|  |
| --- |
| **TAD Priority Queue** |
| Priority Queue = {Elements = 〈〉, Heap Size = 〈heapSize〉}  *∀ n ∈ N* |
| {inv*:*   * *For max heap: ∀ i, j ∈ [1, n], where n is the number of elements, if i < j, then priority () ≥ priority ().* * *For min-heap: ∀ i, j ∈ [1, n], where n is the number of elements, if i < j, then priority () ≤ priority ()}* |
| * CreatePriorityQueue: PriorityQueue * InsertElement: PriorityQueue X Element PriorityQueue * ExtractElement: PriorityQueue Element * FirstElement: PriorityQueue Element * LastElement: PriorityQueue Element * ModifyValue: PriorityQueue X Index X Value PriorityQueue * OrganizeElements: PriorityQueue X Index PriorityQueue |
| **CreatePriorityQueue [***PriorityQueue***] – Constructor**  *“Creates a new and empty priority queue”.*  *{Pre: None}.*  *{Post: Returns a PriorityQueue without elements.}*  **InsertElement [***insert***] – Modifier**  *“Given an element by parameter it inserts it at the end of the queue for reorganized itself by the priority”.*  *{Pre: There must be a priority queue, and the element must be of a compatible type}.*  *{Post: Returns the updated PriorityQueue}.*  **ExtractElement [***extractMax/Min***] - Modifier**  *“Removes and return the element with the least or highest priority depending on the type of organization, else it returns null”.*  *{Pre: There must be a priority queue}.*  *{Post: Returns the updated PriorityQueue and the element removed.}*  **FirstElement [***front***] - Analyser**  “*Returns the element with the least or highest priority depending on the type of organization, else it returns null”.*  *{Pre: There must be a priority queue}.*  *{Post: Returns the elements}.*  **LastElement [***back***] - Analyser**  “*Returns the element with the least or highest priority depending on the type of organization, else it returns null”.*  *{Pre: There must be a priority queue}.*  *{Post: Returns the elements}.*  **ModifyKey [***increaseKey/decreaseKey***] – Modifier**  *“Changes the key or value of an element given the index by parameter while maintaining the priority order, but first checks if the change has sense based on the organization of the value priorities or else do nothing”.*  *{Pre: There must be a priority queue, the index must be in-bounds and the value must be of a compatible type}.*  *{Post: Returns the updated PriorityQueue}.*  **OrganizeElements [***heapify***] – Modifier.**  *“Reorganizes the elements ‘below on the heap’ in the priority queue, typically used after modifying a value at a specific index to maintain the priority order.”.*  *{Pre: There must be a priority queue and the index must be in-bounds}.*  *{Post: Returns the updated PriorityQueue}.* |