. Use Runnable, ScheduledThreadPoolExecutor and ScheduledExecutorService.

* Allows automatic actions to be performed based on a pre-set time in which they are to be performed.
* Time is handled on a per second basis and represents the time intervals in which predefined actions are performed.
* ScheduledExecutorService gives the possibility to cancel the entire task thread so you can safely stop the timer and restart it when restarting the task thread.

**Phase 5**

**Evaluation Criteria:**

To find the best solution for each module, it is necessary to have an evaluation system that assigns a value to each option according to the fulfillment of measurable criteria. To achieve this, we established three criteria were each one has three possible values (1,2,3) to evaluate its compliance:

1. that is efficient.
2. That it is not difficult to implement.
3. That is maintainable.

**Evaluación de propuestas por módulo:**

1. **Load and save:**

|  | **Criteria 1** | **Criteria 2** | **Criteria 3** | **Total** |
| --- | --- | --- | --- | --- |
| 1.1 Json | **3** | **3** | **3** | **9** |
| 1.2 Txt | **3** | **2** | **3** | **8** |

1. **Database implementation:**

|  | **Criteria 1** | **Criteria 2** | **Criteria 3** | **Total** |
| --- | --- | --- | --- | --- |
| 2.2. BST | **2** | **1** | **3** | **6** |
| 2.3. Hash Table | **3** | **2** | **1** | **6** |
| 2.4. Chained Hash Table | **3** | **2** | **3** | **8** |

1. **Undo Actions:**

|  | **Criteria 1** | **Criteria 2** | **Criteria 3** | **Total** |
| --- | --- | --- | --- | --- |
| 3.1 Stack Linked List | **3** | **2** | **3** | **8** |
| 3.2 Stack ArrayList | **2** | **3** | **3** | **8** |

1. **Attention queues:**

|  | **Criteria 1** | **Criteria 2** | **Criteria 3** | **Total** |
| --- | --- | --- | --- | --- |
| 4.1. One priority queue implemented with heap for each unit | **3** | **3** | **3** | **9** |
| 4.2. One priority queue and one regular queue for each unit in order of arrival | **2** | **2** | **3** | **8** |

1. **Automatic attention:**

|  | **Criteria 1** | **Criteria 2** | **Criteria 3** | **Total** |
| --- | --- | --- | --- | --- |
| 5.2. Use Timer and TimerTask | **3** | **2** | **1** | **6** |
| 5.3. Use Runnable and ScheduledThreadPoolExecutor | **3** | **1** | **3** | **7** |
| 5.4. Use Runnable, ScheduledThreadPoolExecutor and ScheduledExecutorService | **3** | **3** | **3** | **9** |

**Finally we decided to go with:**

* Json format to load and save since we have more experience with it.
* Chained Hash table since a basic hash table could run out of space and a BST would be slow on searching side by side while the hash function returns the index immediately, comparing an O(lg(n)) for BST against O(1) of a hash table for searching.
* For the undo action, we choose ArrayList even though we know Linked List is more efficient since it doesn’t have to clone the actual structure to increase or decrease its size, we have much more experience with ArrayList, so we took the last word.
* One priority queue since it was just two queues, one for each unit, instead of the four that we would need to implement 4.2, and to prevent patients without a priority to not be attendant, every time a patient get out of the queue, the priorities of the ones who are already waiting will increase in one unit
* ScheduledExecutorService for the auto attention, since it allows us to use the cancel method which stops the timer, while the shutdown method from 5.3 doesn’t cancel the running task so we couldn’t stop the timer instantly. And Timer just receives periods and delays given in milliseconds, and both TimerTask and Timer are deprecated.