OEL 01

Question 01:

Techviz is a company who wants to launch their new web browser “RobustGo” which would be more secure and user friendly.

1. Your team must work on a mechanism that can keep track of all the opened tabs in one window, in a way that it can return all the tabs opened from beginning to the very recent tab but in a sequence order of recent tab till first tab.
2. The user can create a new tab by typing “NEW+”.
3. Warn the user when tabs limit exceeds more than 4.
4. The user can have the option to view the last tab active by typing “BACK” every time it takes user to the previous tab opened and on last tab the option of “BACK” is automatically disabled.

Solution:

Main:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

webbrowse browser = new webbrowse();

while (true)

{

Console.Write("Enter a command (NEW+, BACK, VIEW, SHOW, or QUIT): ");

string input = Console.ReadLine();

if (input == "QUIT")

{

break;

}

else if (input == "NEW+")

{

Console.Write("Enter URL: ");

string url = Console.ReadLine();

browser.OpenTab(url);

}

else if (input == "BACK")

{

browser.NavigateBack();

}

else if (input == "VIEW")

{

browser.ViewCurrentTab();

}

else if (input == "SHOW")

{

browser.ShowAllTabsInReverseOrder();

}

else

{

Console.WriteLine("Invalid command. Please try again.");

}

}

}

}

}

Browser (Class):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

internal class webbrowse

{

private List<string> tabs = new List<string>();

private int currentTab = -1;

private const int maxTabs = 4;

public void OpenTab(string url)

{

if (tabs.Count >= maxTabs)

{

Console.WriteLine("Tabs limit exceeded. Please close some tabs.");

return;

}

tabs.Add(url);

currentTab = tabs.Count - 1;

Console.WriteLine($"Opened tab {tabs.Count}: {url}");

}

public void NavigateBack()

{

if (currentTab > 0)

{

currentTab--;

Console.WriteLine($"Navigated to tab {currentTab + 1}: {tabs[currentTab]}");

}

else

{

Console.WriteLine("You are already on the first tab. BACK is disabled.");

}

}

public void ViewCurrentTab()

{

if (currentTab >= 0)

{

Console.WriteLine($"Current tab: {tabs[currentTab]}");

}

else

{

Console.WriteLine("No tab is open.");

}

}

public void ShowAllTabsInReverseOrder()

{

if (tabs.Count == 0)

{

Console.WriteLine("No tabs are open.");

return;

}

for (int i = tabs.Count - 1; i >= 0; i--)

{

Console.WriteLine($"Tab {i + 1}: {tabs[i]}");

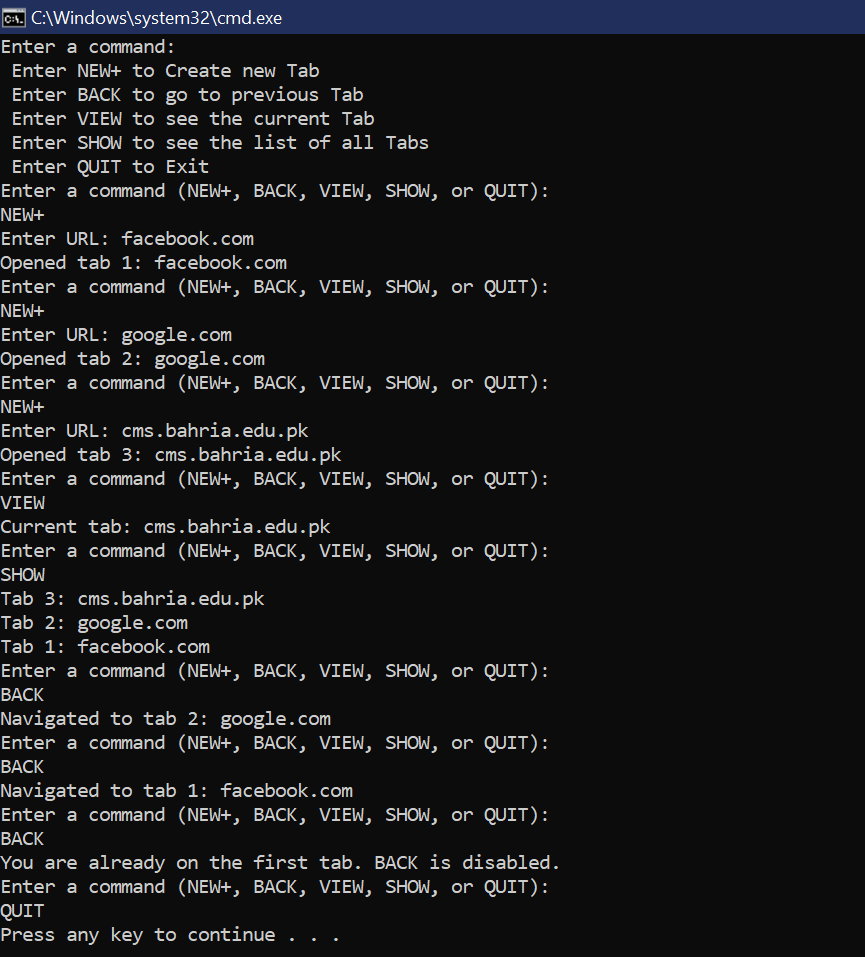
}

}

}

}

Output:



Question 02:

“Khalqat Hospital” has hired a team to build the system for patient record keeping. Every patients record is labeled with its Serial number (four digits) ,department initials and patient’s name separated by “-” **e.g. 1356-NUERO-ALI** for department “NUERO” and patient name is “ALI” in department wise shelves.

You must perform following operations:

1. Allow Admin to add new patients’ data to hospital’s each record shelf according to department (Neuron, Heart, Bones). If the Admin wants to **add new patient**, it should be added in the related department After adding new record to shelf, show the updated record with their **count.**
2. Admin can **remove any** record from any shelf. Show the updated shelf after deletion.
3. **Sort** the records in each shelf with reference to serial number in descending order.
4. Allow Admin to **search** any patient on any shelf.

Solution:

Main:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

HospitalRecordSystem recordSystem = new HospitalRecordSystem();

Console.WriteLine("Choose an operation:");

Console.WriteLine("1. Add Patient Record");

Console.WriteLine("2. Remove Patient Record");

Console.WriteLine("3. Sort Records by Department");

Console.WriteLine("4. Search Patient Record");

Console.WriteLine("5. Exit");

while (true)

{

Console.WriteLine("1. Add, 2. Remove, 3. Sort, 4. Search, 5. Exit");

Console.Write("Enter your choice: ");

int choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Console.Write("Enter serial number: ");

string serialNumber = Console.ReadLine();

Console.Write("Enter department: ");

string department = Console.ReadLine();

Console.Write("Enter patient name: ");

string patientName = Console.ReadLine();

recordSystem.AddPatientRecord(serialNumber, department, patientName);

break;

case 2:

Console.Write("Enter serial number: ");

serialNumber = Console.ReadLine();

Console.Write("Enter department: ");

department = Console.ReadLine();

recordSystem.RemovePatientRecord(serialNumber, department);

break;

case 3:

Console.Write("Enter department: ");

department = Console.ReadLine();

recordSystem.SortRecordsByDepartment(department);

break;

case 4:

Console.Write("Enter serial number: ");

serialNumber = Console.ReadLine();

Console.Write("Enter department: ");

department = Console.ReadLine();

recordSystem.SearchPatientRecord(serialNumber, department);

break;

case 5:

Environment.Exit(0);

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

}

}

}

}

PatientRecord (Class):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class PatientRecord

{

public string SerialNumber { get; set; }

public string Department { get; set; }

public string PatientName { get; set; }

public override string ToString()

{

return $"{SerialNumber}-{Department}-{PatientName}";

}

}

}

HospitalRecordSystem (Class):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class HospitalRecordSystem

{

private Dictionary<string, List<PatientRecord>> recordShelves = new Dictionary<string, List<PatientRecord>>();

public void AddPatientRecord(string serialNumber, string department, string patientName)

{

if (!recordShelves.ContainsKey(department))

{

recordShelves[department] = new List<PatientRecord>();

}

PatientRecord newRecord = new PatientRecord

{

SerialNumber = serialNumber,

Department = department,

PatientName = patientName

};

recordShelves[department].Add(newRecord);

Console.WriteLine($"Patient record added: {newRecord}");

ShowRecordsByDepartment(department);

}

public void RemovePatientRecord(string serialNumber, string department)

{

if (recordShelves.ContainsKey(department))

{

var records = recordShelves[department];

var recordToRemove = records.Find(r => r.SerialNumber == serialNumber);

if (recordToRemove != null)

{

records.Remove(recordToRemove);

Console.WriteLine($"Patient record removed: {recordToRemove}");

ShowRecordsByDepartment(department);

}

else

{

Console.WriteLine("Record not found in the specified department.");

}

}

else

{

Console.WriteLine("Department not found.");

}

}

public void SortRecordsByDepartment(string department)

{

if (recordShelves.ContainsKey(department))

{

recordShelves[department] = recordShelves[department].OrderByDescending(r => r.SerialNumber).ToList();

Console.WriteLine($"Records in department {department} sorted by serial number in descending order:");

ShowRecordsByDepartment(department);

}

else

{

Console.WriteLine("Department not found.");

}

}

public void SearchPatientRecord(string serialNumber, string department)

{

if (recordShelves.ContainsKey(department))

{

var record = recordShelves[department].Find(r => r.SerialNumber == serialNumber);

if (record != null)

{

Console.WriteLine($"Patient record found: {record}");

}

else

{

Console.WriteLine("Patient record not found.");

}

}

else

{

Console.WriteLine("Department not found.");

}

}

public void ShowRecordsByDepartment(string department)

{

if (recordShelves.ContainsKey(department))

{

Console.WriteLine($"Records in department {department}:");

foreach (var record in recordShelves[department])

{

Console.WriteLine(record);

}

Console.WriteLine($"Total records in department {department}: {recordShelves[department].Count}");

}

else

{

Console.WriteLine("Department not found.");

}

}

}

}

Output:

