3.4 Stacks



This section will guide you to:

* Create a Windows Console project in Visual Studio to implement a stack using a linked list
* Create a class, Node, that will store data for the stack
* Create a class, Stack, that will implement a stack
* Create a method, runApp(), that will show the use of a stack

**Development Environment**

* Visual Studio 2019 Community Version

This guide has seven subsections, namely:

* + 1. Creating a Windows Console project in Visual Studio to implement a stack using a linked list
    2. Creating a class, Node, that will store data for the stack
    3. Creating a class, Stack, that will implement a stack
    4. Adding a method, runApp(), that will show the use of a stack
    5. Building the project
    6. Publishing and running the project
    7. Pushing the code to your GitHub repositories

**Step** **3.4.1:** Creating a Windows Console project in Visual Studio to implement a stack using a linked list

* Open Visual Studio.
* From the top menu, select **File->New->Project.**
* In **Create A New Project** screen, select **Console app (.NET Core)** from the list of available project types and click on **Next.**
* Enter **Project Name** as **Phase1Section4.9** and click on **Create.**
* This will create the files for a Windows Console project.

**Step** **3.4.2:** Creating a class, Node, that will store data for the stack

* In the **Solution Explorer** window, right click **Phase1Section4.9** and choose **Add->Class.**
* Enter Class name as **Node.cs** and click **Add.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Text;

**namespace** Phase1Section4.\_9

{

**public** **class** Node

{

**private** **string** data;

**private** Node next = **null**;

**private** Node prev = **null**;

**public** **string** Data

{

**get** { **return** **this**.data; }

**set** { **this**.data = value; }

}

**public** Node Next

{

**get** { **return** **this**.next; }

**set** { **this**.next = value; }

}

**public** Node Prev

{

**get** { **return** **this**.prev; }

**set** { **this**.prev = value; }

}

}

}

**Step 3.4.3:** Creating a class, Stack, that will implement a stack

* In **Solution Explorer** window, right click **Phase1Section4.9** and choose **Add->Class**
* Enter Class name as **Stack.cs** and click **Add.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Text;

**namespace** Phase1Section4.\_9

{

**public** **class** Stack

{

**private** Node root = **null**;

**private** Node current = **null**;

**private** **int** length;

**public** Node Root

{

**get** { **return** **this**.root; }

**set** { **this**.root = value; }

}

**public** **int** Length

{

**get** { **return** **this**.length; }

**set** { **this**.length = value; }

}

**public** **void** push(Node node)

{

node.Next = **null**;

**if** (root == **null**)

{

root = node;

length = 1;

current = root;

root.Prev = **null**;

}

**else**

{

current.Next = node;

node.Prev = current;

current = node;

length++;

}

}

**public** Node pop()

{

**if** (current == **null**)

{

**return** **null**;

} **else** **if** (current == root)

{

Node n = current;

current = **null**;

root = **null**;

length = 0;

**return** n;

} **else**

{

Node n = current;

current = current.Prev;

current.Next = **null**;

length--;

**return** n;

}

}

**public** Node peek()

{

**if** (current == **null** || current.Next == **null**)

**return** **null**;

**else**

**return** current.Next;

}

**public** Node getRoot()

{

current = root;

**return** **this**.root;

}

**public** **int** getLengthOfStack()

{

**return** **this**.length;

}

**public** **string** listContents()

{

**string** retVal = "";

current = root;

**while** (current != **null**)

{

retVal += current.Data + " ";

current = current.Next;

}

current = root;

**return** retVal;

}

}

}

**Step 3.4.4:** Adding a method, runApp(), that will show the use of a stack

* Select **Program.cs** as the current Code tab.
* Enter the following code:

**using** System;

**namespace** Phase1Section4.\_9

{

**class** Program

{

**static** **void** Main(**string**[] args)

{

runApp();

}

**public** **static** **void** runApp()

{

Stack stack = **new** Stack();

Console.WriteLine("length of stack=" + stack.getLengthOfStack());

**if** (stack.peek() == **null**)

Console.WriteLine("Stack contains nothing");

**for** (**int** i =0; i < 10; i ++)

{

Node n = **new** Node();

n.Data = i.ToString();

stack.push(n);

Console.WriteLine("Length of stack=" + stack.getLengthOfStack());

}

Node node = stack.pop();

Console.WriteLine("Popped " + node.Data);

node = stack.pop();

Console.WriteLine("Popped " + node.Data);

Console.WriteLine(stack.listContents());

}

}

}

**Step 3.4.5:** Building the project

* From the top menu, choose **Build->Build Solution.**
* If any compile errors are shown, fix them as required.

**Step 3.4.6:** Publishing and running the project

* From the top menu, select **Debug->Start Without Debugging.**
* This will execute the program in a console window.

**Step 3.4.7:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master