

Engineering-Methode made by Juan Diego Rosero, Sara Cardona and David Vergara

*Phase 1: Identification of the problem:*

A health care institution requires a database that allows to manage (search, register, or also undo) the admission and discharge of patients in a clinical laboratory, these patients will be stored in a database where their complete information and their records on admission to the laboratory will be stored, to know if there are priority patients.

Attention depends on the patient, if he/she is a priority and has one of these characteristics: they have an important underlying disease, are elderly, pregnant, among other data that depend on the laboratory; and there is the patient with general access who does not have any characteristic that represents a priority.

Finally, the system to be created will also need a panel to always control the flow of people in the laboratory, it means, the list of people currently in the laboratory, the order of care and the exit of patients, to continue with the other patients who are still in the queue.

*Phase 2: Gathering of the necessary information:*

The information we need to meet the needs of the company that hired our services would be basically 5 requirements that must have the system that we are going to create, among which we find:

*R1 -> Create database:* We need a database to store patient information, there are characteristics which will be stored in the patient information, which will allow us to know if the patient is a priority or not.

*R2 -> Manage database:* It is estimated that databases can have many patients, the laboratory needs to add patients with their information, search for a patient or patients that meet certain parameters and finally undo a patient.

*R3 -> Check in and out:* The system will register the admission of a new patient to the clinical laboratory and will also register the discharge of these patients.

*R4 -> Patient identification:* The system needs to fulfill the need to identify if a patient meets one of the characteristics to be part of the "priority", otherwise it will be part of the general access group.

*R5 -> Monitoring:* Finally, the application will have a people monitoring, it means a list where the list of current people, people in queue and people who have left the laboratory will be reported.

|  |  |
| --- | --- |
| Customer | Institution |
| User | ● |
| Functional requirements | * R1: Create database * R2: Manage database * R3: Control entry and exit * R4: Patient identification * R5: Monitoring |
| Context of the problem | A health care institution requires a database that allows to manage (search, register, or also undo) the admission and discharge of patients in a clinical laboratory, these patients will be stored in a database where their complete information and their records on admission to the laboratory will be stored, in order to know if there are priority patients. |
| Non-functional requirements | * That the program is made in java |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R1: Create database | | |
| Summary | We need a database to store patient information, there are characteristics which will be stored in the patient information, which will allow us to know if the patient is a priority or not. | | |
| Tickets | **Name of entry** | **Data type** | **Selection or repetition condition** |
| Patient information | String |  |
| General activities necessary to obtain the results | 1. Read the information 2. Save information | | |
| Result or Postcondition | Saving patient information within the program | | |

|  |  |
| --- | --- |
| Name or identifier | R2: Manage database |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | |
| Summary | It is estimated that the databases may have many patients, the laboratory needs to add patients with their information, search for a patient or patients that meet certain parameters and finally undo a patient. | | |
| Tickets | **Name of entry** | **Data type** | **Selection or repetition condition** |
| data | String |  |
| General activities necessary to obtain the results | 1. Have registered a patient 2. To have the patient's class 3. Know if it is priority or general access | | |
| Result or Postcondition | Having the patient in the system | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R3: Control entry and exit | | |
| Summary | The system will register the admission of a new patient to the clinical laboratory and will also register their discharge. | | |
| Tickets | **Name of entry** | **Data type** | **Selection or repetition condition** |
| number | int | Option in the menu |
| General activities necessary to obtain the results | 1. Have patients in queue 2. Previously registered patients 3. Have selected the option to view in the menu | | |
| Result or Postcondition | Display the queue on the screen | | |
| Exits | **Output name** | **Data type** | **Selection or repetition condition** |
| Name | String | If the patient exists |
| Id | String |
| Genre | String |

|  |  |  |  |
| --- | --- | --- | --- |
|  | phone | String |  |
| priority | Boolean |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R4: patient identification | | |
| Summary | The system needs to meet the need to identify whether a patient meets one of the characteristics to be part of the "priority", otherwise it will be part of the general access group. | | |
| Tickets | **Name of entry** | **Data type** | **Selection or repetition condition** |
| priority | String | Option in the menu |
| Patient | Patient<T> | Exist previously or be made a patient at that time |
| Shift | int | Have available shifts |
| General activities necessary to obtain the results | 1. Knowing which patients are a priority 2. Have a queue for priority and general access patients | | |
| Result or Postcondition | Separate priority patients from general access | | |
| Exits | **Output name** | **Data type** | **Selection or repetition condition** |
| Patient | Patient<T> | Have successfully registered the patient |
| priority | Boolean |
| Shift | int |
|  |  |

|  |  |
| --- | --- |
| Name or identifier | R5: Monitoring |
| Summary | Finally, the application will have a people monitoring, it means, a list where the list of current people, people in queue and those who have left the laboratory will be reported. |

|  |  |  |  |
| --- | --- | --- | --- |
| Tickets | **Name of entry** | **Data type** | **Selection or repetition condition** |
| number | int | Option in the menu |
| General activities necessary to obtain the results | 1. Have patients in queue 2. Previously registered patients 3. Have selected the option to view in the menu | | |
| Result or Postcondition | Display the queue on the screen | | |
| Exits | **Output name** | **Data type** | **Selection or repetition condition** |
| Cola | String | If the patient exists |
| numberOfPeople | int |
|  | | |

*Phase 3: Search for creative solutions:*

To solve this problem, we need an algorithm capable of fulfilling the requirements, this also requires a code made in java that in turn requires the use of data structures or TAD's, which allow the correct operation of the application.

As a list of attributes that go according to the problem, we have the attributes of the patient that consist of name, age, id, gender, telephone and if it is a priority.

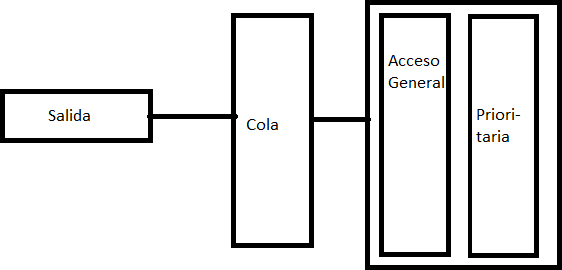
As connections we have that there must be places where patients are stored to know their information which must have a unique access key for each patient. You can add and delete patients, as well as see which patients are in queue in the system.

*Alternative 1:*

We need a system to fulfill the 5 requirements, this makes us keep in mind that in each requirement it is important to consider a method to solve the problem being carried out.

The system requires a code capable of solving the above mentioned, therefore it must have specific characteristics, such as: queues and hips, because we need to organize the information and/or eliminate objects that are part of the queues.

This system requires elements (which would be the patients), connections and interactions between them, that the elements are stored in a place and that these elements have connections between them to know if there is an element that is a priority or not, all this information will be stored in the queues. Finally, we will use the hip to eliminate the patients that the user wants.



This is a graphical representation of how we could see how the system would work, the output would be what the user could observe, and we have a database where is the general access and priority, this database goes through a method that will print the "final" queue where you will see the order of priority and general access patients.

*Alternative 2:*

We know the requirements we need to carry out the code, these requirements are 5. Of these we need to analyze one by one and then make a flowchart representing the information we collected and analyzed.

For the first 2 requirements we need a database, this database obviously serves to store the information of the people, but to manage the information of these people and to know if there are patients with priority or not, we need two queues, one for general access and the other for priority. These queues will help us to organize the information and evaluate it to fulfill the objective of providing the required care service.

For requirement 3 we can use a method within the code that helps us to monitor the people entering and leaving, likewise for requirement 5 we use a method to monitor the patients in queue and/or who are priority to have a better visualization of the customer service they are providing.

Finally, requirement 4 is to store the people with their respective data, which will help us to organize or place them in the queues.

*Alternative 3:*

The best in this type of cases where the important thing is to work with an object, which in this case would be a patient, and at the same time to be able to monitor it is to use a data structure, the data

structures as we already know allow us not only to organize the information but also to store it and to be able to distribute it in a correct way.

Two things will be considered, firstly the user interface that will be used to provide greater comfort to the person using the service, and secondly a java code will be implemented containing the methods and data structures that will be used to fulfill the requirements. This user interface allows to adapt to the code that will be made in java, thus achieving to show the information required by the users.

For the java code as mentioned above, data structures are needed to manage, store and organize patient information, so it is necessary to use the following structures:

* Hash table: we need to use hash tables to be able to save the patient with a unique ID, this would solve the problem of saving each patient as unique.
* Priority queues: for priority patients
* General access queues: for normal patients

Finally, executing and connecting these data structures allows the elements (patients) of the code to be placed in stacks and queues, these elements in turn will have a key to access the information of the element, this would solve the problem of being able to delete a single element. Finally, this code has a user interface to indicate the monitored information, where we will find inputs, outputs and the option to add or delete an element.

*Alternative 4:*

To provide a clear solution to the problem, it is important to develop a code with data structures that allow the organization of patient information both internally in the system and for the user interface that will carry out the records. Within the system it is necessary to distribute the information from a scanner that will receive the information so that it can be processed within the program.

The idea on how the information will be treated is as mentioned above. The data structures we use are distributed as follows, with the following conditions:

* Patient object: We have an object that is going to be saved with each patient registered with their respective information, and this a key factor to save each patient object is whether this is priority or not.
* Queues and stacks: The patient will be stored inside a queue.
* Hash Table: This data structure is used to store the patient and its attributes as unique with a unique id(key).
* Arraylist: We store the patients inside a file or the file that would be our database of patients.

Now that we have the structure of the system, we only need to execute it with the respective procedures, from registering the user to save it within a data structure to search and delete it.

*Phase 4. Transition from Ideas to Preliminary Designs*

To proceed to select the best option we need to analyze if the alternatives meet the required objective, having a correct procedure to execute it.

*Alternative 1:*

For the first alternative we have a data structure that can help us store patient information, however, it is not consistent with the procedure for organizing and distributing the information, since there is confusion in placing priority and general access patients.

*Alternative 2:*

Alternative 2 gives us a very good point to separate the patient information and to be able to organize it in the code that is going to be done, the problem with alternative 2 is that unlike the others it is very extensive in how it wants to solve the problem.

*Alternative 3:*

Alternative 3 gives us a good generalization and specification of the problem, so consider using data structures that allow us to gather the elements needed to execute the necessary processes.

This alternative has the data structures that allow us to separate priority and non-priority patients, as well as the implementation of Hash table to search for each of the saved patients.

*Alternative 4:*

Finally, there is alternative 4, which consists of several aspects that we can highlight, among which we find:

* Patients
* HashTable
* Tails
* ArrayList

Alternative four consists of these three elements that, when interacting, allow the program to function, which leads us to conclude that there is an order in how to execute the procedures to fulfill each requirement.

It should be noted that patients have different attributes, so each patient is different, knowing this, alternative 4 also has a solution to both queue patients, as well as to save and execute actions with them.

*Phase 5: evaluation and selection of the best solution*

The criteria that we can evaluate to carry out the choice of the correct selection, in order to collect the necessary criteria, it is important to evaluate the requirements, where we can find:

* Criterion A: keeping patient
  1. Partial
  2. All specific data
* Criterion B: Queuing the patient [1]Prioritize patient

[2]Set general access patient

[4]Separating patients according to priority [3]Include both patients in the same queue.

* Criterion C: Manage patients (include, remove) [1]Successfully separating patients

[2] Place patients in queues

[4]Performing methods in the code to manage patients [3]Partial implementation of patients

* Criterion D: Ease of algorithmic implementation [1]Compatible with required data structures [2]Not compatible with data structures

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Criterion A** | **Criterion B** | **Criterion C** | **Criterion D** | **Total** |
| **Alternative 3** | All specific data | Include both patients in the same queue | Partial implementation of patients | Compatible with required data structures  1 |  |
| **Alternative 4** | All specific data | Separate patients according to priority | Perform methods in the code to manage patients | Compatible with required data structures  1 |  |

*Selection:*

According to what we have seen above, alternative 3 has many things implemented that would be useful to make the code, however, alternative 4 is the most complete, not only separating the patients in a correct way but also managing them with data structures that would be implemented in a correct way.

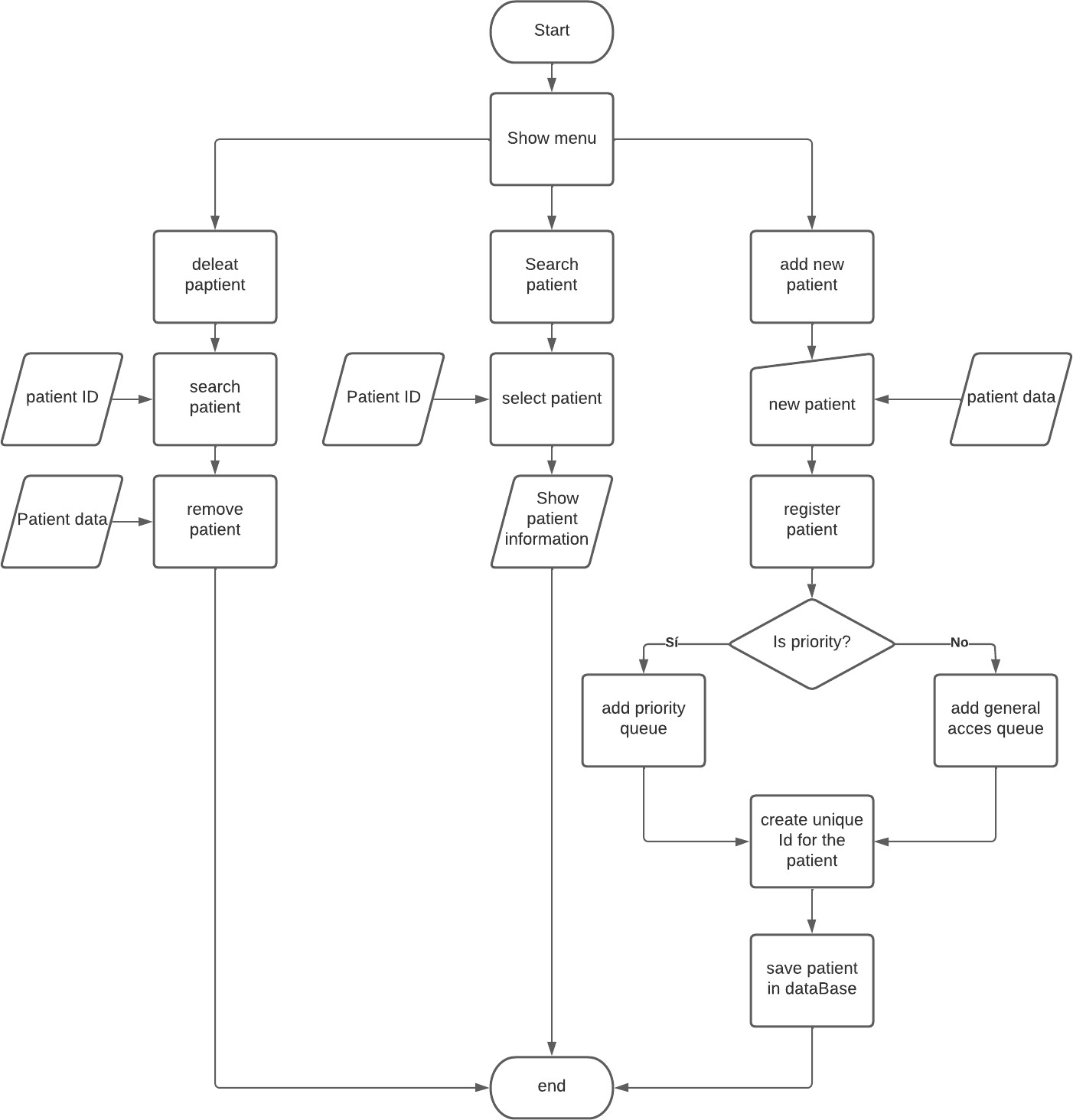
*Preparation of Reports and Specifications Specification of the problem:*

*Problem:* Manage the patients of a clinic, which need to know if they are priority or not, the way to manage them is to enter them, delete them and be able to see who is in queue to be attended.

*Inputs*: Patients and specifications of their attributes. *Outputs:* Monitoring of patients and their status for care. *Considerations:*

1. Know if patients are a priority
2. Monitoring of patients in queue
3. Patient administration

*Flow chart:*

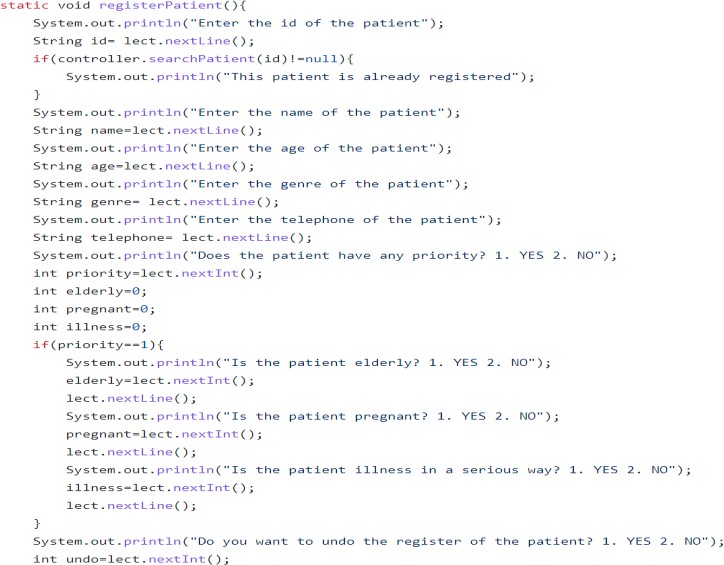


*Step 7: Design implementation*

Java implementation.

List of tasks to be implemented:

* 1. Register patient
  2. Patient Search
  3. Enter patient
  4. Remove patient
  5. Read data
  6. Write in data base
  7. Patient who has been registered
  8. Remove from hash table
  9. Show patients
  10. Eliminate patient with key
  11. Insert in hash table
  12. Add ultimo in the hash table
  13. Search hash table

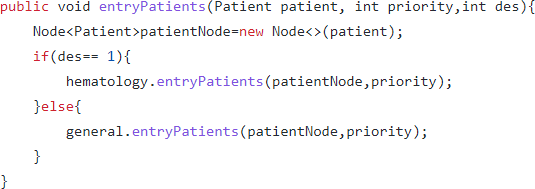
Register patient

|  |  |
| --- | --- |
| Name: | registerPatient |
| Description: | Register a patient in the system |
| Entrance: | * Name: String * Age: String * Genre: String * Telephone: String * Priority:int |
| Return: | Patient |

Patient Search

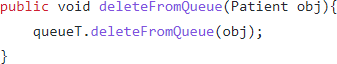
|  |  |
| --- | --- |
| Name: | searchPateint |
| Description: | Search for a patient in the system |
| Entrance: | - Id : String |
| Return: | Message |

Patient Login



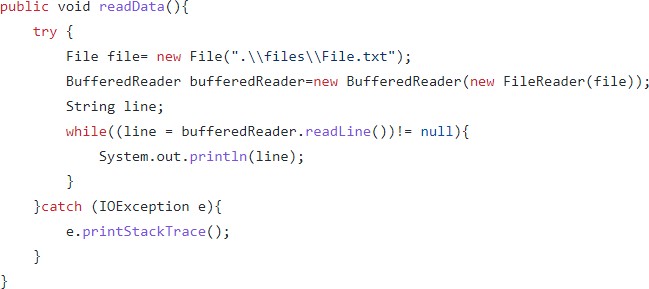
|  |  |
| --- | --- |
| Name: | EntryPateint |
| Description: | Putting a patient in the queue |
| Entrance: | * Patient : Patient * Priority : int * Des : int |
| Return: | void |

Remove patient



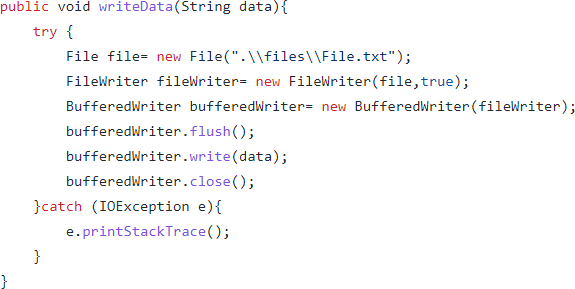
|  |  |
| --- | --- |
| Name: | deletefromQueue |
| Description: | Remove a patient from the queue |
| Entrance: | - Patient: obj |
| Return: | void |

Read Data



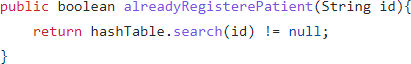
|  |  |
| --- | --- |
| Name: | readData |
| Description: | Read the database being saved |
| Entrance: | - |
| Return: | void |

Write Data



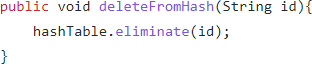
|  |  |
| --- | --- |
| Name: | WriteData |
| Description: | Write to the database that you have created |
| Entrance: | - |
| Return: | void |

Patient who has been registered

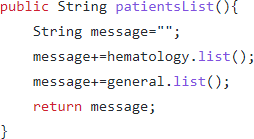


|  |  |
| --- | --- |
| Name: | alreadyRegisterPatient |
| Description: | Know if a patient has already been registered in the database. |
| Entrance: | - Id : String |
| Return: | boolean |

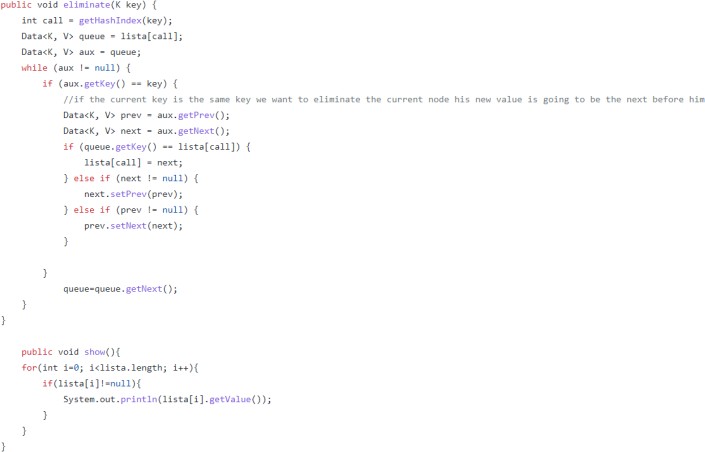
Remove from hash table



|  |  |
| --- | --- |
| Name: | deleteFromHash |
| Description: | Delete patient from the table with his id containing his information |
| Entrance: | - Id : String |
| Return: | void |

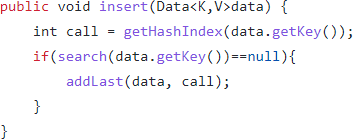
Show patients

|  |  |
| --- | --- |
| Name: | patientList |
| Description: | Show all patients in queue |
| Entrance: |  |
| Return: | message |

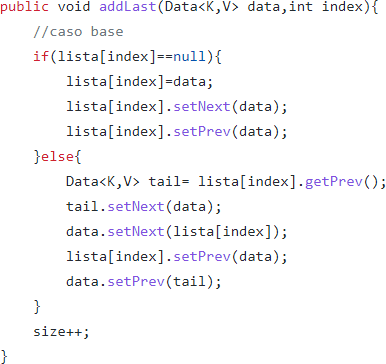
Eliminate patient with key

|  |  |
| --- | --- |
| Name: | eliminate |
| Description: | Remove patient from hash table |
| Entrance: |  |
| Return: | void |

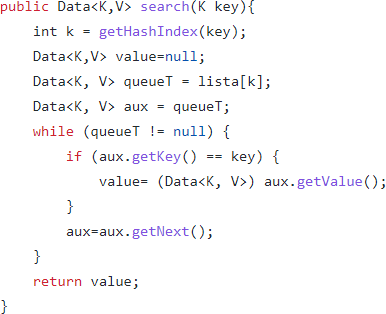
Insert in hash table



|  |  |
| --- | --- |
| Name: | insert |
| Description: | Insert in hash table |
| Entrance: | - Data< K,V> : data |
| Return: | void |

Add ultimo in the hash table

|  |  |
| --- | --- |
| Name: | addLast |
| Description: | Add the last one in the hash table |
| Entrance: | * Data< K,V> : data * Index : int |
| Return: | void |

Search hash table

|  |  |
| --- | --- |
| Name: | search |
| Description: | Search hash table |
| Entrance: | - K : key |
| Return: | Data < K,V> value |