TAD Data Estructures

TAD Hash Table		
Hash Table = {Arraylist<HashNode<K, V>> nodes, Bucket=<bucket>, Size=<size>		
$\{\text{inv: k1} = \text{v1}, \text{k2} = \text{v2}, \text{k3} = \text{v3}, \text{k4} = \text{v4} \ldots . \text{kn} = \text{vn}\}$		
Operaciones Primitivas:		
• HashTable: • Insert: • Search: • Delete: • GetIndex:	 Key, Value Key Key Key	→HashTable →HashTable →Value →Node →Integer

HashTable()
"Create a new empty Hash Table"
{pre: TRUE}
{post: new empty Hash Table}

Insert(K key, V value)
"Add a new Node to the Hash Table"
{pre: Hash Table has to be created}
{post: new key added to the Hash Table}

Search(K key)
"Search if the key is inside of the Hash Table"
{pre: Hash Table has to be created and Hash Table != empty }
{post: value that was stored in the node identified with the input key}

Delete(K key)
"Delete a particular node in the Hash Table according to the key"
{pre: Hash Table has to be created and k is in the Hash Table}
{post: key doesn't exist in the Hash Table and slot is null}

GetIndex(k Key)
"Calculates the index where the node should be added according to its key"
{pre: Hash Table has to be created}
{post: index where the node should be added according to its key}

TAD QUEUE		
Queue ={ Node<T>= first}		
{inv: If size $== 1, n_0 ==$ top and tail}		
Operaciones Primitivas:		
• Queue: • Add: • Poll: • Peek:	Value	→Queue →Queue →Value →Value

Queue()
"Create a new empty Queue"
{pre: TRUE}
{post: new empty Queue}

Add(T object)
"Add a new element to the Queue"
{pre: Queue has to be created}
{post: new element added to the Queue}

Poll()
"Returns and removes the element at the front to the Queue"
{pre: Queue has to be created and Queue != empty}
{post: Returns the first node value and if the size $= 1$, Queue $==$ empty ||if the size ≥ 2, the second node is now the first }

Peek()
"Returns the element at the front to the Queue"
{pre: Queue has to be created and Queue != empty}
{post: Returns the first node value}

TAD Stack		
Stack= {Node<T> =last}		
$\{inv: Si\ nodo == n_{m-1} \rightarrow next == null\}$		
Operaciones Primitivas:		
• Stack: • Push: • Pop: • Peek:	Value	→Stack →Stack →Value →Value

Stack()
"Create a new empty Stack"
{pre: TRUE}
{post: new empty Stack}

Push(T object)
" Add a new element to the Stack"
{pre: Stack has to be created}
{post: new element added to the Queue}

Pop()
"Returns and removes the element at the end of the Stack"
{pre: Stack has to be created and Stack != empty }
{post: Returns the last node value and if the size $= 1$, Stack $==$ empty ||if the size ≥ 2, the n_{m-2} node is now the last }

Peek()
"Returns the element at the end to the Stack"
{pre: Stack has to be created and Stack != empty}
{post: Returns the last node value}