Jhon Stiven Arboleda, Juan Sebastián Barrera, Alejandro García

Tarea Integradora 1

1. **Identification of the problem**

A large bank wants to develop software that models the operation of one of its offices with the highest flow of people. Its main need is to be able to fulfill all the client's needs.

The bank needs:

* Manage shifts by entering the lines either the customer line or a line with different priorities.
* Manage data tables with all customer information.
* Allow the customer to perform different operations at the time of serving, that is, when their turn comes.

Problem: The bank has no way to handle these types of actions effectively in a software for a high flow of problems.

1. **Research**

**Functional requirements:**

|  |  |
| --- | --- |
| **Name** | **FR1:** Register a customer and non-customer |
| **Summary** | The system must register a client at the time of obtaining their turn with their respective data |
| **Input** | **name** of the client and **identification** of the client |
| **Output** | The client has been successfully registered |

|  |  |
| --- | --- |
| **Name** | **FR2:** Assign to a row |
| **Summary** | The system must register a client at the time of obtaining their turn with their respective data. In case that the person that arrive is a non-customer it would be register as a new client in the bank |
| **Input** | **name** of the client and **identification** of the client |
| **Output** | The client has been successfully registered |

|  |  |
| --- | --- |
| **Name** | **FR3:** Search the database |
| **Summary** | The person in charge of the attention will be able to look for the client in the database with his identity card before the client arrives at his office also it would display all the user information. |
| **Input** | **Identification card** of the client |
| **Output** | Table with customer data. Such as name, identity card, bank account, debit / credit cards, date of payment of the credit card and date it was incorporated into the bank |

|  |  |
| --- | --- |
| **Name** | **FR4:** Withdrawals |
| **Summary** | The client may modify the amount of his savings account when requesting a withdrawal |
| **Input** | **Bank account** of the client |
| **Output** | The customer has made a withdrawal from the account |

|  |  |
| --- | --- |
| **Name** | **FR5:** Consignment |
| **Summary** | The client may modify the amount of his savings account when requesting a consignment. |
| **Input** | **Bank account** of the client |
| **Output** | The customer has made a transfer to the account. |

|  |  |
| --- | --- |
| **Name** | **FR6:** Cancel account |
| **Summary** | Deletes your information from the customer database and adds them to a database exclusively for those who cancel their accounts at the bank. In any case, both the date and the reason for cancellation will be saved in a new database |
| **Input** | **Bank account** of the client |
| **Output** | The account has been deleted |

|  |  |
| --- | --- |
| **Name** | **FR7:** Card payment |
| **Summary** | The user can pay the amount used with the credit card so far.  You can make the payment in cash or through your savings account.  Person can only pay the full card or the fee |
| **Input** | **Bank account** of the client |
| **Output** | The amount of money has been paid. |

|  |  |
| --- | --- |
| **Name** | **FR8:** Perform *undo* |
| **Summary** | It will serve to undo mistakes, even after they have been saved |
| **Input** | **----------** |
| **Output** | The action has been undone. |

|  |  |
| --- | --- |
| **Name** | **FR9:** Perform *redo* |
| **Summary** | It will serve to redo mistakes, even after they have been saved |
| **Input** | **----------** |
| **Output** | The action has been redone. |

|  |  |
| --- | --- |
| **Name** | **FR10:** Create savings account |
| **Summary** | The user is created a savings account only if he is not a customer of the bank |
| **Input** | **Name** of the client, **id** of the client, **gender** of the client, **age** of the client, **pregnated, owe** of the card, **cardSpace, quotas, fees, paymentDay** |
| **Output** | The account was created |

|  |  |
| --- | --- |
| **Name** | **FR11:** Create card |
| **Summary** | The user is created a card only if he is not a customer of the bank |
| **Input** | **Name** of the client, **id** of the client, **gender** of the client, **age** of the client, **pregnated,** |
| **Output** | The action has been redone. |

|  |  |
| --- | --- |
| **Name** | **FR12:** Attend users from a row |
| **Summary** | In case that attend the priority row it would show the client be the highest priority else it would attend in FIFO order |
| **Input** | **-------------** |
| **Output** | It would be able to do actions on the user account. |

|  |  |
| --- | --- |
| **Name** | **FR13:** Show up in the graphic user interface the users |
| **Summary** | It would show all the clients in a table view format and the would be order by name, id, amount and time register |
| **Input** | **-------------** |
| **Output** | The users information |

**Non-functional requirements**

|  |  |
| --- | --- |
| **Name** | **NFR1:** The normal row will be use queue data structure |
| **Summary** | The implementation of the normal row should be will a generic queue |

|  |  |
| --- | --- |
| **Name** | **NFR2:** The priority row will be use heap data structure |
| **Summary** | The implementation of the priority row should be will a generic heap |

|  |  |
| --- | --- |
| **Name** | **NFR3:** Order algorithms |
| **Summary** | The software will have four order algorithms, only one can have O(n2) complexlity the others must be better in complexlity |

|  |  |
| --- | --- |
| **Name** | **NFR4:** The search of the clients |
| **Summary** | It should be an efficient search with a O(1) complexlity |

|  |  |
| --- | --- |
| **Name** | **NFR5:** Do *undo* and *redo* |
| **Summary** | The implementation of the undo and redo will be with a generic stack |

**Stacks:** Stacks are data structures wherein the last element that enters is the first element that leaves (LIFO, **L**ast **I**n **F**irst **O**ut).

**Queues:** Queues are data structures wherein the first elements that enters is the first element that leaves (FIFO, **F**irst **I**n **F**irst **O**ut)

**Hash Tables:** Hash tables are data structures that map keys to values. In the Java programming language, keys and values can be any object that is not null. The hash tables use a hash function to map the key to the value. This means that to retrieve a value the user must enter the key associated with that value.

A problem that might arise with hash tables are collisions, which happen when a given key has more than 2 values associated with it. These problems are solved with open addressing and chaining. In open addressing, the second value is stored in another address. In chaining, both values are stored in a linked list.

**Heap:** A heap is a tree-like data structure with information belonging to an ordered set. Maximum mounds have the characteristic that each parent node has a value greater than that of any of its child nodes, while in minimum mounds, the value of the parent node is always less than that of its child nodes. A tree satisfies the heap condition if it satisfies the previous condition and is also a nearly complete binary tree. A binary tree is complete when all levels are full, with the exception of the last one, which is filled from left to right.

**Bank:** Attention in a bank generally works as follows: There are two lines, general users and priority users (usually customers with a preferential account or people with disabilities). To assign the clients to a line and give them their turn to be attended, they must register with their name and ID, the correspondent in the cubicle will obtain the information from the client and will be able to carry out the operations that the client needs (withdraw, consign, cancel the account and / or pay an amount)

In the event that the client does not have an account within the bank, he or she may create it once it is his turn to be served.

1. **Creative solutions**

**Brainstorming:**

* 1. **Keep a manual record of users:** Each time a user enters, they will have to manually register in a book and thus an advisor will assign them a shift.
  2. **Hire third-party software that fulfill its functions:** The bank will buy the software from a company outside the same bank. The software will have features similar to those you need.
  3. **Create a software:** Hire a software development company to make software to the bank with the essential needs they need.
  4. **Have an online database:** Customers can register in the database online and managers in each cubicle can review the information in the same way.

1. **Selection of the best solution**
   1. **Keep a manual record of users:** It´s inefficient and slow for a high flow of people. **It´s not a good solution.**
   2. **Hire third-party software that fulfill its functions:** It is expensive to acquire software from another company and it is not guaranteed that all needs will be met. **It´s not a good solution**
   3. **Create a software:** It is a good option, because you can have a suitable software for the bank, meeting the exact needs that they require. **It´s a good solution.**
   4. **Have an online database:** When the bank does not have internet, the database will not be accessible and attention will be lost. **It´s not a good solution**

Final decision: CREATE A SOFTWARE.

**ADT (Abstract Data Type)**

|  |  |
| --- | --- |
| 1. NAME | QUEUE |
| REPRESENTATION | Java Queue  Queue = <<e1, e2,…,en>,front,back> |
| INVARIANT | Size(Queue) = n front = e1 back = en 0 >= n |
| OPERATIONS | Queue - 🡪 Queue  Offer Queue x Element -> Queue  Poll Queue 🡪 Element  Peek Queue 🡪 Element  Size 🡪 Integer  IsEmpty🡪boolean  Clear 🡪 void |

Constructor operation

|  |
| --- |
| Queue - 🡪 Queue  “Builds an empty queue”  {Pre: -}  {Pos: Queue q =} |

Modifier operation

|  |
| --- |
| Offer Queue x Element -> Queue  “Insert a new Element e in the Queue”  {Pre: Queue q = <e1, e2,…,en> and element e or q = ø and element e}  {Pos: Queue q = <e1, e2,…,en, e> or q = <e> } |

|  |
| --- |
| Poll Queue 🡪 Element  “Extracts the element in Queue q’s front”  {Pre: ø i.e. q = <e1, e2,…,en> }  {Pos: Queue q = <e2,e3,…,en-1> and element e1 } |

|  |
| --- |
| Peek Queue 🡪 Element  “Recovers the value of the element on the front of the queue.”  {Pre: Queue q != i.e q = <e1, e2,…,en> }  {Pos: Element e1} |

|  |
| --- |
| Size 🡪 integer“Get the QUEUE size” {Pre: Queue q = or q = <e1, e2,…,en> }  {Pos: 0 or n} |

|  |
| --- |
| IsEmpty 🡪 boolean“Determines if the Queue q is empty or not” {Pre: Queue q}  {Pos: true if queue is empty, false if is not} |

Destructor operation

|  |
| --- |
| Clear“Destroys queue q freeing memory” {Pre: Queue q}  {Pos: -} |

|  |  |
| --- | --- |
| NAME | HASH TABLE |
| REPRESENTATION | HashTable in Java Example | Java HashTable Tutorial |
| INVARIANT | Inv {All keys cannot be null,  Every value will have a key} |
| OPERATIONS | HashTable 🡪 HashTableHashFunction: Key Search: Key 🡪 Value  Insert: Key, Value  Delete: Key |

Constructor operation

|  |
| --- |
| HashTable 🡪 HashTable  “Creates a new hashTable”  {Pre: true}  {Pos: An empty hashTable} |

Modifiers operation

|  |
| --- |
| HashFunction: Key  “Convert the key to an integer”  {Pre: True}  {Pos: An integer that indicates the object position in the array} |

|  |
| --- |
| Search: Key 🡪 Value  “Get the value searched”  {Pre: ø}  {Pos: Returns the element found} |

|  |
| --- |
| Insert: Key, Value  “Add a new element in the hash table”  {Pre: Key && Value ≠ null}  {Pos: Insert the element in the hash table} |

|  |
| --- |
| Delete: Key  “Delete an element with a key”  {Pre: Key ≠ null}  {Pos: Erase the found element} |

|  |  |
| --- | --- |
| NAME | STACK |
| REPRESENTATION | Stack Class in Java Explained with Examples | CodeAhoy  Stack = <<e1, e2,…,en>,top > |
| INVARIANT | n >= 0 Size = n top = en |
| OPERATIONS | Stack - 🡪 Stack  Pop Stack 🡪 Element  Top Stack 🡪 Element  Push Stack x Element -> Stack  isEmpty Stack 🡪 Boolean  Size 🡪 integer  Clear |

Constructor operation

|  |  |
| --- | --- |
| Stack - 🡪 Stack  “Builds an empty stack”  {Pre: -}  {Pos: Stack s = } |  |

Modifiers operations

|  |
| --- |
| Push Stack x Element -> Stack  “Adds the new element e to stack s”  {Pre: Stack s = and element e or s = <e1, e2,…,en> and element e}  {Pos: Stack s = s = <e1, e2,…,en, e> or s = <e> } |

|  |
| --- |
| isEmpty Stack 🡪 Boolean  “Determines if the stack s is empty or not”  {Pre: Stack s}  {Pos: true if s = False if s != } |

|  |
| --- |
| Top Stack 🡪 Element  “Recovers the value of the element on the top of the stack”  {Pre: : Stack s != i.e s = <e1, e2,…,en> }  {Pos: Element en} |

|  |
| --- |
| Pop Stack 🡪 Element  “Extracts from the stack s, the most recently inserted element, and recover that element”  {Pre: Stack s != i.e s = <e1, e2,…,en> }  {Pos: Stack s = <e1, e2,…,en-1> , Element en} |

|  |
| --- |
| Size 🡪 integer  “Get the size of the Stack”  {Pre: Stack s = <e1, e2,…,en> or s = }  {Pos: n or 0} |

Destructor operation

|  |
| --- |
| Clear “Destroys stack s freeing memory” {Pre: Stack s}  {Pos: -} |

|  |  |
| --- | --- |
| NAME | HEAP |
| REPRESENTATION | Heap Data Structure - GeeksforGeeks |
| INVARIANT | Size >= 0 |
| OPERATIONS | Heap🡪 Heap  buildMaxHeap  maxHeapify  exist 🡪 Boolean |

Constructor operation

|  |
| --- |
| Heap - 🡪 Heap  “Builds an empty heap”  {Pre: -}  {Pos: Heap h = } |

Modifier operations

|  |
| --- |
| Maxheapify  “moves the element to the correct position”  {pre: Heap h}  {pos: Heap h} |
| BuildMaxheap  “Moves all the elements to the correct position to start the heapsort”  {pre: Heap h}  {pos: Heap h in descendent order} |

|  |
| --- |
| HeapSort  “Order all the elements”  {pre: Heap h with buildMaxheap}  {pos: heap h in ascendant order} |

|  |
| --- |
| Swap  “swap two elements in the heap h”  {pre: heap h}  {pos: heap h} |

|  |
| --- |
| Exist int i  “determinates if that position exist”  {pre: heap h}  {pos: true if ei exist, false if not} |

1. **Test Case**

**HashTable**

**testDelete2()**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | | **7** | | **8** | | **9** | | **10** | |
| **0** | **724520** | **null** | **null** | **null** | **null** | **-91454** | **137** | **null** | **null** | **82** | **-4134** | **null** | **null** | **null** | **null** | **1820** | **7** | **null** | **null** | **null** | **null** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | | **7** | | **8** | | **9** | | **10** | |
| **0** | **724520** | **null** | **null** | **null** | **null** | **-91454** | **137** | **null** | **null** | **82** | **-4134** | **null** | **null** | **null** | **null** | **null** | **nul l** | **null** | **null** | **null** | **null** |

**testInsert2()**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | | **7** | | **8** | | **9** | | **10** | |
| **0** | **724520** | **null** | **null** | **null** | **null** | **-91454** | **137** | **null** | **null** | **82** | **-4134** | **null** | **null** | **null** | **null** | **1820** | **7** | **null** | **null** | **null** | **null** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | | **7** | | **8** | | **9** | | **10** | |
| **0** | **724520** | **null** | **null** | **null** | **null** | **-91454** | **137** | **null** | **null** | **82** | **-4134** | **null** | **null** | **null** | **null** | **1820** | **7** | **null** | **null** | **450** | **4235** |

**Heap**

**testBuildMaxHeap2()**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **3** | **2** | **16** | **9** | **10** | **14** | **8** | **7** |
| **4** | **3** | **0** | **2** | **0** | **0** | **1** | **0** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **3** | **2** | **16** | **9** | **10** | **14** | **8** | **7** |
| **4** | **3** | **0** | **2** | **0** | **0** | **1** | **0** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **3** | **14** | **16** | **9** | **10** | **2** | **8** | **7** |
| **4** | **3** | **0** | **0** | **0** | **0** | **1** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **3** | **14** | **16** | **9** | **10** | **2** | **8** | **7** |
| **4** | **3** | **0** | **0** | **0** | **0** | **1** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **10** | **14** | **16** | **9** | **3** | **2** | **8** | **7** |
| **4** | **3** | **1** | **0** | **0** | **0** | **0** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **10** | **14** | **16** | **9** | **3** | **2** | **8** | **7** |
| **4** | **3** | **1** | **0** | **0** | **0** | **0** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **16** | **10** | **14** | **1** | **9** | **3** | **2** | **8** | **7** |
| **4** | **0** | **1** | **0** | **3** | **0** | **0** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **16** | **10** | **14** | **7** | **9** | **3** | **2** | **8** | **1** |
| **4** | **0** | **1** | **0** | **5** | **0** | **0** | **2** | **1** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **16** | **10** | **14** | **7** | **9** | **3** | **2** | **8** | **1** |
| **4** | **0** | **1** | **0** | **5** | **0** | **0** | **2** | **1** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | **4** | **10** | **14** | **7** | **9** | **3** | **2** | **8** | **1** |
| **0** | **4** | **1** | **0** | **5** | **0** | **0** | **2** | **1** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | **14** | **10** | **4** | **7** | **9** | **3** | **2** | **8** | **1** |
| **0** | **0** | **1** | **4** | **5** | **0** | **0** | **2** | **1** | **3** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | **14** | **10** | **8** | **7** | **9** | **3** | **2** | **4** | **1** |
| **0** | **0** | **1** | **1** | **5** | **0** | **0** | **2** | **4** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | **14** | **10** | **8** | **7** | **9** | **3** | **2** | **4** | **1** |
| **0** | **0** | **1** | **1** | **5** | **0** | **0** | **2** | **4** | **3** |

**testHeapSort2**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **3** | **2** | **16** | **9** | **10** | **14** | **8** | **7** |
| **4** | **3** | **0** | **2** | **0** | **0** | **1** | **0** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **3** | **2** | **16** | **9** | **10** | **14** | **8** | **7** |
| **4** | **3** | **0** | **2** | **0** | **0** | **1** | **0** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **3** | **14** | **16** | **9** | **10** | **2** | **8** | **7** |
| **4** | **3** | **0** | **0** | **0** | **0** | **1** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **3** | **14** | **16** | **9** | **10** | **2** | **8** | **7** |
| **4** | **3** | **0** | **0** | **0** | **0** | **1** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **10** | **14** | **16** | **9** | **3** | **2** | **8** | **7** |
| **4** | **3** | **1** | **0** | **0** | **0** | **0** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **1** | **10** | **14** | **16** | **9** | **3** | **2** | **8** | **7** |
| **4** | **3** | **1** | **0** | **0** | **0** | **0** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **16** | **10** | **14** | **1** | **9** | **3** | **2** | **8** | **7** |
| **4** | **0** | **1** | **0** | **3** | **0** | **0** | **2** | **1** | **5** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **16** | **10** | **14** | **7** | **9** | **3** | **2** | **8** | **1** |
| **4** | **0** | **1** | **0** | **5** | **0** | **0** | **2** | **1** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **16** | **10** | **14** | **7** | **9** | **3** | **2** | **8** | **1** |
| **4** | **0** | **1** | **0** | **5** | **0** | **0** | **2** | **1** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | **4** | **10** | **14** | **7** | **9** | **3** | **2** | **8** | **1** |
| **0** | **4** | **1** | **0** | **5** | **0** | **0** | **2** | **1** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | **14** | **10** | **4** | **7** | **9** | **3** | **2** | **8** | **1** |
| **0** | **0** | **1** | **4** | **5** | **0** | **0** | **2** | **1** | **3** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | **14** | **10** | **8** | **7** | **9** | **3** | **2** | **4** | **1** |
| **0** | **0** | **1** | **1** | **5** | **0** | **0** | **2** | **4** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **16** | **14** | **10** | **8** | **7** | **9** | **3** | **2** | **4** | **1** |
| **0** | **0** | **1** | **1** | **5** | **0** | **0** | **2** | **4** | **3** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **14** | **1** | **10** | **8** | **7** | **9** | **3** | **2** | **4** | **16** |
| **0** | **3** | **1** | **1** | **5** | **0** | **0** | **2** | **4** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **14** | **1** | **10** | **8** | **7** | **9** | **3** | **2** | **4** | **16** |
| **0** | **3** | **1** | **1** | **5** | **0** | **0** | **2** | **4** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **14** | **8** | **10** | **1** | **7** | **9** | **3** | **2** | **4** | **16** |
| **0** | **1** | **1** | **3** | **5** | **0** | **0** | **2** | **4** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **14** | **8** | **10** | **4** | **7** | **9** | **3** | **2** | **1** | **16** |
| **0** | **1** | **1** | **4** | **5** | **0** | **0** | **2** | **3** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **8** | **10** | **4** | **7** | **9** | **3** | **2** | **14** | **16** |
| **3** | **1** | **1** | **4** | **5** | **0** | **0** | **2** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **10** | **8** | **1** | **4** | **7** | **9** | **3** | **2** | **14** | **16** |
| **1** | **1** | **3** | **4** | **5** | **0** | **0** | **2** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **10** | **8** | **9** | **4** | **7** | **1** | **3** | **2** | **14** | **16** |
| **1** | **1** | **0** | **4** | **5** | **3** | **0** | **2** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **10** | **8** | **9** | **4** | **7** | **1** | **3** | **2** | **14** | **16** |
| **1** | **1** | **0** | **4** | **5** | **3** | **0** | **2** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **2** | **8** | **9** | **4** | **7** | **1** | **3** | **10** | **14** | **16** |
| **2** | **1** | **0** | **4** | **5** | **3** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **9** | **8** | **2** | **4** | **7** | **1** | **3** | **10** | **14** | **16** |
| **0** | **1** | **2** | **4** | **5** | **3** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **9** | **8** | **3** | **4** | **7** | **1** | **2** | **10** | **14** | **16** |
| **0** | **1** | **0** | **4** | **5** | **3** | **2** | **1** | **0** | **0** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **2** | **8** | **3** | **4** | **7** | **1** | **9** | **10** | **14** | **16** |
| **2** | **1** | **0** | **4** | **5** | **3** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **8** | **2** | **3** | **4** | **7** | **1** | **9** | **10** | **14** | **16** |
| **1** | **2** | **0** | **4** | **5** | **3** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **8** | **7** | **3** | **4** | **2** | **1** | **9** | **10** | **14** | **16** |
| **1** | **5** | **0** | **4** | **2** | **3** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **8** | **7** | **3** | **4** | **2** | **1** | **9** | **10** | **14** | **16** |
| **1** | **5** | **0** | **4** | **2** | **3** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **7** | **3** | **4** | **2** | **8** | **9** | **10** | **14** | **16** |
| **3** | **5** | **0** | **4** | **2** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **7** | **1** | **3** | **4** | **2** | **8** | **9** | **10** | **14** | **16** |
| **5** | **3** | **0** | **4** | **2** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **7** | **4** | **3** | **1** | **2** | **8** | **9** | **10** | **14** | **16** |
| **5** | **4** | **0** | **3** | **2** | **1** | **0** | **1** | **0** | **0** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **7** | **4** | **3** | **1** | **2** | **8** | **9** | **10** | **14** | **16** |
| **5** | **4** | **0** | **3** | **2** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **2** | **4** | **3** | **1** | **7** | **8** | **9** | **10** | **14** | **16** |
| **2** | **4** | **0** | **3** | **5** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **2** | **3** | **1** | **7** | **8** | **9** | **10** | **14** | **16** |
| **4** | **2** | **0** | **3** | **5** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **4** | **2** | **3** | **1** | **7** | **8** | **9** | **10** | **14** | **16** |
| **4** | **2** | **0** | **3** | **5** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **2** | **3** | **4** | **7** | **8** | **9** | **10** | **14** | **16** |
| **3** | **2** | **0** | **4** | **5** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **3** | **2** | **1** | **4** | **7** | **8** | **9** | **10** | **14** | **16** |
| **0** | **2** | **3** | **4** | **5** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **3** | **2** | **1** | **4** | **7** | **8** | **9** | **10** | **14** | **16** |
| **0** | **2** | **3** | **4** | **5** | **1** | **0** | **1** | **0** | **0** |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **2** | **3** | **4** | **7** | **8** | **9** | **10** | **14** | **16** |
| **3** | **2** | **0** | **4** | **5** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **2** | **1** | **3** | **4** | **7** | **8** | **9** | **10** | **14** | **16** |
| **2** | **3** | **0** | **4** | **5** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **2** | **3** | **4** | **7** | **8** | **9** | **10** | **14** | **16** |
| **3** | **2** | **0** | **4** | **5** | **1** | **0** | **1** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **2** | **3** | **4** | **7** | **8** | **9** | **10** | **14** | **16** |
| **3** | **2** | **0** | **4** | **5** | **1** | **0** | **1** | **0** | **0** |

**Queue**

**testOffer2()**

|  |
| --- |
| **67** |

|  |
| --- |
| **128764** |

|  |
| --- |
| **836763** |

|  |
| --- |
| **2359926** |

|  |
| --- |
| **67** |

|  |
| --- |
| **128764** |

|  |
| --- |
| **836763** |

|  |
| --- |
| **67** |

|  |
| --- |
| **128764** |

|  |
| --- |
| **836763** |

|  |
| --- |
| **2359926** |

**testSize2()**

|  |
| --- |
| **67** |

|  |
| --- |
| **128764** |

|  |
| --- |
| **836763** |

**size = 0**

|  |
| --- |
| **67** |

|  |
| --- |
| **128764** |

|  |
| --- |
| **836763** |

**size = 1**

|  |
| --- |
| **67** |

|  |
| --- |
| **128764** |

|  |
| --- |
| **836763** |

**Size = 2**

|  |
| --- |
| **67** |

|  |
| --- |
| **128764** |

|  |
| --- |
| **836763** |

**Size = 3**

**Stack**

**testPop2()**

|  |
| --- |
| **4272942** |

|  |
| --- |
| **99472** |

|  |
| --- |
| **534** |

|  |
| --- |
| **4272942** |

|  |
| --- |
| **99472** |

|  |
| --- |
| **534** |

|  |
| --- |
| **4272942** |

**testPush2()**

|  |
| --- |
| **4272942** |

|  |
| --- |
| **99472** |

|  |
| --- |
| **534** |

|  |
| --- |
| **2359936** |

|  |
| --- |
| **2359936** |

|  |
| --- |
| **4272942** |

|  |
| --- |
| **99472** |

|  |
| --- |
| **534** |

**Queue**

|  |  |
| --- | --- |
| **setupStage0()** | An empty queue |
| **setupStage1()** | A queue with node with value 67 |
| **setupStage2()** | A queue with three nodes with values 67, 1287643, 836763 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scene** | **Input** | **Result** |
| **Queue** | testOffer0 | setupStage0() | Make the offer function for the node with value 9643 | The front and the back of the Queue is the Node with value 9643 |
| **Queue** | testOffer1 | setupStage1() | Make the offer function for the node with value 4707543 | The node is added in FIFO, it is in the last position of the queue |
| **Queue** | testOffer2 | setupStage2() | Make the offer function for the node with value 2359926 | The node is added in FIFO, it is in the last position of the queue |
| **Queue** | testPoll0() | setupStage0() | Delete the first node of the queue | Null, the queue is empty in this case |
| **Queue** | testPoll1() | setupStage1() | Delete the first node of the queue | Null, because the only node in the queue was removed |
| **Queue** | testPoll2() | setupStage2() | Delete the first node of the queue | 1287643, because it is the value of the node that is in front of the queue. |
| **Queue** | testIsEmpty0() | setupStage0() | A queue empty | True, the queue is empty |
| **Queue** | testIsEmpty1() | setupStage1() | A queue with one node | False, the queue isn´t empty |
| **Queue** | testIsEmpty2() | setupStage2() | A queue with three nodes | False, the queue isn´t empty |
| **Queue** | testPeek0() | setupStage0() | A queue empty | Null, because the queue is empty |
| **Queue** | testPeek1() | setupStage1() | A queue with one node | The value 67 of the node at the front of the queue |
| **Queue** | testPeek2() | setupStage2() | A queue with three nodes | The value 67 of the node at the front of the queue |
| **Queue** | testSize0() | setupStage0() | A queue empty | 0, the queue is empty |
| **Queue** | testSize1() | setupStage1() | A queue with one node | 1, the queue has one node |
| **Queue** | testSize2() | setupStage2() | A queue with three nodes | 3, the queue has three nodes |
| **Queue** | testClear0() | setupStage0() | A queue empty | 0, the queue is empty |
| **Queue** | testClear1() | setupStage1() | A queue with one node | 0, the queue is empty |
| **Queue** | testClear2() | setupStage2() | A queue with three nodes | 0, the queue is empty |

**Stack**

|  |  |
| --- | --- |
| **setupStage0()** | An empty stack |
| **setupStage1()** | A stack with node with value 532 |
| **setupStage2()** | A stack with three nodes with values 532, 99472, 4272942 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scene** | **Input** | **Result** |
| Stack | testPush0() | setupStage0() | Make a push function with node with value 75343 | 75343, the value of the node that is on top of the stack |
| Stack | testPush1() | setupStage1() | Make a push function with node with value 582935 | 582935, the value of the node that is on top of the stack |
| Stack | testPush2() | setupStage2() | Make a push function with node with value 2359926 | 2359926, the value of the node that is on top of the stack |
| Stack | testPop0() | setupStage0() | Make a pop function. Remove the top node of the stack | Null, because the stack is empty |
| Stack | testPop1() | setupStage1() | Make a pop function. Remove the top node of the stack | Null, because the only item was removed from the stack |
| Stack | testPop2() | setupStage2() | Make a pop function. Remove the top node of the stack | 9472, the value of the node that is on top of the stack |
| Stack | testIsEmpty0() | setupStage0() | An empty stack | True, the stack is empty |
| Stack | testIsEmpty1() | setupStage1() | A stack whit one node with value 532 | False, the stack isn´t empty |
| Stack | testIsEmpty2() | setupStage2() | A stack whit three nodes with values 532, 99472, 4272942 | False, the stack isn´t empty |
| Stack | testTop0() | setupStage0() | An empty stack | Null, the stack is empty |
| Stack | testTop1() | setupStage1() | A stack whit one node with value 532 | 532, the value of the node that is in the top of the stack |
| Stack | testTop2() | setupStage2() | A stack whit three nodes with values 532, 99472, 4272942 | 4272942, the value of the node that is in the top of the stack |
| Stack | testSize0() | setupStage0() | An empty stack | 0, the stack is empty |
| Stack | testSize1() | setupStage1() | A stack whit one node with value 532 | 1, the stack has one node |
| Stack | testSize2() | setupStage2() | A stack whit three nodes with values 532, 99472, 4272942 | 3, the stack has three nodes. |

**Hash Table**

|  |  |
| --- | --- |
| **setupStage0()** | An empty hast table |
| **setupStage1()** | A stack with one element, with key 82 and value -4134 |
| **setupStage2()** | A stack with three elements, the first with key -91354 and value 137, the second with key 0 and value 72450 and the third whit key 1820 and value 7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scene** | **Input** | **Result** |
| Hash Table | testDelete0() | setupStage0() | One element to delete with key 2957 | Null, the has table was empty |
| Hash Table | testDelete1() | setupStage1() | An element to delete with the key 82, it was already inside the hash | Null, after searching with the same key |
| Hash Table | testDelete2() | setupStage2() | An element to delete with the key -1820, it was already inside the hash | Null, after searching with the same key |
| Hash Table | testSearch0() | setupStage0() | One element to search with key 42 | Null, the has table was empty |
| Hash Table | testSearch1() | setupStage1() | One element to search with key 82, it was already inside the hash | -4134, the value associated with the key |
| Hash Table | testSearch2() | setupStage2() | One element to search with key 0. it was already inside the hash | 724520, the value associated with the key |
| Hash Table | testInsert0() | setupStage0() | One element to insert, with key 75343 and value 63436 | 63436, the value after searching with the key |
| Hash Table | testInsert1() | setupStage1() | One element to insert, with key 953 and value 23536 | 23536, the value after searching with the key |
| Hash Table | testInsert2() | setupStage2() | One element to insert, with key 450 and value 4235 | 4235, the value after searching with the key |

**Heap**

|  |  |
| --- | --- |
| **setupStage0()** | An empty heap |
| **setupStage1()** | A heap with an array size of three elements and a heap size of three elements. Its values ​​are 1, 2 and 3 in that order |
| **setupStage2()** | A heap with an array size of ten elements and a heap size of ten elements. Its values ​​are 4, 1, 3, 2, 16, 9, 10, 14, 8 y 7 in that order. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scene** | **Input** | **Result** |
| Heap | testBuildMaxHeap0() | setupStage0() | An empty heap | 0 and 0,  heap size and array size |
| Heap | testBuildMaxHeap1() | setupStage1() | A heap with three elements | The array like max heap |
| Heap | testBuildMaxHeap2() | setupStage1() | A heap with three elements | True, where each parent is older than their children |
| Heap | testDecreaseKey0() | setupStage0() | An empty heap | Out of bounds exception, heap is empty |
| Heap | testDecreaseKey1() | setupStage1() | Decrease the value of the object key in the two position, changing the value from 4 to -1 | The reduced key the element in the new position, |
| Heap | testDecreaseKey2() | setupStage2() | Decrease the value of the object key in the zero position, changing the value from 4 to -1 | The reduced key of the element in the new position |
| Heap | testExist0() | setupStage0() | An empty heap | False, the heap is empty |
| Heap | testExist1() | setupStage1() | A heap whit 3 elements | False to position 6 and true to position 2 |
| Heap | testExist2() | setupStage2() | A heap whit 10 elements | False to position 65 and true position 8 |
| Heap | testExtractMaxHeap0() | setupStage0() | An empty heap | Heap under flow exception, the heap is empty |
| Heap | testExtractMaxHeap1() | setupStage1() | A heap whit 3 elements | 1, is the element extracted |
| Heap | testExtractMaxHeap2() | setupStage2() | A heap whit 10 elements | 4, is the element extracted, and 7 is the element extracted |
| Heap | testHeapSort0() | setupStage0() | An empty heap | 0, because the heap is empty |
| Heap | testHeapSort1() | setupStage1() | A heap whit three elements with values 1, 2, 3 in that order | 1, 2, 3, the heap was sorted |
| Heap | testHeapSort2() | setupStage2() | A heap whit ten elements with values 4, 1, 3, 2, 16, 9, 10, 14, 8 y 7 in that order | 1, 2, 3, 4, 7, 8, 9, 10, 14, 16, the ordered heap |
| Heap | testIncreaseKey0() | setupStage0() | An empty heap | Out of bounds exception, heap is empty |
| Heap | testIncreaseKey1() | setupStage1() | Increase the value of the object key in the zero position, changing the value from 6 to 7 | The incremented key of the element in the new position |
| Heap | testIncreaseKey2() | setupStage2() | Increase the value of the object key in the nine position, changing the value from 5 to 6 | The incremented key of the element in the new position |
| Heap | testMaxHeapify0() | setupStage0() | //TODO | //TODO |
| Heap | testMaxHeapify0() | setupStage1() | //TODO | //TODO |
| Heap | testMaxHeapify0() | setupStage2() | //TODO | //TODO |
| Heap | testSwap0() | setupStage0() | An empty heap | Out of bounds exception, heap is empty |
| Heap | testSwap1() | setupStage1() | A heap with three elements and position i and j for the change. | 2, 1, 3. The order after the exchange in these positions |
| Heap | testSwap2() | setupStage2() | A heap with ten elements and position i and j for the change. | 4 ,1 ,14 ,2 ,16 , 9, 10, 3, 8, 7. The order after the exchange in these positions |

**Bank**

|  |  |
| --- | --- |
| **setupStage()** | Creation of 3 people with all their data, including entry date and default cards. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scene** | **Input** | **Result** |
| Bank | orderByNameHeapSort0() | setupStage() | Sort three people by name | True, they are ordered according to the criteria |
| Bank | orderByIdMergeSort() | setupStage() | Sort three people by id | True, they are ordered according to the criteria |
| Bank | orderByTimeQuickSort() | setupStage() | Sort three people by ingress | True, they are ordered according to the criteria |
| Bank | orderByAmountBubuleSort() | setupStage() | Sort three people by amount | True, they are ordered according to the criteria |