**Report On**

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

**Advanced Object Oriented Programming Laboratory**

**(15UCSL605)**

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

**Submitted**

**By**

**Name: SHRIKRISHNA BHAT USN: 2SD15CS099**

**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**

**Date: 12/02/2018**

**Complete problem definition of the Term work-1**

**Using** state diagram, **design** and **describe** the behavior of STACK which contains maximum of FOUR integer 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.

**Source code/Program**

**Contents of package stackpack**

package stackpack;

class StackFullException extends Exception{

String detail;

StackFullException(String a){

detail=a;

}

public String toString(){

return "StackFullException[ "+detail+" ]";

}

}

class StackUnderflowException extends Exception{

String detail;

StackUnderflowException(String a){

detail=a;

}

public String toString(){

return "StackUnderflowException[ "+detail+" ]";

}

}

class arrayStack {

protected int stck[];

protected int tos,size,len;

public arrayStack(int n){

size=n;

len=0;

stck=new int[size];

tos=-1;

}

public boolean isEmpty(){

return tos==-1;

}

public boolean isFull(){

return tos==size-1;

}

public int getSize(){

return len;

}

/\*public int peek(){

if(isEmpty())

throw new StackUnderflowException("Underflow Exception");

return stck[top];

}

\*/

public void push(int i) throws StackFullException{

if(tos+1 >= size)

throw new StackFullException("Overflow");

if(tos+1<size)

stck[++tos]=i;

len++;

}

public int pop() throws StackUnderflowException{

if(isEmpty())

throw new StackUnderflowException("Underflow");

len--;

return stck[tos--];

}

public void display(){

System.out.println("Stack is ");

if(len==0){

System.out.println("Empty");

return;

}

for(int i=tos;i>=0;i--)

System.out.println(stck[i]+" ");

System.out.println();

}

}

**Contents of main program**

import stackpack.\*;

import java.util.\*;

public class TestStack2{

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

System.out.println("Enter size");

int size=sc.nextInt();

arrayStack stk=new arrayStack(size);

for(;;){

System.out.println("\nStack OP");

System.out.println("1.push, 2.pop, 3.display 4.exit");

int choice=sc.nextInt();

switch(choice){

case 1:

System.out.println("Enter element to be pushed");

try{

stk.push(sc.nextInt());

}catch(StackFullException e){

System.out.println("Error: "+e);

}

break;

case 2:

try{

System.out.println("Popped element is "+stk.pop());

}catch(StackUnderflowException e){

System.out.println("Error: "+e);

}

break;

case 3:

stk.display();

break;

default:

System.exit(0);

}

}

}

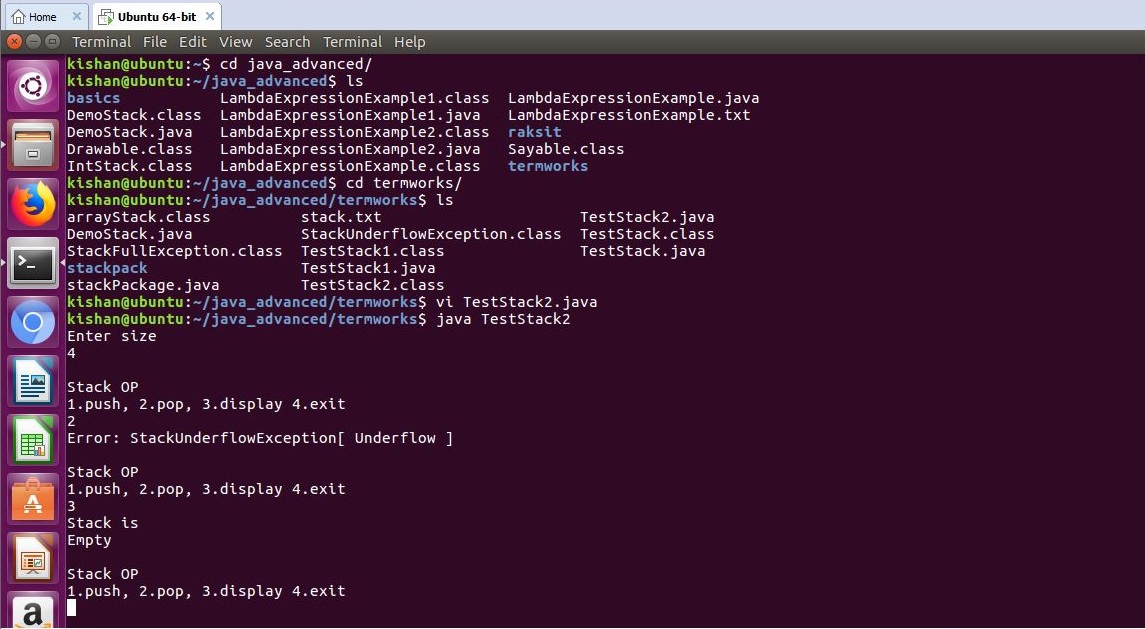
}

**Case 1: Underflow and no elements in stack**

**Input:**

**Size=4**

**Output with screenshot.**

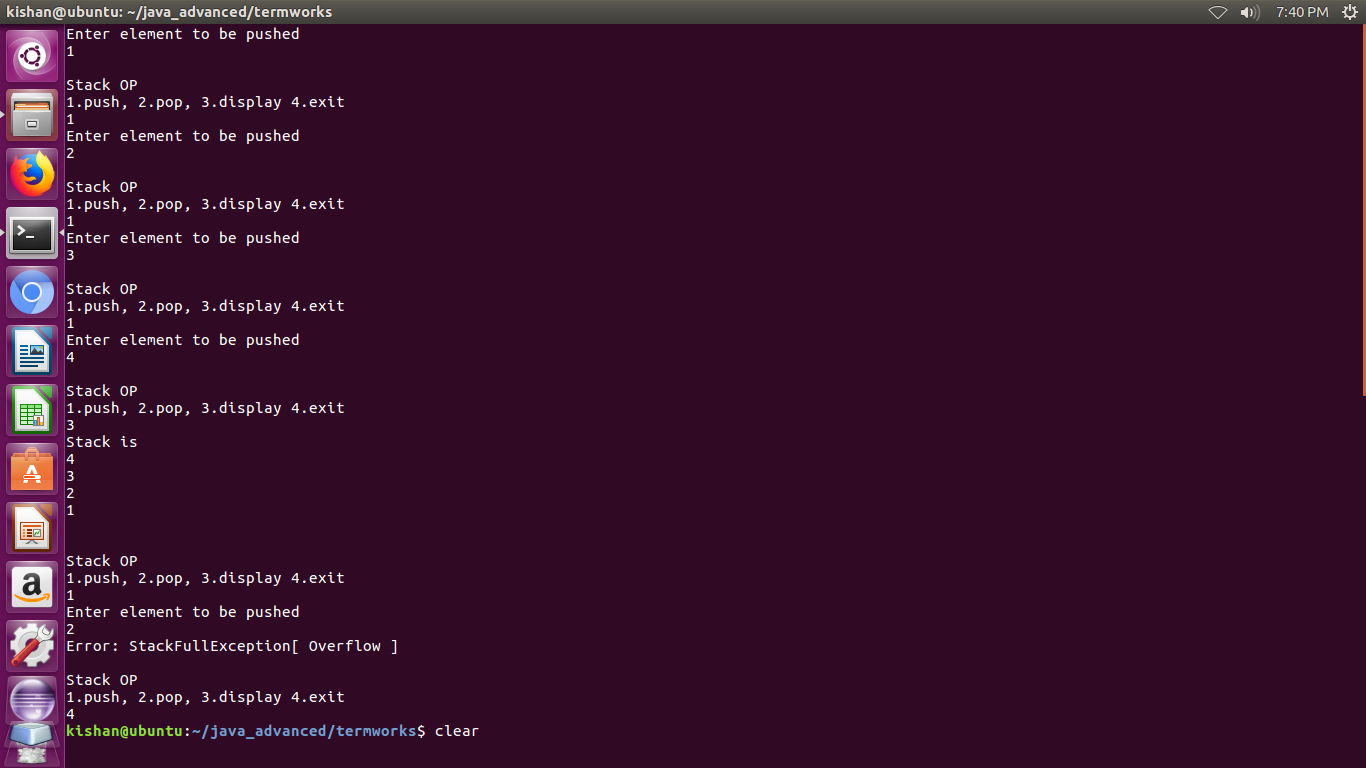
****

**Case 2: Overflow and elements in stack**

**Input:**

**Size=4**

**Output with screenshot.**



**Case 3: Push elements pop and display**

**Input:**

**Size=4**

**Output with screenshot.**

