|  |  |
| --- | --- |
| Client | Health Provider Institution (IPS) |
| User | * IPS staff with role allowed to use the software |
| Functional requirements | * RF1: Load the data of the users of the IPS. * RF2: Save the data of the users of the IPS. * RF3: Register new patient. * RF4: Search patient. * RF5: Register entry of a patient. * RF6: Register out of a patient. * RF7: Establish the type of patient care. * RF8: Undo an action of entry or out. * RF9: Show panel that allows monitoring patients in the laboratory. |
| Problem Context | to calculate all the required data, it is necessary to know personal information such as full name, date of birth, gender and photo. And labor information such as the date of entry, benefits and salary of the company's employees. |
| No functional requirements | * RNF1: The functionalities of the software must be efficient and effective the more possible. * RNF2: The system must be secure enough to prevent possible improper access by third parties and theft of private information. |

**- RF1: Load the data of the users of the IPS.**

The system must be able to load previously stored IPS user data (in this case, in a flat text file) for use in running the program.

**- RF2: Save the data of the users of the IPS**.

The system must allow new IPS user data to be stored in the same file from which they are loaded, so that they persist even if the system is closed.

**- RF3: Register new patient.**

The system must be able to register a new patient. This, asking the user for the patient's identification number, their name, sex, and their date of birth and if they have priority (pregnant, base disease, etc).

**- RF4: Search patient.**

The system must allow a patient to be searched by their identification number. In the search result, the patient information should be displayed.

**- RF5: Register entry of a patient.**

The system must offer the functionality of registering the entry of a patient to the place with the date and time of that moment. Every time a patient logs in, their last time logged in is updated.

**- RF6: Register out of a patient.**

The system must offer the functionality of registering the discharge of a patient from the place with the date and time of that moment. Every time a patient is discharged, their last discharged time is updated.

**- RF7: Establish the type of patient care.**

The system must allow the user to establish the priority of each of the patients already admitted. The above depending on whether the patient has any important underlying disease, are older adults, are pregnant, etc. If none of these conditions are met, the patient will be seen on a first-come, first-served basis.

**- RF8: Undo an action of entry or out.**

The system must allow the user to undo an entry or exit action that he has previously performed. When using this functionality, a patient's recorded check-in or check-out is deleted. The foregoing in order to correct errors by IPS employees.

**- RF9: Show panel that allows monitoring patients in the laboratory.**

The system must have a panel that allows the list of people currently in the laboratory to be monitored at all times, this includes showing the order of attention of the people in each unit.

**Step 2: Information gathering**

In this step, we need search information about all concepts related with the problem context, we do this with the purpose of have an idea to solve the problem, therefore, its important have definitions of concepts that later can be useful during way.

**Definitions**

**Stack**

a stack is an abstract data type that serves as a collection of elements, with two main operations: Push, which adds an element to the collection, and. Pop, which removes the most recently added element that was not yet removed.

**Hash Table**

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 index value. Data access becomes very fast if we know the index of the desired data.

**Queue**

A queue is an object that represents a data structure designed to have the element inserted at the end of the queue, and the element removed from the beginning of the queue. Java. Util. Queue contains multiple elements before the process. The order of elements of the queue in Java is FIFO (first-in-first-out).

**LinkedList**

A linked list is a sequence of data structures, which are connected via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array.

**Priority Queue**

a priority queue is an abstract data-type similar to a regular queue or stack data structure in which each element additionally has a priority associated with it. In a priority queue, an element with high priority is served before an element with low priority

**Generics**

Generics means parameterized types. The idea is to allow type (Integer, String, … etc., and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types.

**JOptionPane**

JOptionPane is a class library that makes it easy to pop up a simple dialog box that either provides an information message or asks for a simple input from the user. While the class has a lot of methods, most uses of this class are through a few static methods

**Interface**

Java, an interface specifies the behavior of a class by providing an abstract type. As one of Java's core concepts, abstraction, polymorphism, and multiple inheritance are supported through this technology. Interfaces are used in Java to achieve abstraction

**FileReader**

FileReader is a class in the java.io package which can be used to read a stream of characters from the files. This class uses either specified charset or the platform's default charset for decoding from bytes to characters.

**BufferedReader**

BufferedReader is a public Java class that reads text, using buffering to enable large reads at a time for efficiency, storing what is not needed immediately in memory for later use. Buffered readers are preferable for more demanding tasks, such as file and streamed readers.

**Step 3: search for creative solutions**

In this step, we can think in owners’ solutions, now we know that the problem consists in simulate a queue of patient´s, therefore, we need search in different web pages information about the best data structures appropriate to solve the problem. next, we will present different alternatives of data structure.

Structure of data to save patient’s data

**Alternative 1: ArrayList**

Array list is a data structure that can be useful to save data of patient’s also because array list have a size undefine

**Alternative 2: Hash Table**

Hash table is an good option, being that, a hash table have as advantage that the access to the data is so fast if the next condition fulfill: few entries because it more entries of 75% produced collision and the table become inefficient, in this case we are going to save data in a hash table for the id of a patients

**Attention System**

**Alternative 3 : Queue**

A queue will be a good option to implement the attention system of the patients, first this data structure FIFO, allows a great simulation for attend a queue of patients.

**Alternativa 4: Priority Queue**

a variant of a queue will be the priority queue, being that, his principal function consists in add elements in order by a value assign, for example: if it is a number, they are entered from lowest to highest, this can become useful in the case of older patients to give them priority in care.

**Alternativa 5: Stack**

Allows save objects and later get them back in the reverse order that was joined in the stack,

Is characterized for have access to the data type LIFO, this structure can be used to the method undo, that consists in return an action pass, and this help to save again o delete a patient in a queue per a error generated by user.

**Step 4: Transition from ideas to preliminary designs**

In this step its important discard some ideas previously mentioned that are no workable to develop a solution to the problem, therefore, we need to set a correct structure of design problem.

Well now, the first idea that we are going to eliminate is Alternative 1 array list to save data of patients, we do this because in the problem we need efficiency and keep a fine maintainability of the software, therefore, we will focus in the next ideas for his advantages:

**Alternative 2 hash table:**

Well, this alternative satisfies our requirements 1, 2,3, 4 points a favor:

* Is quickly
* good key distribution
* If two keys generate a hash pointing to the same index, the corresponding records can be stored in a linked list

**Alternative 3 Queue:**

Clearly this alternative easily satisfies the requirements 5, 6, 9 point a favor:

* Allows simulate a queue
* Have methods necessary to register an entry or out of a respective laboratory
* Have the structure FIFO

**Alternative 4 Priority Queue:**

Effectively this alternative its necessary because queue complements the priority queue, this alternative satisfies the requirement 7, 9 point a favor:

* Allows increase priority of a patient in the queue
* Allows compare patients with different priorities
* Allows organize my queue by the highest priority.

**Alternative 5 Stack:**

In effect this alternative satisfies the requirement 8 because we can take advantage of the structure stack that is LIFO, but only we can use for undo action like entry of patients or out of a patient. Point a favor:

* Have structure LIFO
* Allows save the last action realized
* Have methods useful

**Step 5:**

Definitely we have the solution for each requirement identified in the problem, well now, finally for have a organized design we are going to specify for each requirement the solution, and how to do it in the context of the problem.

RF1: Load the data of the users of the IPS: BufferedReader – FileReader

RF2: Save the data of the users of the IPS: BufferedReader – FileReader

RF3: Register new patient: Hash table

RF4: Search patient: Hash table methods

RF5: Register entry of a patient: Queue

RF6: Register out of a patient: Queue

RF7: Establish the type of patient care: Priority Queue

RF8: Undo an action of entry or out: Stack

RF9: Show panel that allows monitoring patients in the laboratory: Queue, Priority Queue