Assessment 3 part 3 Code

//Robert Lothian

//HND Software Development 2

//26.04.22

//This excercise develops a stack class using a linked list and implenting an ADT interface

using System;

namespace Assessment\_3.\_3

{

internal class Program

{/// <summary>

/// this creates the interface that will be used with intLinkedList

/// </summary>

interface StackADT

{

void push(int value);

int pop();

bool isEmpty();

void display();

int size();

}

/// <summary>

/// This is the intLinkedList that we will use to create a stack

/// </summary>

class intLinkedList

{

/// <summary>

/// This creates variables

/// </summary>

class Node

{

public int value;

public Node nextnode;

/// <summary>

/// This instanciates the variables

/// </summary>

/// <param name="v"></param>

public Node(int v)

{

value = v;

nextnode = null;

}

/// <summary>

/// This is used to add a node

/// </summary>

/// <param name="v"></param>

public void add(int v)

{

if (nextnode == null)

nextnode = new Node(v);

else

nextnode.add(v);

}

}

/// <summary>

/// This creates a Node variable and int count

/// </summary>

private Node start;

private int count;

/// <summary>

/// Intanciates variables

/// </summary>

public intLinkedList()

{

start = null;

count = 0;

}

/// <summary>

/// This is used to add to the stack

/// </summary>

/// <param name="value"></param>

public void addLast(int value)

{

if (start == null)

{

start = new Node(value);

}

else

{

start.add(value);

}

count++;

}

/// <summary>

/// Used if the stack is empty

/// </summary>

/// <returns></returns>

public bool isEmpty()

{

return (count == 0);

}

/// <summary>

/// Used to add an element to the beginning of the stack

/// </summary>

/// <param name="value"></param>

public void addFirst(int value)

{

if (isEmpty())

{

addLast(value);

}

else

{

Node temp = start;

start = new Node(value);

start.nextnode = temp;

count++;

}

}

/// <summary>

/// Used to remove the first element from the stack, used for FIFO

/// </summary>

/// <returns></returns>

/// <exception cref="Exception"></exception>

public int removeFirst()

{

int rvalue;

if (isEmpty())

{

throw new Exception("list is empty");

}

else

{

rvalue = start.value;

if (start.nextnode == null)

start = null;

else

start = start.nextnode;

count--;

}

return rvalue;

}

/// <summary>

/// Removes last element from stack, used for FILO

/// </summary>

/// <returns></returns>

/// <exception cref="Exception"></exception>

public int removeLast()

{

int rvalue;

if (isEmpty())

{

throw new Exception("list is empty");

}

else

{

if (count == 1)

{

rvalue = start.value;

start = null;

}

else

{

Node temp = start.nextnode;

Node prev = start;

while (temp.nextnode != null)

{

prev = temp;

temp = temp.nextnode;

}

rvalue = temp.value;

prev.nextnode = null;

}

count--;

}

return rvalue;

}

/// <summary>

/// Displays elements in stack

/// </summary>

public void display()

{

if (count == 0)

Console.WriteLine("list is empty");

else

{

Console.WriteLine("list has " + count + " items");

Node curnode = start;

while (curnode != null)

{

Console.WriteLine("value: " + curnode.value);

curnode = curnode.nextnode;

}

}

}

/// <summary>

/// To check the number of elements in a stack

/// </summary>

/// <returns>Number of items in the stack</returns>

public int size()

{

return count;

}

}

/// <summary>

/// Creates a class that uses the Interface

/// </summary>

class MyLinkedStack : StackADT

{

/// <summary>

/// Creates a variable of intLinkedList

/// </summary>

public intLinkedList list;

/// <summary>

/// instaciates element

/// </summary>

public MyLinkedStack()

{

list = new intLinkedList();

}

/// <summary>

/// calls display in intLinkedList

/// </summary>

public void display()

{

list.display();

}

/// <summary>

/// calls isEmpty in intLinkedList

/// </summary>

/// <returns>if the stack is empty</returns>

public bool isEmpty()

{

return list.isEmpty();

}

/// <summary>

/// calls pop in intLinkedList

/// </summary>

/// <returns>stack with last element removed</returns>

public int pop()

{

return list.removeLast();

}

/// <summary>

/// calls addLast in intLinkedList

/// </summary>

/// <param name="value"></param>

public void push(int value)

{

list.addLast(value);

}

/// <summary>

/// calls size in intLinkedList

/// </summary>

/// <returns>the size of the stack</returns>

public int size()

{

return list.size();

}

}

/// <summary>

/// driver to enter parameters and test data

/// </summary>

static void linkedstackdriver()

{

MyLinkedStack astack = new MyLinkedStack();

Console.WriteLine("testing Stack ");

Console.WriteLine("testing is empty " + astack.isEmpty());

for (int i = 1; i < 6; i++)

astack.push(i);

Console.WriteLine("num values in stack: " + astack.size());

astack.display();

Console.WriteLine("popping value" + astack.pop());

Console.WriteLine("value 5 should have been removed");

astack.display();

}

/// <summary>

/// main method used to call driver

/// </summary>

/// <param name="args"></param>

static void Main(string[] args)

{

linkedstackdriver();

}

}

}