**Report On**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Advanced Object Oriented Programming Laboratory**

**(15UCSL605)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Submitted**

**By**

**NAME: RITESH KUMAR SINGH USN: 2SD15CS083**

**VI Semester, B Division Academic Year: 2017-18**

**Belonging to**

**Batch: B2**

**Faculty In-charge: (Faculty in-charge for lab)**

**Course Teacher: Indira R Umarji**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Department of Computer Science & Engineering**

**SDM College of Engineering and Technology**

**TERMWORK-1 Date: 12/02/2018**

**PROBLEM DEFINITION OF THE TERMWORK -1**

Using state diagram, design and describe the behaviour of

STACK which contains maximum of FOUR integers

Elements .

Implement the above design in JAVA Programming

Language .Design the TEST-DRIVER class to include

Minimum number of TEST CASES to test the complete

Features of STACK class designed.

**.**

**Expected Learning:**

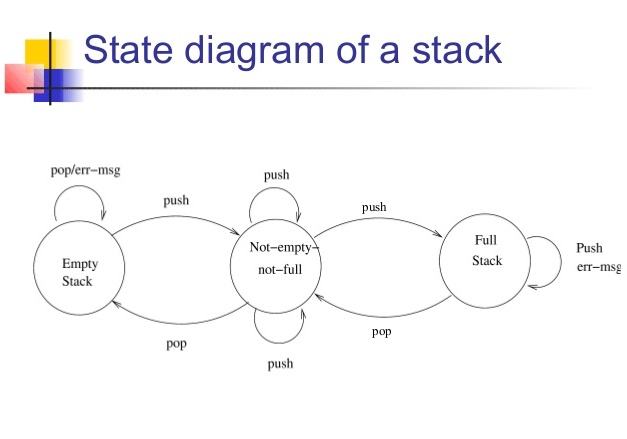
How to define the class, Use of Instance Variables, data types, operators, control structures, Understanding of access specifies, Declaring methods, parameterized methods, constructor, Interface, finalize() method, Compilation procedures, use of package, class

Path and other basic features .

**INTRODUCTION:**

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. push adds an item to the top of the stack, pop removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top.

|  |  |
| --- | --- |
| Push (new Entry) | Place a new element into the collection. The value provided becomes  the new topmost item in the collection. Usually there is no output  associated with this operation. |
| Pop () | Remove the topmost item from the stack. |
| Top () | Returns, but does not remove, the topmost item from the stack. |
| isEmpty () | Determines whether the stack is empty |

**State diagram**

**Program :**

vi mystack.java

package stack;

import java.util.\*;

public class mystack <E>

{

public int top;

public E s[];

public mystack()

{

top=-1;

this.s =(E[]) new Object[4];

}

public int push (E item)

{

if(top>=3)

{

return -1;

}

else

{

this.s[++top]=item;

return 1;

}

}

public E pop ()

{

if(top<0)

{

return null;

}

return s[top--];

}

}

**vi TestDriver.java**

import stack.\*;

import java.util.Scanner;

import java.util.\*;

public class TestDriver

{

public static void main (String[]args)

{

Scanner sc = new Scanner (System.in);

for (int j = 1; j <= 9; j++)

{

List < Long > l = new ArrayList < Long > ();

mystack < Integer > s1 = new mystack < Integer > ();

mystack < Long > s2 = new mystack < Long > ();

mystack < Double > a2 = new mystack < Double > ();

mystack < Character > a3 = new mystack < Character > ();

switch (j)

{

case 1:

if (s1.pop () == null)

{

System.out.println ("TEST CASE 1 : UNDERFLOW CONDITION" + "\t" + ": PASSED");

}

else

{

System.out.println ("TEST CASE 1 : UNDERFLOW CONDITION" + "\t" +" : FAILED");

}

break;

case 2:

s1.push (10);

s1.push (20);

int a = s1.push (30);

if (a == 1)

{

System.out.println ("TEST CASE 2 : PUSH 10,20,30 TO STACK" + "\t" + " : PASSED");

}

else

{

System.out.println ("TEST CASE 2 : PUSH 10,20,30 TO STACK" + "\t" + " : FAILED");

}

break;

case 3:

s1.push (10);

s1.push (20);

s1.push (30);

s1.push (40);

int b = s1.push (50);

if (b == -1)

{

System.out.println ("TEST CASE 3 : OVERFLOW CONDITION" + "\t" + " : PASSED");

}

else

{

System.out.println ("TEST CASE 3 : OVERFLOW CONDITION" +"\t" + " : FAILED");

}

break;

case 4:

s1.push (10);

s1.push (20);

s1.push (30);

s1.push (40);

long e = s1.pop ();

if (e != 40)

{

System.out.println ("TEST CASE 4 : POPING 40" + "\t" + ": FAILED");

}

long y = s1.pop ();

if (y != 30)

{

System.out.println ("TEST CASE 4 : POPING 30" + "\t" + FAILED");

}

long z = s1.pop ();

if (z != 20)

{

System.out.println ("TEST CASE 4 : POPING 20" + "\t" + ": FAILED");

}

long xyz = s1.pop ();

if (xyz != 10)

{

System.out.println ("TEST CASE 4 : POPING 10" + "\t" +”: FAILED");

}

else

{

System.out.println ("TEST CASE 4 : POPING 40,30,20,10" + "\t" + ":PASSED");

}

break;

case 5:

s1.push (10);

s1.push (20);

s1.pop (); //verify that pop cleans up after the removed object

s1.push (30);

int c = s1.push (40);

if (c == 1)

{

System.out.

println

("TEST CASE 5 : VERIFYING THAT POP CLEANS UP AFTER THE REMOVED OBJECT" + "\t" + " : PASSED");

}

else

{

System.out.

println

("TEST CASE 5 : VERIFYING THAT POP CLEANS UP AFTER THE REMOVED OBJECT" + "\t" + " : FAILED");

}

break;

case 6:

s2.push (10000000000L);

s2.push (20000000000L);

s2.push (30000000000L);

int d = s2.push (40000000000L);

if (d == 1)

{

System.out.

println ("TEST CASE 6 : PUSHING LONG INTEGER TO STACK" +"\t" + ": PASSED");

}

else

{

System.out.

println ("TEST CASE 6 : PUSHING LONG INTEGER TO STACK" + "\t" +” : FAILED");

}

break;

case 7:

a2.push (20.0);

a2.push (10.0);

a2.push (30.0);

int y1 = a2.push (40.0);

if (y1 == 1)

{

System.out.

println

("TEST CASE 7 : PUSHING FLOATING POINT INTEGER TO STACK" + "\t" + ": PASSED");

}

else

{

System.out.

println

("TEST CASE 7 : PUSHING FLOATING POINT INTEGER TO STACK" + "\t" + " : FAILED");

}

break;

case 8:

a3.push ('c');

a3.push ('x');

a3.push ('d');

int z1 = a3.push ('e');

if (z1 == 1)

{

System.out.

println ("TEST CASE 8 : PUSHING CHARACTER TO STACK" + "\t" +” : PASSED");

}

else

{

System.out.

println ("TEST CASE 8 : PUSHING CHARACTER TO STACK" + "\t" +” : FAILED");

}

break;

case 9:

System.exit (0);

break;

}

}

}

}

**INPUT/OUTPUT (SCREENSHOT) :**

