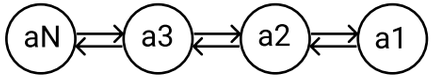
ADT Queue

Queue = {aN... a3, a2,a1}

-aN is the latest element added, and a1 is the first element that was added.



Inv: { aN= last; a1= first}
first in is first out

Primitive operations:

-createQueue:	->Queue
-Enqueue: Element x Queue	->Queue
-isEmpty: Queue	->Boolean
-front: Queue	->Element
-getLatest: Queue	->Element
-dequeue: Queue	->Element
-size: Queue	->Whole Number

Enqueue(T newItem) : Modifier

"Enqueue a new item to the bottom of the line"

{ pre: Queue initializated }
{ post: Queue size increment +1 and a new item added before the last element: newItem <- TopElement }

Dequeue( ) : Modifier

"Dequeue the last element from the Queue and delete it"

{ pre: The queue must stay initializated }
{ post: Queue size decrement -1 and a the last element will be removed: LastElement / TopElement }

IsEmpty( ): Analyzer

"Check if the queue is empty or not and return a boolean value"

{ pre: TRUE }
{ post: True if the queue it's empty or False if isn't empty }

Size( ) : Analyzer

"Return the total length of the queue in a integer variable"

{ pre: The queue must stay initializated }
{ post: Integer: The total size (Length) of the queue counting the enlazed nodes }

Front( ):Analayzer

"Method to return an element that is opposite another"

{ pre: The Queue isn't Empty and it must stay initializated }
{ post: Element: The element of front of other }

CreateQueue( ) : Constructor

"Create (Initializate) a new empty queue to add new elements"

{ pre: TRUE }
{ post: NewQueue: The new created queue ready to add new elements }

GetLatest( ): Analyzer

"Method to return the last item in the list without deleting it"

{ pre: The queue must stay initializated }
{ post: Element: The Element at end of the Queue }