**Example of a reasonably documented CLASS definition.**

**All lab programs must be documented at least this well.**

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

//\* Class name : DblLinkListClass \*

//\* \*

//\* Written by : John Urrutia (C) 2013, All rights reserved \*

//\* \*

//\* Purpose : Constructs an unordered Double link list with \*

//\* no restriction on duplicate data elements. \*

//\* \*

//\* Inputs : objects to add, insert, change, delete or find \*

//\* Display list request \*

//\* \*

//\* Outputs : Return boolean result or found node to caller \*

//\* \*

//\* Methods : AddNode(), InsertNode(), DeleteNode(), \*

//\* FindNode() and DisplayList() \*

//\* \*

//\*---------------------------------------------------------------------\*

//\* Change Log: \*

//\* Revision \*

//\* Date Changed Rel Ver Mod Purpose \*

//\* 06/06/12 jurrutia 000.000.000 Initial release of program \*

//\* 03/28/13 jurrutia 000.001.000 Modified AddNode(), InsertNode(),\*

//\* Added DisplayList \*

//\* \*

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace LinkedListNew

{

public class DblLinkListClass

{

protected DblLinkListNode header;

public DblLinkListClass()

{

header = new DblLinkListNode(null);

}

// All nodes are added by default after the header node

// can be overridden by providing the node to insert after.

public void AddNode(object data, DblLinkListNode currNode = null )

{

DblLinkListNode newNode = new DblLinkListNode(data);

if (currNode == null)

currNode = header;

newNode.prevLink = currNode;

newNode.nextLink = currNode.nextLink;

currNode.nextLink = newNode;

if(newNode.nextLink != null)

newNode.nextLink.prevLink = newNode;

}

// Populates and inserts a new node into the list after a specified node

// can optionally insert before the specified node.

public bool InsertNode(object data, object current, char position ='A')

{

DblLinkListNode newNode = new DblLinkListNode(data),

currNode = new DblLinkListNode();

currNode = FindNode(current);

if(currNode==null)

return false;

if (char.ToUpper(position) == 'A')

currNode = currNode.prevLink;

AddNode(data, currNode);

return true;

}

// Finds the node with the specified data and changes it

public bool ChangeNode(object oldData, object newData)

{

DblLinkListNode newNode = FindNode(oldData);

if (newNode == null)

{

return false;

}

else

{

newNode.nodeElement = newData;

return true;

}

}

// Finds the node with the specified data and deletes it

public bool DeleteNode(object data)

{

DblLinkListNode currNode = FindNode(data);

if (currNode == null) return false;

currNode.prevLink.nextLink = currNode.nextLink;

currNode.nextLink.prevLink = currNode.prevLink;

return true;

}

// Finds the node with the specified data.

// Returns a null node if not found.

public DblLinkListNode FindNode(object data)

{

DblLinkListNode nodeFind = header.nextLink;

while ( nodeFind.nodeElement != null)

{

if (nodeFind.nodeElement == data)

{

return nodeFind;

}

else

{

nodeFind = nodeFind.nextLink;

}

}

return null;

}

// Displays all data element in the list Backward (LIFO order) or Forward (FIFO order)

public void DisplayList(DblLinkListClass dblLinkList, char Direction = 'B')

{

DblLinkListNode node = header;

if(Direction == 'B')

{

//Starts with the header node

do

{

node = node.nextLink;

Console.WriteLine(node.nodeElement);

}

while (node.nextLink != null);

}

else

{

//Starts with the last node

while (node.nextLink != null)

node = node.nextLink;

do

{

Console.WriteLine(node.nodeElement);

node = node.prevLink;

}

while (node.prevLink != null);

}

}

}

}