Kevin Vincent Loachamin Almeida A00382106

Luisa Fernanda Castaño Pino A00380290

Manuel Alejandro Herrera Uribe A00381987

**TI 1 Hospital CyEd l**

**Engineering method:**

**Phase 1: Problem identification**

There is a problem for a health care institution, this requires a process to give shifts to patients, and thus direct them to their respective laboratories to be attended depending on their priority, and to perform the admission and discharge of patients, this is given through a database, this will have the information of patients, according to the order, also with this information can be searched, recorded and undo, taking into account the type of care.

As the possible error of the person in charge of doing this process, it is also needed a way to monitor the list of people who will be in this process, that is, who are in the laboratory, also report the order of care of people, in addition to reporting the output and thus be able to continue with the process required by the institution.

The necessary requirements for the institution are:

| Client | Hospital |
| --- | --- |
| User | Receptionists |
| Functional requirements | R1. system for entering and leaving the institution.  R2. system of shifts assigned at check-in time.  R3. "undo" function to avoid human error.  R4. priority system for certain people.  R5. Read and write databases with customer information.  R6. Panel with the information of people and their shifts within a unit. |
| Context | A hospital that requires a mechanism which provides a correct, agile and easy use of the data entered from the patients and which allows it to be managed by priority as well. |
| Non-functional requirements | Adequate operability, showing instructions on the correct way to induce the information. |

**Phase 2: Collection of necessary information:**

From the health care provider institution presents us with the type of information as the specification of the previously established requirements, so we have:

**R1**. system of entry and exit of the institution: the institution, as presented in the problem, requires the efficient entry and exit of patients to and from the laboratories.

**R2**. system of shifts assigned at the time of admission: as it requires that shifts be established according to the criteria if it meets priority or not, otherwise it will be attended on a first-come, first-served basis.

**R3**. Priority system for certain people: the document does not provide the information on which it is determined which people have priority or not, in this case priority is given when the person is pregnant, elderly or has a basic illness.

**R4**. Read and write databases with customer information: we are told that an initial data upload is needed as this will present all the necessary information of the registered persons.

**R5.** Panel with the information of the people and their shifts within a unit: finally, it is required to monitor the process of the people registered, i.e. those who are currently in any of the laboratories and the priority for this person, so it would also be possible to see the order of attention of this person.

**Phase 3: Search for creative solutions:**

Brainstorming is one of the most popular techniques to solve the type of problems that need to be solved as a group, in addition to actively stimulating the generation of ideas even if they are not possible, as this minimizes the possibility of negative emotional state among participants and thus contributes as a team to find the solution to the problem.

Through brainstorming, the following ideas were generated:

1. Shift management system by means of a dispenser of these, which are ordered from lowest to highest and in this order is that people received their turn.
2. Attention designated by a person at the reception desk who is in charge of establishing the order of priority and directing people to their corresponding laboratories.
3. A program that allows, by means of a computer, to facilitate the process of admittance, registration, direction and attention of people.
4. Request medical attention via telephone call with an advisor of the health care institution.
5. Requesting medical care through self-generated chats on communication networks such as WhatsApp, which is then attended to by a staff member of the health care institution.
6. Requesting medical attention via e-mail, which will then be attended by authorized personnel.

**Phase 4: Preliminary design development**

Given the brainstorming, we will proceed to choose one of them and define the reason why the other options are not valid or do not solve the problem correctly and efficiently.

**Idea:**

1. A system for the administration of shifts by means of a shift dispenser, which are ordered from smallest to largest and in this order is that people received their turn.

This idea does not provide an efficient solution to the problem posed, since it would not be providing a solution to one of the requirements of the health care institution, such as assigning shifts depending on the priority of the person, since this option would not give priority to the elderly, pregnant women and people with basic illnesses, nor would they be directed to the corresponding laboratory.

**Idea:**

1. Attention designated by a person at the reception desk who is in charge of establishing the order of priority and directing people to their corresponding laboratories.

In this option it would not be done efficiently and given this it is not a good idea, because being only one person in charge of this option and handling a large number of individuals is even more likely to fall into human error and thus direct people to laboratories not corresponding to their situation or requirements, in addition to making the process slower and more difficult to work with.

**Idea:**

1. A program managed by the reception staff that allows, by means of a computer, to facilitate the process of check-in, check-out, registration, laboratory assignment and attention of people depending on their priority or order of arrival.

This idea is the best option, because it meets the requirements established in the problem statement, since it will allow to carry out in an agile and fast way all this process of entry, exit, the registration of people attended by a person but saved in the database provided by the program, in addition to determining the priority of the attention of people and thus generate turn of attention, so to meet the requirements makes it the right choice for the solution of this problem.

**Idea:**

1. Request medical attention via telephone call with a counselor of the health care institution.

This not being attended in person may hinder communication between the person who will take the data to the user, thus hindering the proper collection of them and thus can not be solved correctly the requirement to establish the shifts given a priority, in addition to the attention could be delayed more by this means taking into account a high flow of calls.

**Idea:**

1. Requesting medical care through self-generated chats on communication networks such as WhatsApp, which is then attended by some staff of the health care provider institution.

As in the previous idea, it will take time to assign a shift, register and enter, and it may contain errors because the information required for the fulfillment of the requirements may not be obtained.

**Idea:**

1. Request for medical attention by e-mail, which will then be attended by authorized personnel.

This of the options to register information by means of communication can be the most problematic because it can consist of many errors at the time that the user submits his information by not having alerts that mention what is and is not required, in addition it would take longer for the assignment and registration because it would not be done by a machine or program, it will be done by a person and to be a high flow of mail can be lost and therefore leave some people without medical care.

**Phase 5: Evaluation**

Se establece el criterio de selección:

**Criterio A. Precisión de la solución.** ­

[2] Exacta

­[1] Aproximada ­

**Criterio B. Eficiencia.**

[3] Exacta

­[2] Aproximada ­

[1] Nula­

**Criterio C. Completitud.**

[3] Exacta

­[2] Aproximada ­

[1] Nula­

**Criterio D. Facilidad en implementación algorítmica: ­**

[2] Compatible con las operaciones aritméticas básicas de un equipo de cómputo moderno ­

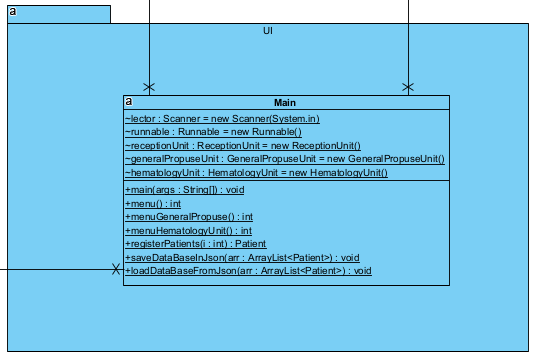
[1] No compatible completamente con las operaciones aritméticas básicas de un equipo de cómputo moderno

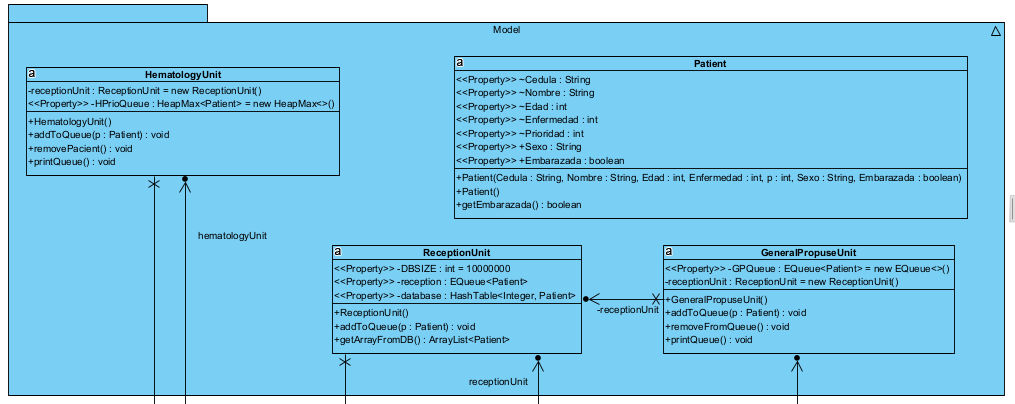
|  | Criterio A | Criterio B | Criterio C | Criterio D | Total |
| --- | --- | --- | --- | --- | --- |
| Idea 2 | [1] | [2] | [1] | [1] | 4 |
| Idea 3 | [2] | [3] | [3] | [2] | 10 |

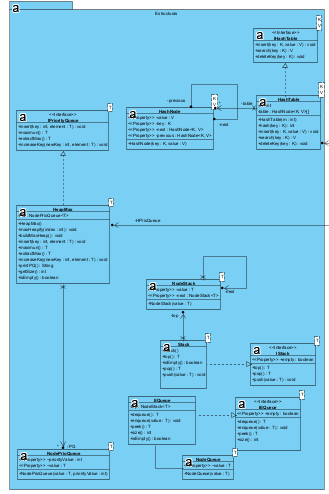
De acuerdo con lo anterior, se logra establecer que la solución ideal para dar respuesta al problema planteado es la idea 3, teniendo la idea seleccionada, se realiza un diagrama presentando como se debería ver implementado la solución lo cual es ideal pues permite ordenar de forma estructurada posibles métodos de código como solución y sus conecciones con las demás clases para establecer cómo funciona el programa.

Por medio de la implementación del código, se realizarán los casos de prueba con los datos aleatorios al principio solo para probar que el programa responde dando solución correcta y eficiente a los requerimientos, y alertando de posibles errores de ejecución y poder recibir la información correcta.

**DESIGN**

****

****

****

**(For better visualization enter the file DiagramaT1C.vpp)**

**SPECIFICATION REQUIREMENTS**

| NAME OR IDENTIFIER | R1. The program must have a system for entering and leaving the institution. | | |
| --- | --- | --- | --- |
| SUMMARY | The system shall be able to register the entrance to a clinical laboratory and the exit of a person from this laboratory. | | |
| INPUTS | Input name | Data type | Selection or repetition condition |
| nameOption | String | **-----------** |
| idOption | String | **-----------** |
| gnerOption | String | **-----------** |
| pregnantOption | boolean | **-----------** |
| ageOption | int | **-----------** |
| illOption | int | **-----------** |
| priority | int | **-----------** |
| GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS | Enter the person's data so that it is finally registered, and in this way be able to search for the person and check out when the person leaves. | | |
| RESULT | The data will be entered and saved for further processing after this. | | |
| OUTPUT | Output name | Data type | Selection or repetition condition |
| Patient | Object | **----------** |

| NAME OR IDENTIFIER | R2. The program must manage a system of assigned shifts. | | |
| --- | --- | --- | --- |
| SUMMARY | People should be able to be assigned turns to go to the laboratories at the time of admission. | | |
| INPUTS | Input name | Data type | Selection or repetition condition |
| database | Integer, Patient | **-------------** |
| DBSIZE | int | **-------------** |
| reception | EQueue<Patient> | **-------------** |
| GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS | This process takes place thanks to a previous registration of the patient, in order to be able to assign shifts to these patients after their admission. | | |
| RESULT | People will have a shift so that they can be given effective care. | | |
| OUTPUT | Output name | Data type | Selection or repetition condition |
| message | String | **-------------** |

| NAME OR IDENTIFIER | R3. priority system for certain people. | | |
| --- | --- | --- | --- |
| SUMMARY | It should generate priority for patients depending on specified cases. | | |
| INPUTS | Input name | Data type | Selection or repetition condition |
| pregnantOption | boolean | **---------** |
| ageOption | int | **---------** |
| illOption | int | **---------** |
| GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS | Thanks to the information entered by the patients, a validation of these data is carried out in order to establish the priority of the patients. | | |
| RESULT | People will be given priority according to their conditions. | | |
| OUTPUT | Output name | Data type | Selection or repetition condition |
| priority | int | **---------** |

| NAME OR IDENTIFIER | R4.Read and write databases with customer information. | | |
| --- | --- | --- | --- |
| SUMMARY | We are told that an initial data upload is required as this will present all the necessary information of the registered persons. | | |
| INPUTS | Input name | Data type | Selection or repetition condition |
| receptionUnit | ReceptionUnit | **-------------** |
| GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS | The information of the individuals will be recorded and stored in a database to be able to access this information in later situations. | | |
| RESULT | The information will be stored in the database. | | |
| OUTPUT | Output name | Data type | Selection or repetition condition |
| database | Integer, Patient | **----------** |

| NAME OR IDENTIFIER | R5. Panel with information about people and their shifts within a unit. | | |
| --- | --- | --- | --- |
| SUMMARY | It is required to monitor the process of registered persons, i.e. those who are currently in any of the laboratories and the priority for this person, so it would also be possible to see the order of attention of this person. | | |
| INPUTS | Input name | Data type | Selection or repetition condition |
| database | Integer, Patient | **----------** |
| GENERAL ACTIVITIES NECESSARY TO OBTAIN THE RESULTS | This process receives and uses the information of the users of the health care institution stored in the database in order to be shown to the public. | | |
| RESULT | The user is presented with the necessary or required information on the screen. | | |
| OUTPUT | Output name | Data type | Selection or repetition condition |
| message | String | **----------** |

**Time complexity analysis:**

**Register method:**

public static Patient registerPatients(int i) {

String nameOption; //1

String idOption; //1

String gnerOption; //1

boolean pregnantOption; //1

int ageOption; //1

int illOption; //1

int priority; //1

String auxName = regPatientName(); //1

if (auxName == "exit") { //1

return null; //1

} else {

nameOption = auxName; //1

}

String auxID = regPatientID(); //1

while (auxID.equals("UNDO")) { //n

if (undo.pop().equals("regID")) { //n+1

auxName = regPatientName(); //1

if (auxName.equals("exit")) { //n+1

return null; //1

}

auxID = regPatientID(); //1

}

}

String auxGner = regPatientGner(); //1

while (auxGner.equals("UNDO")) { //n+1

if (undo.pop().equals("regGner")) { //n

auxID = regPatientID(); //n

if (auxID.equals("UNDO")) { //n

auxName = regPatientName(); //n

if (auxName.equals("exit")) { //n

return null; //n

}

auxID = regPatientID(); //n

}

auxGner = regPatientGner(); //n

}

}

String auxPreg = ""; //1

if (auxGner.equals("1")){ //1

auxPreg = regPatientPregnant(); //1

while (auxPreg.equals("UNDO")){ //n+1

if (undo.pop().equals("regPreg")){ //n

auxGner = regPatientGner(); //n

if (auxGner.equals("UNDO")){ //n

auxID = regPatientID(); //n

if (auxID.equals("UNDO")){ //n

auxName = regPatientName(); //n

if (auxName.equals("exit")){ //n

return null; //n

}

auxID = regPatientID(); //n

}

auxGner = regPatientGner(); //n

if (auxGner == "2"){ //n

break;

}

}

auxPreg = regPatientPregnant(); //n

}

}

}

String auxAge = regPatientAge(); //1

while (auxAge.equals("UNDO")){ //n+1

if (undo.pop().equals("regAge")){ //n

if (auxGner.equals("1")){ //n

auxPreg = regPatientPregnant(); //n

if (auxPreg.equals("UNDO")){ //n

auxGner = regPatientGner(); //n

if (auxGner.equals("UNDO")){ //n

auxID = regPatientID(); //n

if (auxID.equals("UNDO")){ //n

auxName = regPatientName(); //1

if (auxName.equals("exit")){ //n

return null; //n

}

auxID = regPatientID(); //n

}

auxGner = regPatientGner(); //n

}

if (auxGner.equals("1")){ //n

regPatientPregnant(); //n

}

}

auxAge = regPatientAge(); //n

}else {

auxGner = regPatientGner(); //n

if (auxGner.equals("UNDO")){ //n

auxID = regPatientID(); //n

if (auxID.equals("UNDO")){ //n

auxName = regPatientName(); //n

if (auxName.equals("exit")){ //n

return null;

}

auxID = regPatientID(); //n

}

auxGner = regPatientGner(); //n

}

auxAge = regPatientAge(); //n

}

}

}

String auxIll = regPatientIll(); //1

while(auxIll.equals("UNDO")){ //n+1

if (undo.pop().equals("regIll")){ //n

auxAge = regPatientAge(); //n

if (auxAge.equals("UNDO")){ //n

if (auxGner.equals("1")){ //n

auxPreg = regPatientPregnant(); //n

if (auxPreg.equals("UNDO")){ //n

auxGner = regPatientGner(); //n

if (auxGner.equals("UNDO")){ //n

auxID = regPatientID(); //n

if (auxID.equals("UNDO")){ //n

auxName = regPatientName(); //n

if (auxName.equals("exit")){ //n

return null;

}

auxID = regPatientID(); //n

}

auxGner = regPatientGner(); //n

}

auxPreg = regPatientPregnant(); //n

}

}else{

auxGner = regPatientGner(); //n

if (auxGner.equals("UNDO")){ //n

auxID = regPatientID(); //n

if (auxID.equals("UNDO")){ //n

auxName = regPatientName(); //n

if (auxName.equals("exit")){ //n

return null;

}

auxID = regPatientID(); //n

}

auxGner = regPatientGner(); //n

}

}

auxAge = regPatientAge(); //n

}

auxIll = regPatientIll(); //n

}

}

nameOption = auxName; //1

idOption = auxID; //1

if (Integer.parseInt(auxGner)==1){ //1

gnerOption = "Women"; //1

}else{

gnerOption = "Man"; //1

}

if (Integer.parseInt(auxPreg)==1){ //1

pregnantOption = true; //1

} else{

pregnantOption = false; //1

}

ageOption = Integer.parseInt(auxAge); //1

illOption = Integer.parseInt(auxIll); //1

if (ageOption >= 65 && pregnantOption == true && illOption == 1) { //1

priority = 6; //1

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the hematology row."); //1

} else if (ageOption >= 65 && pregnantOption == true && illOption == 2) { //1

priority = 4;

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the hematology row."); //1

} else if (ageOption >= 65 && pregnantOption == false && illOption == 1) {

priority = 3; //1

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the hematology row."); //1

} else if (ageOption <= 65 && pregnantOption == true && illOption == 1) {

priority = 3; //1

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the hematology row."); //1

} else if (ageOption <= 65 && pregnantOption == false && illOption == 1) {

priority = 1; //1

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the hematology row.");

} else if (pregnantOption == true && illOption == 1 && ageOption <= 65) {

priority = 3; //1

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the hematology row."); //1

} else if (pregnantOption == false && illOption == 2 && ageOption <= 65) {

priority = 1; //1

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the main row.");//1

} else if (pregnantOption == true && illOption == 2 && ageOption <= 65) { //1

priority = 2; //1

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the hematology row.");

} else {

priority = 0; //1

JOptionPane.showMessageDialog(null,"The patient " + (i) + " is on the main row.");//1

}

return new Patient(idOption,nameOption,ageOption,illOption,priority,gnerOption,pregnantOption);

}

**Total: 76n+51 = O(n)**

**Method IncreasKey from HeapMax:**

public void increaseKey(int newKey, T element) {

int index = -1; //1

for (int i = 0; i < PQ.size(); i++) { //n+1

if (PQ.get(i).getValue() == element) { //1

index = i; //1

}

}

if (newKey < PQ.get(index).getPriorityValue()) { //1

System.out.println("New key is smaller than actual key"); //1

} else {

PQ.get(index).setPriorityValue(newKey); //1

}

while (index > 0 && PQ.get(index / 2).getPriorityValue() < PQ.get(index).getPriorityValue()) { //n+1

NodePrioQueue<T> temp1 = PQ.get(index); //1

NodePrioQueue<T> temp2 = PQ.get(index / 2); //

PQ.set(index, temp2); //1

PQ.set(index / 2, temp1); //

index = index / 2; //

}

}

**Total: 2n+9+3 = O(n)**

**Spatial complexity analysis:**

**Register method:**

| **Tipo** | **Variable** | **Tamaño de 1 valor atómico** | **Cantidad de valores atómicos** |
| --- | --- | --- | --- |
| **Entrada** | i | 32 bits | 1 |
| **Auxiliar** | nameOption | 160 bits | 1 |
| **Auxiliar** | idOption | 128 bits | 1 |
| **Auxiliar** | gnerOption | 160 bits | 1 |
| **Auxiliar** | pregnantOption | 8 bits | 1 |
| **Auxiliar** | ageOption | 32 bits | 1 |
| **Auxiliar** | illOption | 32 bits | 1 |
| **Auxiliar** | priority | 32 bits | 1 |
| **Auxiliar** | auxName | 112 bits | 1 |
| **Auxiliar** | auxID | 80 bits | 1 |
| **Auxiliar** | auxGner | 112 bits | 1 |
| **Auxiliar** | auxPreg | 112 bits | 1 |
| **Salida** | Patient (Object) | 552 bits | n |

**Complejidad Espacial Total =** n+12 = θ(n)

**Complejidad Espacial Auxiliar =** 11 = θ(1)

**Complejidad Espacial Auxiliar + Salida =** 11 + n = θ(n)

**Method IncreasKey from HeapMax:**

| **Tipo** | **Variable** | **Tamaño de 1 valor atómico** | **Cantidad de valores atómicos** |
| --- | --- | --- | --- |
| **Entrada** | newKey | 32 bits | 1 |
| **Entrada** | element | 32 bits | 1 |
| **Auxiliar** | i | 32 bits | 1 |
| **Auxiliar** | index | 32 bits | 1 |
| **Auxiliar** | PQ | 32 bits | n |
| **Auxiliar** | temp1 | 32 bits | 1 |
| **Auxiliar** | temp2 | 32 bits | 1 |
| **Salida** | void | void | void |

**Complejidad Espacial Total =** n+6 = θ(n)

**Complejidad Espacial Auxiliar =** 4= θ(1)

**Complejidad Espacial Auxiliar + Salida =** 4 = θ(1)

**TAD design of data structures used**

| TAD Stack | | |
| --- | --- | --- |
| Abstract object: Stack = | | |
| Invariant: | | |
| Operations: | | |
| Operation | Input | Output |
| Stack |  | Stack |
| Top |  | Element |
| isEmpty |  | Booleano |
| pop |  | Element |
| push | Element | Stack |

| Stack |
| --- |
| < Constructor > |
| \*Generates an empty Stack\* |
| Pre: —------- |
| Pos: |

| Top |
| --- |
| < Analyzer > |
| \*Gets the first value (the top value) in the Stack.\* |
| Pre: |
| Pos: Element |

| isEmpty |
| --- |
| < Analyzer > |
| \*Indicates if the created Stack has objects or if it is empty.\* |
| Pre: Stack s |
| Pos: true if , false if |

| pop |
| --- |
| < Modifier > |
| \*Extracts the newly added element\* |
| Pre: |
| Pos: |

| push |
| --- |
| < Modifier > |
| \*Inserts a new element to the Stack\* |
| Pre: |
| Pos: |

| TAD Queue | | |
| --- | --- | --- |
| Abstract object: | | |
| Invariant: | | |
| Operations: | | |
| Operation | Input | Output |
| Queue |  | Queue |
| dequeue |  | Element |
| enqueue | Element | Queue |
| peek |  | Element |
| size |  | Entero |
| isEmpty |  | Booleano |

| Queue |
| --- |
| < Constructor > |
| \*Generates an empty Queue\* |
| Pre: —------- |
| Pos: |

| dequeue |
| --- |
| < Modifier > |
| \*Extracts the item that first entered the Queue\* |
| Pre: |
| Pos: |

| enqueue |
| --- |
| < Modifier > |
| \*Inserts a new element to the Back of the Queue\* |
| Pre: |
| Pos: |

| peek |
| --- |
| < Analyzer > |
| \*Extracts the element in front of the Queue\* |
| Pre: |
| Pos: |

| size |
| --- |
| < Analyzer > |
| \*Returns the size that the queue has\* |
| Pre: |
| Pos: Int n |

| isEmpty |
| --- |
| < Analyzer > |
| \*Indicates if the created Queue has objects or if it is empty\* |
| Pre: Queue q |
| Pos: true if , false if |

| TAD PriorityQueue | | |
| --- | --- | --- |
| Abstract object: | | |
| Invariant: | | |
| Operations: | | |
| Operation | Input | Output |
| PriorityQueue |  | PriorityQueue |
| Insert | Integer, Element | PriorityQueue |
| extract Maximum |  | Elemento |
| Maximum |  | Elemento |
| Increase Key | Integer, Element | PriorityQueue |

| PriorityQueue |
| --- |
| < Constructor > |
| \*Generates an empty PriorityQueue\* |
| Pre: —------- |
| Pos: |

| Insert |
| --- |
| < Modifier > |
| \*Enters an element to the queue and ordered does a buildMaxHeap\* |
| Pre: |
| Pos: |

| Extract Maximum |
| --- |
| < Modifier > |
| \*Returns the element with the highest priority (element in the first position) and removes it from the PriorityQueue\* |
| Pre: |
| Pos:  and |

| Maximum |
| --- |
| < Analyzer > |
| \*Returns the element with the highest priority in the Queue (element in first position).\* |
| Pre: |
| Pos: |

| Increase Key |
| --- |
| < Modifier > |
| \*Searches for Element e and increments its Key (priority)\* |
| Pre: and Element e |
| Pos: |

| TAD HashTable | | |
| --- | --- | --- |
| Abstract object: | | |
| Invariant: | | |
| Operations: | | |
| Operation | Input | Output |
| HashTable |  | HashTable |
| Insert | Element, Element | HashTable |
| search | Element | Elemento |
| deleteKey | Element | HashTable |

| HashTable |
| --- |
| < Constructor > |
| \*Generates an empty HashTable\* |
| Pre: —------- |
| Pos: |

| Insert |
| --- |
| < Modifier > |
| \*Enter an element e with a key k in the hashTable\* |
| Pre: |
| Pos: |

| search |
| --- |
| < Analyzer > |
| \*searches for and returns the value contained in the key k\* |
| Pre: |
| Pos: Element e |

| delete Key |
| --- |
| < Modifier > |
| \*Searches for and deletes the element(s) contained in the key k\* |
| Pre: |
| Pos: |