|  |
| --- |
| TAD STACK |
| STACK = Pila de elementos con principio LIFO |
| (inv: La longitud de la pila no puede ser negativa) |
| Operaciones primitivas:   * Stack - ->Stack * pop: Satck ->Stack * push: Stack x Element ->Stack * peek: Stack ->Element * isEmpty: Stack ->Boolean |

|  |
| --- |
| TAD PRIORITY QUEUE |
| PRIORITY QUEUE = Cola de elementos que se procesan según su prioridad |
| (inv: Ningún elemento puede ser nulo, la longitud de la cola no puede ser negativa) |
| Operaciones primitivas:   * PriorityQueue- - ->PriorityQueue * BuildMaxHeapfy: PriorityQueue ->PriorityQueue * maxHeapfy: value ->PriorityQueue * extractMax: - ->Element * inserElement: priority x Element ->PriorityQueue * showMax: - ->Element * contPatient: - ->Contador * showQueue - ->Element * deleteElement Element ->PriorityQueue * increaseKey Elemetn x Key ->PrioirityQueue |

|  |
| --- |
| TAD HASHTABLE |
| HASHTABLE = Estructura de datos que permite almacenar llaves y valores |
| (inv: El value o valor que queramos almacenar siempre deberá tener una key o llave) |
| Operaciones primitivas:   * HashTable Sizelist ->Stack * insert: Key x Patient ->HashTable * search: Key ->Patient * delete: Key ->HashTable * hashKey: Key ->HashCode |