**ENGINEERING DESIGN METHOD**

1. **Requirements**

|  |  |
| --- | --- |
| Client | airline |
| User | tribulation |
| Functional requirements | R1: Simulate database by initial passenger loading into the system  R2: Search for complete passenger information efficiently as soon as you reach the boarding area.  R3: Record passenger arrival time to the boarding lounge and create a reward for their timeliness.  R4: Allow entry to the aircraft on a first-come, first-served basis and bearing in mind that the call is still in sections, starting with the sections furthest from the entrance door to them.  R5: Allow the first class to have exclusive benefits at the time of entry, that is, prioritize other data such as accumulated miles, in addition to special attention required or other relevant data.  R6: Allow the crew in charge to see in which order passengers should leave taking into consideration the seating configuration. where those in the first rows are the first to leave. and, in the rows the order is followed with reference to the proximity to the corridor. |
| Context of the problem | After excellent performance in your academic work, you and your colleagues have been selected by a renowned airline to make a first version of a system whose main objective is to improve the order in the process of entry and exit of the plane. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R1: Simulate database by initial passenger loading into the system | | |
| Summary | the system enters a txt file where all the data of the passengers, their priority and their class, which this file is loaded for the crew to know | | |
| Inputs | **Input name** | **Data Kind** | **Condition of selection** |
| Passenger | txt | input is not null |
| Previous needed activities. | that the file already exists, because from there comes the passenger information | | |
| Results | a database saved in a hash table | | |
| Outputs | **Output name** | **Data Kind** | **Condition of selection** |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R2: Search for complete passenger information efficiently as soon as you reach the boarding area. | | |
| Summary | the database is loaded and passengers are searched for these to be organized by priority. | | |
| Inputs | **Input name** | **Data Kind** | **Condition of selection** |
| key | String | input is not null and the key exists |
| Previous needed activities. | the passenger data has already been loaded into the hash table correctly | | |
| Results | a database saved in a hash table | | |
| Outputs | **Output name** | **Data Kind** | **Condition of selection** |
| keyPassenger | passenger | Passenger isnot null |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R3: Record passenger arrival time to the boarding lounge and create a reward for their timeliness. | | |
| Summary | in the txt loaded at the start, an attribute is included which tells us if a person arrived on time or not, so that this is the order of arrival of passengers. | | |
| Inputs | **Input name** | **Data Kind** | **Condition of selection** |
| dataPassengers | txt | Txt is not null |
| Previous needed activities. | that the file already exists, because from there comes the passenger information | | |
| Results | passenger information and arrival order and improve the preference of passengers who arrived first | | |
| Outputs | **Output name** | **Data Kind** | **Condition of selection** |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R4: Allow entry to the aircraft on a first-come, first-served basis and bearing in mind that the call is still in sections, starting with the sections furthest from the entrance door to them. | | |
| Summary | order the passengers for entry where they are ordered forward and when they are going to enter the chairs, either in order of arrival | | |
| Inputs | **Input name** | **Data Kind** | **Condition of selection** |
| dataPassengers | Passengers | Passengers isnot null |
| Previous needed activities. | the passenger data has already been loaded into the hash table correctly | | |
| Results | an arrangement where this the order of entry of the passengers of the plane | | |
| Outputs | **Output name** | **Data Kind** | **Condition of selection** |
| enterPassengers | arrayList | The array is in order. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R5: Allow the first class to have exclusive benefits at the time of entry, that is, prioritize other data such as accumulated miles, in addition to special attention required or other relevant data. | | |
| Summary | to the first class, enters first than the plane, and its entry is by order of arrival, if the order of arrival is the same look data as miles or indiscretions. | | |
| Inputs | **Input name** | **Data Kind** | **Condition of selection** |
| dataPassengers | Passengers | Passengers isnot null |
| Previous needed activities. | the passenger data has already been loaded into the hash table correctly | | |
| Results | an arrangement where this the order of entry of the passengers of the plane | | |
| Outputs | **Output name** | **Data Kind** | **Condition of selection** |
| enterFistClassPassengers | arrayList | The array is in order. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R6 Allow the crew in charge to see in which order passengers should leave taking into consideration the seating configuration. where those in the first rows are the first to leave. and, in the rows the order is followed with reference to the proximity to the corridor. | | |
| Summary | show the order in which the tribulation should go out, where the first class comes out first, and then the economy class, considering that it goes out near the door and first the one closest to the corridor. | | |
| Inputs | **Input name** | **Data Kind** | **Condition of selection** |
| dataPassengers | Passengers | Passengers isnot null |
| Previous needed activities. | the passengers are already seated in the plane and wait for their moment of departure | | |
| Results | an arrangement where this the order of out of the passengers of the plane | | |
| Outputs | **Output name** | **Data Kind** | **Condition of selection** |
| outPassengers | arrayList | The array is in order. |

**Complexity analysis**.

Temporal:

Line : code: amount :

|  |  |  |
| --- | --- | --- |
| 1 | File file = new File(archivo); | 1 |
| 2 | hasTable = new HasTable<>(num); | 1 |
| 3 | try{ | 1 |
| 4 | FileInputsStream fis = new FileInputsStream(file); | 1 |
| 5 | BufferedReader reader = new BufferedReader(new inpustreamReader(fis)); | 1 |
| 6 | String line; | 1 |
| 7 | While((line = reader.readLine()) ¡= null) | n+1 |
| 8 | String [] atributs = line.split(“;)”; | n |
| 9 | Passenger Passenger = new Passenger (atributs[0.]) | n |
| 10 | hasTable.insert(atributs[0], passenger); | n |
| 11 | Fis.close(); | 1 |
| 12 | Catch (FileNotFoundException e) | 1 |
| 13 | e.printStackTrace() | 1 |
| 14 | Catch(IOException e ) | 1 |
| 15 | e.printStackTrace(); | 1 |

Total: 4**n+12**

Notation O:

where g(n) = 4n +12 => O(n^2)

|  |  |  |
| --- | --- | --- |
| 1 | Int index = hash(key) | 1 |
| 2 | NodeHash<K,V> node = list[index] | 1 |
| 3 | While(node ¡= null) | n+1 |
| 4 | If(node.getkey().compareTo(key) == 0) | n |
| 5 | Return node.getValue(); | 1 |
| 6 | Node = node.gerNext(); | n |
| 7 | Return null | 1 |

Total:  **3n+5**

Notation O:

Where g(n) = 3n + 5 => O(n^2)

**Space:**

**Line : code: size: amount:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | File file = new File(archivo); | 8 bytes | 1 |
| 2 | hasTable = new HasTable<>(num); | 32 bytes | 1 |
| 3 | try{ |  | 1 |
| 4 | FileInputsStream fis = new FileInputsStream(file); | 8KB: 8192 bytes | 1 |
| 5 | BufferedReader reader = new BufferedReader(new inpustreamReader(fis)); | 8 KB :8192 bytes | 1 |
| 6 | String line; | 16 bytes | 1 |
| 7 | While((line = reader.readLine()) ¡= null) |  | n+1 |
| 8 | String [] atributs = line.split(“;)”; | 16 \* n bytes | n |
| 9 | Passenger Passenger = new Passenger (atributs[0.]) | 32 bytes | n |
| 10 | hasTable.insert(atributs[0], passenger); | 32 bytes | n |
| 11 | Fis.close(); |  | 1 |
| 12 | Catch (FileNotFoundException e) |  | 1 |
| 13 | e.printStackTrace() |  | 1 |
| 14 | Catch(IOException e ) |  | 1 |
| 15 | e.printStackTrace(); |  | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Int index = hash(key) | 4 bytes | 1 |
| 2 | NodeHash<K,V> node = list[index] | 32 bytes | 1 |
| 3 | While(node ¡= null) |  | n+1 |
| 4 | If(node.getkey().compareTo(key) == 0) |  | n |
| 5 | Return node.getValue(); | 8192 bytes | 1 |
| 6 | Node = node.gerNext(); | 81292 | n |
| 7 | Return null | 0 bytes | 1 |

1. **Problem definition:**

**Needs and symptoms:**

**General:**

* Deficiency in the airline's passenger check-in and check-out system.  Generating discomfort and dissatisfaction on the part of passengers.

**Specific:**

* No place to load passenger information.

* Need for an improvement in the search of passengers for check-in at the boarding lounge.

* Improve the call of passengers, by sections and classes, at the entrance to the aircraft and show them to the staff in charge of their call.

* Improve the relationship with your best customers and prioritize those with preferential needs.

Suggestions: Prioritize first class passengers.

* Establish a better way for passengers to exit at the time of disembarkation. Taking as a reference their location in the aircraft and their proximity to the aisle.

**General Problem:**

The airline requires a system to optimize the orderly check-in and check-out process of the contracting airline's aircraft.

1. **Gathering of information:**

Learning from others about related issues:

a. Amadeus' Altéa Suite is a complete passenger services system that offers booking, inventory management and departure control functionality in a single integrated solution. Airline passenger service systems are essential and highly complex platforms.

1. "Airline Operations: A Practical Guide" by Peter J. Bruce. This book provides a practical guide to airline operations management, including passenger management systems.

1. AeroCRS was founded in 2006, focusing on solutions for the travel industry, providing an excellent, innovative and very easy to use passenger service system, with over 90 transportation providers worldwide as our customers. The system includes inventory management, reservation and routing management, fare management, B2C, B2B, DCS, FPP, full reporting center, BI dashboard, API for third party integrations including GDS, OTA and much more.

Search for definitions related to the problem:

* **Check-in system:** is the set of processes and procedures used to allow passengers access to the services and flights offered by the airline. This check-in system consists of several steps and is carried out both at the airport and online, through the Internet.
* **Departure system:** is the set of processes and procedures used to allow passengers to depart in an orderly and safe manner from the aircraft and the airport once they have arrived at their destination.
* **First class:** the most exclusive and luxurious category offered to passengers. This category offers additional amenities and services over and above those offered in lower classes, such as business class or economy class.
* **Passengers with preferential needs** are those who require additional assistance during their trip due to disability, advanced age, pregnancy, illness or other special circumstances. These passengers have additional rights and services to ensure that they can travel in safety, comfort and dignity.
* **Passenger check-in system:** An airline passenger check-in system is a set of procedures and technologies used to collect and store information on passengers boarding a flight.

1. **Search for creative solutions:**

1. Use stack in the implementation of the passenger entry/exit system; stacks are a data structure that allows storing and accessing elements in an orderly and efficient way. The stack is a set of elements organized in the form of a linear list, where access to the elements is done in a restrictive way, i.e., only the last element that has been added to the stack can be accessed.
2. Using a queue in the implementation of the passenger check-in/check-out system solution, a queue is a data structure that allows storing and accessing elements in an orderly and efficient manner. The queue is a set of items organized in the form of a linear list, where access to items is done on a first-in, first-out (FIFO) basis.
3. Using priority queues in the implementation, a priority queue is a data structure that allows storing and accessing elements in an orderly and efficient manner, but in this case, based on a priority associated with each element. That is, each item has a priority assigned to it and the priority queue ensures that items are processed in order of priority, from highest to lowest.
4. Use priority queues and queue in the implementation of the passenger check-in/check-out system; queue is a data structure that allows storing and accessing elements in an orderly and efficient manner; furthermore a priority queue is a data structure that allows storing and accessing elements in an orderly and efficient manner, but in this case, based on a priority associated to each element. In addition to storage of information we used a hash table, which will receive the information of the passengers of the txt.

1. **Transition from Ideas to Preliminary Designs:**

We are going to discard the first 3 options; because implementing only one ordering method would make the implementation of the system much more difficult, besides the ease of using both queue and priority queues will help us to solve the problem in a much more efficient and practical way.

One way to know that this is the best option is the following: by using the queue we will ensure that all people will be served according to their order of arrival at the boarding site of the aircraft, and by using the priority queues we achieve the airline's goal of having a better relationship with its best customers and also give priority to all those people who need them for some reason or another, both people with disabilities, such as pregnant women or the elderly.

1. **Evaluation and Selection of the Best Solution:**

**System evaluation criteria:**

**Criterion 1:** Simulate database by initial passenger loading into the system.

[3] Display the order of passengers by arrival and priority.

[2] Display the order of passengers by arrival.

[1] Any order is displayed.

**Criterion 2:** Search for complete passenger information as soon as the passenger arrives at the boarding lounge.

[3] Efficient solution.

[2] Gives inefficient solution.

[1] No solution

**Criterion 3:** Record the passenger's arrival time at the boarding lounge and create a reward for punctuality.

[3] Record arrival time efficiently and create incentives.

[2] either does not record arrival time or does not create incentives.

[1] does not record arrival time and does not create incentives.

**Criterion 4:** Allow entry to the aircraft by section order.

[2] allows entry in order of arrival and aircraft distribution.

[1] allows entry on a first-come, first-served basis, and does not consider the order in which the aircraft is distributed for entry.

**Criterion 5:** Allow first class to have exclusive benefits at the time of entry.

[5] Efficiently prioritize all special passengers.

[4] Only prioritize other data such as mileage accrual.

[3] only give special attention required.

[2] only gives priority to the best customers.

[1] does not give priority to any passenger.

**Criterion 6:** Once the aircraft doors open, the system must allow the crew in charge to show the order in which passengers enter the aircraft.

[2] shows the ideal order for passenger entry.

[1] displays the list inefficiently.

**Criterion 7:** Allow the crew in charge to see in which order passengers should depart taking into consideration the seating configuration. where those in the first rows are the first to depart. and also in the rows the order is followed with reference to the proximity to the aisle.

[3] Those in the first rows and those closest to the aisle leave first.

[2] either those closest to the aisle do not leave first, or those in the first rows do not leave first.

[1] neither the first rows nor those in the first aisles leave first.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Alternatives/Criteria | C1 | C2 | C3 | C4 | C5 | C6 | C7 | Total |
| use queue | 2 | 3 | 3 | 1 | 1 | 1 | 1 | 12 |
| use queue, priority queues and hash table | 3 | 3 | 3 | 2 | 5 | 2 | 2 | 20 |

Based on the above evaluation, Alternative 2 should be selected, since it obtained the highest score according to the defined criteria. It should be considered that the criterion in which the alternative was worse evaluated than the other alternative should be handled appropriately.

1. **Preparation of Reports and Specifications:**

**Problem specification:**

**Problem:** passenger check-in and check-out at an airline.

**Inputs:**

* Passenger information (first name, last name, passport number, contact information, ticket type, seat preferences, miles accrued, among others).
* Arrival time of passengers at the boarding lounge.
* Seating configuration on the aircraft.

**Departures:**

* List of passengers with the order of entry to the aircraft.
* List of passengers with the order of departure to the aircraft.

the class diagram is in another file in this same folder.

1. **Implementation:**

the code that solves the problem. In the other folder.