|  |  |  |  |
| --- | --- | --- | --- |
| Heap (It implements priority queue methods) | | | |
| Heap H = [h[0]… h[n-1]] ∨ H = Null | | | |
| Invariants:  Their definitions are based on: 0iH.get HeapSize()   * h[n] H * h[0].getPriority() h[i].getPriority 0iH.get HeapSize() * his father of: h[i] * his left son of: h[i] * his right son of: h[i] * The length of each branch must be equal to the height of the tree or its height minus 1 | | | |
| Operation | Input | Output | Type |
| Heap | none | H = [null] | Builder |
| BuildHeap | H | H = [h[0] … h[n-1]] | Modifier |
| Heapify | H i | h[i]h[(i\*2)+1]  h[i]h[(i\*2)+2] | Modifier |
| Insert | H hnew | H = [h[0] … h[n]]  where n is the size before the operation | Modifier |
| Maximum | H | h[0]  “There’s no patients in the unit” | Analyzer |
| ExtractMax | H | h[0]  H = [h[1]… h[n-1]]  where n is the size before the operation  “There’s no patients in the unit” | Modifier |
| IsEmpty | H | true false | Analyzer |
| SearchObject | H  key/id To search    Priority we’re looking for  position | the index of the array where the object is located  “The patient wasn’t found” | Analyzer |
| IncreaseKey | H  oldPriority  key/id To Search  newPriority | H organized with the increase key true  false  “The patient wasn’t found” | Modifier |
| DeleteExact | H  priority  key/id To Search  position | void  print Search exception | Modifier |
| getHeapSize | H | n | Analyzer |
| getArray | H | null  H[h[0]...h[n-1]] | Analyzer |
| getByIndex | H i | H[i] | Analyzer |

|  |  |
| --- | --- |
| Heap | |
| Created the Heap with the empty data | |
| **PRE** | **POST** |
| none | H= [null] |

|  |  |
| --- | --- |
| BuildHeap | |
| Takes a heap and reorganize it to ensure that h[0].getPriority() h[i].getPriority | |
| **PRE** | **POST** |
| H | H |

|  |  |
| --- | --- |
| Heapify | |
| Takes a heap and reorganize it to ensure that h[i]h[(i\*2)+1] h[i]h[(i\*2)+2] | |
| **PRE** | **POST** |
| H, i | H |

|  |  |
| --- | --- |
| Insert | |
| Takes a heap and insert hnew into it and increases the size | |
| **PRE** | **POST** |
| H, hnew | H = [h0 … hn]  where n is the size before the insertion |

|  |  |
| --- | --- |
| Maximum | |
| Return h0 without taking it out of the Heap | |
| **PRE** | **POST** |
| H | h[0]  Throws exception “There’s no patients in the unit” |

|  |  |
| --- | --- |
| ExtractMax | |
| Return h0 and takes it out of the heap reducing its size by one | |
| **PRE** | **POST** |
| H | H = [h1 … hn-1]  where n is the size before the insertion  Throws exception “There’s no patients in the unit” |

|  |  |
| --- | --- |
| IsEmpty | |
| Returns true if the size is empty, false if it doesn’t | |
| **PRE** | **POST** |
| H | true  false |

|  |  |
| --- | --- |
| SearchObject | |
| Return hi without taking it out of the Heap | |
| **PRE** | **POST** |
| H, key/id, priority, position | i (the position of the array where the object is located)  Throws exception “The patient wasn’t found” |

|  |  |
| --- | --- |
| IncreaseKey | |
| Increases they key/priority from an Object of the heap and re organizes it | |
| **PRE** | **POST** |
| H, oldPriority, key/id To Search, newPriority | H  Throws exception “The patient wasn’t found” |

|  |  |
| --- | --- |
| DeleteExact | |
| Search the object in the Heap and delete it from it, decreases the size in one | |
| **PRE** | **POST** |
| H, priority, key/id, position | void  print Search Exception |

|  |  |
| --- | --- |
| getHeapSize | |
| Return the size of the Heap | |
| **PRE** | **POST** |
| H  H has been initialized | n |

|  |  |
| --- | --- |
| getArray | |
| Return the array in the Heap | |
| **PRE** | **POST** |
| H  H has been initialized | null  H[h[0]...h[n-1]] |

|  |  |
| --- | --- |
| getByIndex | |
| Search and return a specific in the Heap | |
| **PRE** | **POST** |
| H i | H[i] |

|  |  |  |  |
| --- | --- | --- | --- |
| Stack | | | |
| Stack S= [Sn… S1] v S=Null | | | |
| Invariants=  [SI ∈ S ∧ Sn ∈ S] → SI ≠ Sn  Sn is always get out first than S1 | | | |
| Operación | Entrada | Salida | Tipo |
| Stack | none | **S** = [null] | Builder |
| empty | **S** = [null] v **S** = [Sn…S1] | true or false | Analyzer |
| peak | **S** = [null] v **S** = [Sn…S1] | “The stack is empty” v  Sn | Analyzer |
| pop | **S** = [null] v **S** = [Sn…S1] | “The stack is empty” v  Sn ∧ S=[S(n-1)… S1 ] | Modifier |
| push | Sn | S=[S(n+1)… S1 ] | Modifier |
| getSize | **S** = [null] v **S** = [Sn…S1] | n | Analyzer |

|  |  |
| --- | --- |
| Stack | |
| Created the Stack with the empty data | |
| **PRE** | **POST** |
| none | S= [null] |

|  |  |
| --- | --- |
| empty | |
| Check if the Stack has an element | |
| **PRE** | **POST** |
| S= [null] v S= [Sn…S1]  S has been initialized | true v false |

|  |  |
| --- | --- |
| peak | |
| show the item on top | |
| **PRE** | **POST** |
| S= [null] v S= [Sn…S1]  S has been initialized | “The stack is empty” v Sn |

|  |  |
| --- | --- |
| pop | |
| show and remove the item on top | |
| **PRE** | **POST** |
| **S** = [null] v **S** = [Sn…S1]  S has been initialized | “The stack is empty” v Sn ∧ S=[S(n-1)… S1 ] |

|  |  |
| --- | --- |
| push | |
| Insert a element in the stack | |
| **PRE** | **POST** |
| Sn | S=[S(n+1)… S0 ] |

|  |  |
| --- | --- |
| getSize | |
| return the size of the stack | |
| **PRE** | **POST** |
| **S** = [null] v **S** = [Sn…S1]  S has been initialized | n |

|  |  |  |  |
| --- | --- | --- | --- |
| HashTable | | | |
| H = [(h1,..,h1.n),....,(hn,..,hn.n) ] v H= [null] | | | |
| Invariants=  [h[i] ∈ H ∧ h[j] ∈ H] → i ≠ j v i = j  0 i j < n  The length of H is a n fixed number  The elements of the HashTable have different keys for each element: h[i].key  ≠ h[j].key | | | |
| Operación | Entrada | Salida | Tipo |
| HashTable | none | H | Builder |
| chainedHashInsert | h | none | Modifier |
| chainedHashDelete | **K** | null  v  hn ∧ H = [(h1,..,h1.n),....,(hn,..,hn.n) ] | Modifier |
| chainedHashSearch | **K** | “null”  v  hn | Analyzer |
| hashFunction | **K** | hashCode | Modifier |

|  |  |
| --- | --- |
| HashTable | |
| Created the HashTable with empty data | |
| **PRE** | **POST** |
| none | H=[null..] |

|  |  |
| --- | --- |
| chainedHashInsert | |
| Insert a element in the HashTable | |
| **PRE** | **POST** |
| h  h dont have the same id than the others h | H=[..hn..] |

|  |  |
| --- | --- |
| chainedHashDelete | |
| Delete and return a specific element in the HastTable | |
| **PRE** | **POST** |
| K  K= key | null  v  hn ∧ H = [(null),....,(hn,..,hn.n) ] |

|  |  |
| --- | --- |
| chainedHashSearch | |
| Search and return a specific element in the HastTable | |
| **PRE** | **POST** |
| K  K= key | “null”  v  hn |

|  |  |
| --- | --- |
| hashFunction | |
| Receive a key and create a hashCode for the HashTable | |
| **PRE** | **POST** |
| K  K= key | hashCode  hashCode<HashTable size |