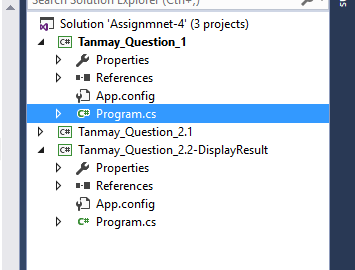
Name: Tanmay Parmar

Student ID: 300872344

Section: 004

Assignment 4



Question 1:

/\*

\* Tanmay Parmar

\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Question\_1

{

class Program

{

static void Main(string[] args)

{

string separator = "---------------------------------------------------------";

// Create stack called stackTest

Stack<string> stackTest = new Stack<string>();

// After adding the first stack we cannot remove it

// untill we remove stack 2 and stack 3

stackTest.Push("Stack 1");

stackTest.Push("Stack 2");

stackTest.Push("Stack 3");

// Displays stackTest

Console.WriteLine("All elements are in the stack:");

foreach (string i in stackTest)

{

Console.WriteLine(i);

}

Console.WriteLine(separator);

Console.WriteLine("Elements in the stack after stackTest.Peek():");

string peek = stackTest.Peek(); // Returns the object at the top of the Stack without removing it.

Console.WriteLine(peek);

Console.WriteLine(separator);

Console.WriteLine("Element that got removed from the stack after stackTest.Pop(): ");

Console.WriteLine(stackTest.Pop()); // Removes and returns the object at the top of the Stack.

Console.WriteLine(separator);

Console.WriteLine("Elements in the stack after stackTest.Peek():");

string peekAfterPop = stackTest.Peek(); // Returns the object at the top of the Stack without removing it.

Console.WriteLine(peekAfterPop);

Console.WriteLine(separator);

Console.WriteLine();

Console.WriteLine();

// Create an object of a Queue<string> called queue

Queue<string> queue = new Queue<string>();

queue.Enqueue("Tanmay"); // Add to the back

queue.Enqueue("Himanshu");

queue.Enqueue("Keval");

queue.Enqueue("Nirav");

Console.WriteLine("All elements in the queue:");

foreach (var item in queue.ToArray())

{

Console.WriteLine(item);

}

Console.WriteLine(separator);

Console.WriteLine("Elements in the queue after queue.Dequeue():");

queue.Dequeue();//removing the first item from the as per queue works as per FIFO

foreach (var item in queue.ToArray())

{

Console.WriteLine(item);

}

Console.WriteLine(separator);

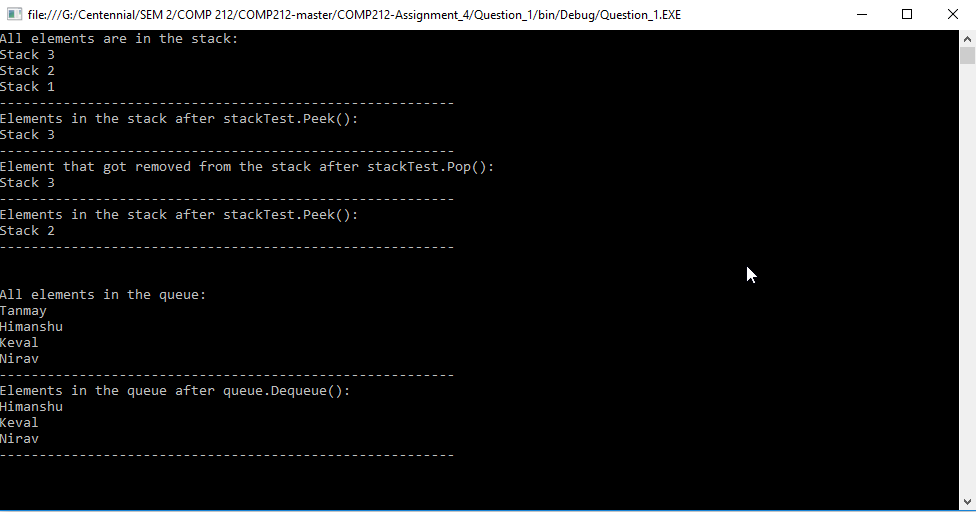
Console.Read();

}

}

}

Screenshot



Question 2:

EmmtyListException.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Question\_2

{

public class EmptyListException: Exception

{

//Empty list

public EmptyListException() : base("The list is empty") { }

// Constructor with one parameter

public EmptyListException(string name): base($"The {name} is empty"){ }

// Constructor with two parameter

public EmptyListException(string exception, Exception inner): base(exception, inner) { }

}

}

List.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

namespace Question\_2

{

public class List

{

private ListNode firstNode;

private ListNode lastNode;

private string name; // string like "list" to display

// construct empty List with specified name

public List(string listName)

{

name = listName;

firstNode = lastNode = null;

}

// construct empty List with "list" as its name

public List() : this("list") { }

// Insert object at front of List. If List is empty,

// firstNode and lastNode will refer to same object.

// Otherwise, firstNode refers to new node.

public void InsertAtFront(object insertItem)

{

if (IsEmpty())

{

firstNode = lastNode = new ListNode(insertItem);

}

else

{

firstNode = new ListNode(insertItem, firstNode);

}

}

// If List is empty then Insert object at end of List.

// firstNode and lastNode will refer to same object.

// else, lastNode's Next property refers to new node.

public void InsertAtBack(object insertItem)

{

if (IsEmpty())

{

firstNode = lastNode = new ListNode(insertItem);

}

else

{

lastNode = lastNode.Next = new ListNode(insertItem);

}

}

// removing first node from List

public object RemoveFromFront()

{

if (IsEmpty())

{

throw new EmptyListException(name);

}

object removeItem = firstNode.Data; // retrieve data

// reset firstNode and lastNode references

if (firstNode == lastNode)

{

firstNode = lastNode = null;

}

else

{

firstNode = firstNode.Next;

}

return removeItem; // return removed data

}

// removing last node from List

public object RemoveFromBack()

{

if (IsEmpty())

{

throw new EmptyListException(name);

}

object removeItem = lastNode.Data; // retrieve data

// reset firstNode and lastNode references

if (firstNode == lastNode)

{

firstNode = lastNode = null;

}

else

{

ListNode current = firstNode;

while (current.Next != lastNode)

{

current = current.Next; // move to next node

}

// lastnode is new current node

lastNode = current;

current.Next = null;

}

return removeItem; // return removed item

}

// true if list is empty

public bool IsEmpty()

{

return firstNode == null;

}

// List of output contents

public void Display()

{

if (IsEmpty())

{

Console.WriteLine($"Empty {name}");

}

else

{

Console.Write($"The {name} is: ");

ListNode current = firstNode;

// output current node data while not at end of list

while (current != null)

{

Console.Write($"{current.Data} ");

current = current.Next;

}

Console.WriteLine("\n");

}

}

// Sort data method

public void Sort()

{

ArrayList arrayList = new ArrayList();

while (IsEmpty() != true)

{

arrayList.Add(this.RemoveFromFront());

}

arrayList.Sort();

//inserting sorted data

foreach (int item in arrayList)

{

InsertAtBack(item);

}

}

}

}

ListNode.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Question\_2

{

class ListNode

{

public object Data { get; private set; }

public ListNode Next { get; set; }

// constructor to create ListNode that refers to dataValue

// and is last node in list

public ListNode(object dataValue) : this(dataValue, null) { }

// constructor to create ListNode that refers to dataValue

// and refers to next ListNode in List

public ListNode(object dataValue, ListNode nextNode)

{

Data = dataValue;

Next = nextNode;

}

}

}

DisplayResult

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Question\_2;

namespace Question\_2\_DisplayResult

{

class Program

{

static void Main(string[] args)

{

string separator = "---------------------------------------------------------";

List List1 = new List();

List List2 = new List();

List1.InsertAtFront(7);

List1.InsertAtBack(5);

List1.InsertAtBack(2);

List1.InsertAtBack(4);

Console.WriteLine("First list.......");

List1.Display();

//displaying sorted list 1

List1.Sort();

Console.WriteLine("Sort List 1............");

List1.Display();

Console.WriteLine(separator);

Console.WriteLine("Second list...."+ "/n");

List2.InsertAtFront(55);

List2.InsertAtBack(42);

List2.InsertAtBack(1);

List2.Display();

//displaying sorted list

Console.WriteLine("Sort List 2..........");

List2.Sort();

List2.Display();

Console.WriteLine(separator);

Console.WriteLine("Merge first and second list ......");

Merge(List1, List2).Display();

}

// Merge two lists

public static List Merge(List list1, List list2)

{

while (!list2.IsEmpty())

{

list1.InsertAtBack(list2.RemoveFromFront());

//sort the merge list

list1.Sort();

}

Console.ReadLine();

//return list

return list1;

}

}

}

Screenshot

