**AIRLINE ADMINISTRATOR ENGINEERING METHOD SPECIFICATION**

1. **Context definition**

An airline needs an application to manage the entrance to the planes. For this problem, it is requested to have a database which contains the information of the passengers of the flight. As soon as the plane opens its doors, the application has to show the information of the flight such as the commissioned crew member and the order of entry of the passengers. Moreover, when each passenger arrives at the boarding room it has to register its information and time of arrival on the system. Nevertheless, the airline wants to prioritize the passengers with special conditions like first class passengers, elderly people or other conditions. Lastly, at the time of landing the system has to show to the commissioned crew member the order in which the passengers must exit.

**Symptoms and needs**:

* The client wants a database simulated with a simple text file.
* The client needs an efficient way of searching a passenger information and register its arrival time.
* The client needs a method of knowing the order of entry of the passengers and the crew member in charge.
* The client wants to give priority to its first class clients and clients with special needs.
* The client needs a way of knowing order in which the passengers are going to come out from the plane.

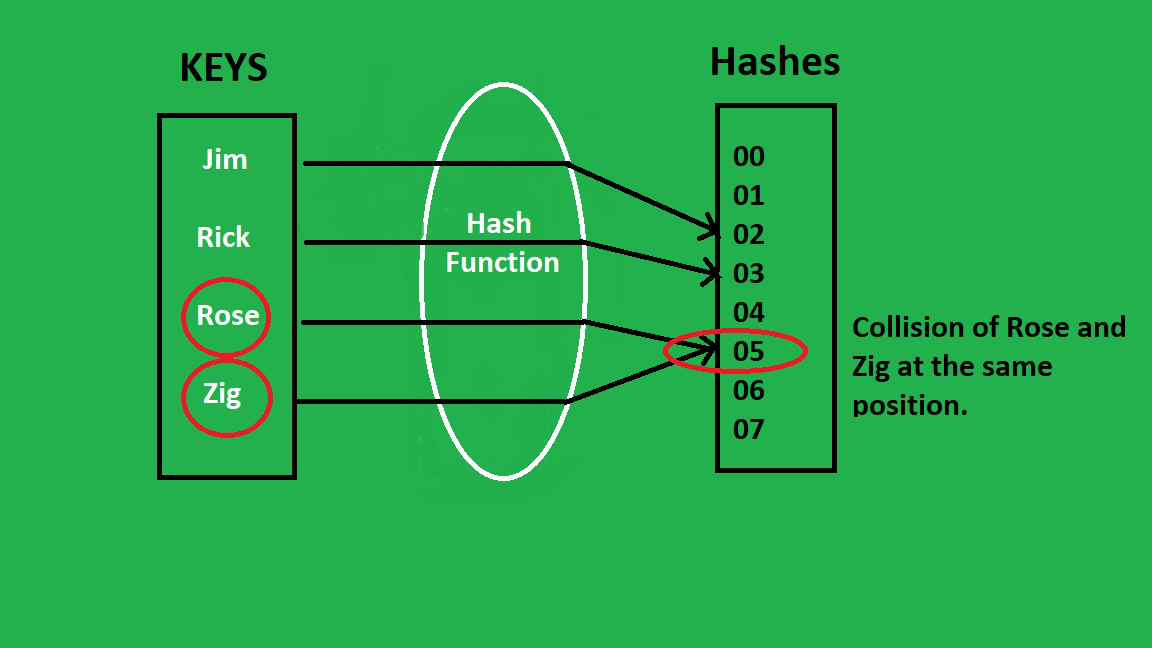
**Problem identification:**

An airline requires a system that allows them to manage the information of each flight, as well as manage the entry and exit of the passengers of a flight.

**Requirement specification:**

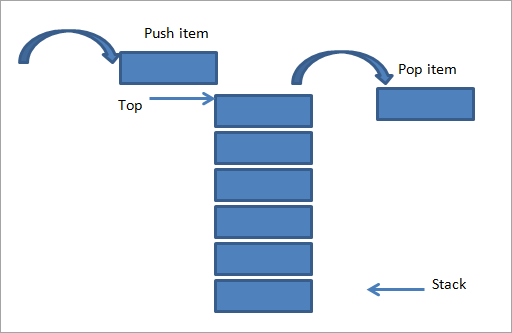
**Note:** The requirement specification document is on the carpet named “docs” and its title is requirement analysis

1. **Required knowledge**
   1. **Data structures:**
      1. **Hash table:** A hash table is a data structure that stores information in an array by a given “key”. Each key is created by a hash function and is assigned to a value. The key gives the position in the array where is going to be stored the value assigned to the key. Many values can have the same key, so in order to solve that it is used a linked list in that position.



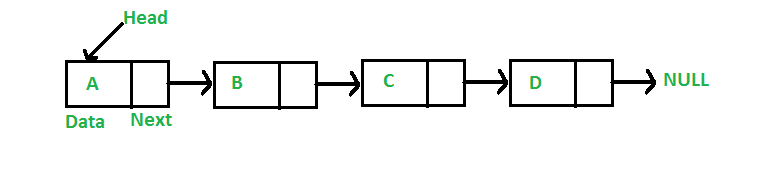
**Source:** [**https://www.geeksforgeeks.org/java-program-to-implement-hash-tables-chaining-with-doubly-linked-lists/**](https://www.geeksforgeeks.org/java-program-to-implement-hash-tables-chaining-with-doubly-linked-lists/)

* + 1. **Stack:** A special type of linked list in which the last object that is inserted is the first object that exits the linked list.

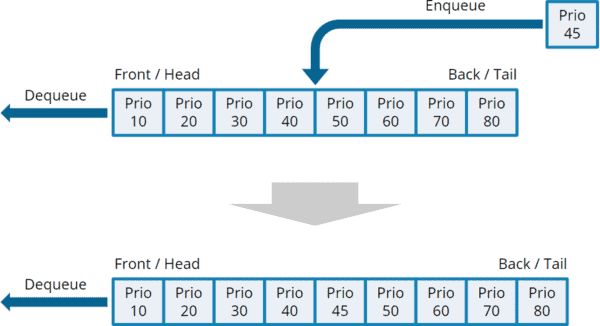


**Soruce:** [**https://www.softwaretestinghelp.com/stack-in-cpp/**](https://www.softwaretestinghelp.com/stack-in-cpp/)

* + 1. **Linked List:** A linear data structure that stores the information sequentially. Each element, at least, is pointing to the next one.

****

**Source:** [**https://www.geeksforgeeks.org/data-structures/linked-list/**](https://www.geeksforgeeks.org/data-structures/linked-list/)

* + 1. **Priority queue:** Special type of linked list in which the first object inserted is the first object removed. Each element has retrieved a priority as an attribute, the higher the priority is, the quicker is the element taken.

**Source:** [**https://www.happycoders.eu/algorithms/priorityqueue-java/**](https://www.happycoders.eu/algorithms/priorityqueue-java/)

* 1. **Time taking:**
     1. **Current:** Can place the current computer time as value for each passenger as they come in, always can compare between passengers and between flight times. If days are needed to be compared, then the date object can be used with the current time for longer comparisons.
  2. **Data Load and Save:**
     1. **Json:** For the loading and saving of information we can use gson library created by google. This library creates a file or rewrite an existing file with the information in Json format, allowing us to serialize Java Objects in Json notation.

**Source:** [**https://www.json.org/json-en.html**](https://www.json.org/json-en.html)

1. **Proposed solutions**

Solutions are divided by different types being Load and Save, Database, Sort, and Date. Which will have multiple solutions depending on specific situations during the airline procedure:

* 1. **Load & Save**
* Json In/Out in txt file
* Simple txt file
* Excel file
  1. **Database**
* Array
* Arraylist
* BST
* HashTable
* Queue
* Stack
  1. **Sort**
* Heap sort
* Bubble sort
* Insertion sort
* Selection sort
* Collections sort
  1. **Date**
* Current time function
* Calendar function
* Date format function

**4. Discarded Solutions**

* 1. **Load and save:**
* Simple txt file.
  + We discarded this option because it didn’t fit the required format.
* Excel file
  + We discarded this option because it didn’t fit the required format.
  1. **Database:**
* Array
  + We discarded it is not efficient for a great mass of people that the airline is expected to receive.
* Array list
  + We discard this idea because don’t need a dynamic data structure because the quantity of chairs is already set in the flight.
* BST
  + We discard this idea because don’t make sense with the requirements because the father-sons structure doesn’t make sense for arrange people in a section of chairs.
  1. **Sort:**.
* Collections sort.
  + We discard this solution because we do not have enough experience with this kind of sort system.
* Selection sort, Bubble sort & Insertion sort.
  + We discarded this tree solution because the algorithm is not efficient enough to sort great mass of people information. Even more, we don’t have the required experience to optimize them.
  + We discard this solution for efficiency reasons.
  1. **Date:**
* Date format function
  + We discard this solution because we need the complete information about the time to compare the time of arrived.
* Current time function
  + We prefer the use of calendar for more easy development of the solution.

**5. Evaluation criteria:**

With the purpose to find the best solution to be implement, we will create an evaluation system with a numeric value to make easier the evaluation:

* Efficiency
* Usability
* Maintainable
* Scalability