TAD Heap

Heap = {Parent = <parent>, Parent.RightChildren = <rightChildren>, Parent.leftChildren = <leftChildren>, Value = <value>}

Invariant: Parent != nill, RightChildren != nill, leftChildren != nill ∧ (Father.leftChildren.value ≤ Father.value ≥ Father.RightChildren.value) v (Father.leftChildren.value ≥ Father.value ≤ Father.RightChildren.value)

Construction operations:
*Create : → Heap
Modifier operations:
*addElement: HeapxValue → Heap
*remove: HeapxValue → Heap
Operaciones analizadoras:
*isEmpty: DoublyLinkedList → booleano
*size: DoublyLinkedList → Integer

Create (value)
"Creates an element of the Heap with the left children and right children empty, but with a defined value"

{pre: TRUE }

{post: elementHeap = {RightChildren = <nill>, LeftChildren = <nill>, Value=<value>}

addElement (value)
"Inserts an element on the Heap structure having into account its value."

{pre: TRUE }

{post: Parent = {RightChildren=<nill>, LeftChildren=<nill>, Value=<value>}}

remove()
"Removes the Parent element from the heap, leaving the element with the hightest(lowest) value in first place, replacing the Parent".

{ pre: the heap has at least one element }

{ post: Parent is returned}

isEmpty(Heap):

"Informs if the heap is empty."

{pre: TRUE}
{pre: Heap={Parent:<parent>,...}

{post: False if the Heap.parent!= nil, True otherwise}

size(Heap):

"Returns an Integer that represents the number of elements currently inserted in the heap."

{pre: TRUE}
{pre: Heap={Parent:<parent>,...}

{post: n | n ∈ Z+}