Assignment 4

1. Create a custom exception class named StackException. The Push()and Pop() method should throw object of StackException when the stack is full or empty respectively.

Program

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StackExceptionPro

{

interface Istack

{

Boolean IsEmpty();

void Push(object element);

object Pop();

object Peek();

void Display();

}

class PushPop : Istack

{

public int top;

object[] item;

public int stack\_size;

public PushPop(int stack\_size) //parametrised constructor

{

this.stack\_size = stack\_size;

item = new object[stack\_size];

top = -1;

}

public PushPop() //default constructor

{

stack\_size = 10;

item = new object[stack\_size];

top = -1;

}

public bool IsEmpty() //Method to check is stack empty or full

{

if (top == (stack\_size - 1))

{

return true;

}

else

return false;

}

public void Push(object element)

{

if (top == (stack\_size - 1))

{

throw new Exception("Exception Stack overflow because stack is full");

}

else

{

item[++top] = element;

Console.WriteLine("Item Pushed Successfully");

}

}

public object Pop()

{

if (IsEmpty())

{

throw new Exception("Exception underflow because stack is empty");

return null;

}

else

{

return item[top--];

}

}

public object Peek()

{

if (IsEmpty())

{

throw new Exception("Stack Underflow exception because stack is empty");

return null;

}

else

{

return item[top];

}

}

public void Display()

{

for(int i = top; i>-1;i--)

{

Console.WriteLine(item[i]);

}

}

}

internal class Program

{

static void Main(string[] args)

{

PushPop st = new PushPop(5);

object element;

while(true)

{

Console.Clear();

Console.WriteLine("1. IsEmpty()");

Console.WriteLine("2. Push()");

Console.WriteLine("3. Pop()");

Console.WriteLine("4. Peek()");

Console.WriteLine("5. Display()");

Console.WriteLine("6. Exit()");

Console.WriteLine("Enter Your Choice");

int choice;

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

Console.WriteLine();

switch(choice)

{

case 1:

if(st.IsEmpty())

{

Console.WriteLine("Stack is Empty");

}

else

{

Console.WriteLine("Stack is not empty");

}

break;

case 2:

Console.WriteLine("Enter element");

element = Console.ReadLine();

try

{

st.Push(element);

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

break;

case 3:

try

{

element = st.Pop();

Console.WriteLine("Element removed is {0} ", element);

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

break;

case 4:

try

{

element = st.Peek();

Console.WriteLine("Element removed is {0} ", element);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

break;

case 5:

st.Display();

break;

case 6:

Environment.Exit(0);

break;

default:

Console.WriteLine("Enter valid choice");

break;

}

Console.ReadLine();

}

}

}

}

Output

