Exp\_no:4

Aim:

To write a java application to implement Stack ADT with neccessary exception handling.

Algorithm:

1. Create a interface to implement stack ADT with neccessary abstract methods.
2. Create a class Stack that implements stack interface.
3. In Stack class, declare an array for stack, top pointer and maxSize variables.
4. Define a constructor that takes size as parameter and initialize the stack array with size.
5. The following methods are implemented on class Stack.
6. Push – It takes value parameter. If the stack is not full, append the value to the array. Otherwise issue StackOverflow Error.
7. Pop – If the stack is not empty, store the last value to temp variable. Decrement the top variable by 1. Return the value of temp.
8. IsFull- If the stack is full, return true. Otherwise return fase.
9. IsEmpty – If the stack is empty, return true. Otherwise return fase.
10. Peak – Return the last value from the stack array.
11. PrintStack – Print the values stored in stack in reverse order.
12. Create a Main class with main method.
13. The user is asked to enter the size of the stack.
14. The user is provided with options to choose from.
15. The selected operation is performed on the Stack.
16. Repeat step 14 and 15 until the user opts to exit the program.

Source Code:

// Interface Stack

package stack;

public interface IStack {

void push(int a);

int pop();

boolean isFull();

boolean isEmpty();

int peak();

void printStack();

}

//Class Stack

package stack;

public class Stack implements IStack{

private int[] array;

private int top;

private int maxSize;

public Stack(int size){

maxSize = size;

array = new int[size];

top = -1;

}

public void push(int a){

try{

top += 1;

array[top] = a;

}catch (Exception e){

System.out.println("Stack Overflow!");

top--;

}

}

public int pop(){

int temp = -95456154;

try{

temp = array[top];

}catch (Exception e){

top++;

}

top--;

return temp;

}

public boolean isFull(){

return (top >= (maxSize-1));

}

public boolean isEmpty(){

return (top == -1);

}

public int peak(){

return array[top];

}

public void printStack(){

System.out.println();

if(top != -1) {

for (int i = top; i >= 0; i--) {

System.out.println(array[i]);

}

}else{

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

}

}

}

//Main Class

package stack;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Stack myStack;

Scanner sc = new Scanner(System.in);

int num;

System.out.println("Enter the number of slots:");

myStack = new Stack(sc.nextInt());

while(true){

System.out.println("\n1.Push to Stack\n2.Pop from Stack\n3.Peak from Stack\n4.Print Stack\n"

+ "5.Is the Stack full\n6.Is the Stack empty\n7.Exit");

System.out.println("Enter your choice:");

int choice = sc.nextInt();

switch(choice){

case 1:

System.out.println("Enter the number:");

num = sc.nextInt();

myStack.push(num);

break;

case 2:

num = myStack.pop();

if(num == -95456154) {

System.out.println("Stack UnderFlow!");

}else{

System.out.println("Popped " + num + "from the Stack");

}

break;

case 3:

num = myStack.peak();

System.out.println("Peak "+ num + "from the Stack");

break;

case 4:

myStack.printStack();

break;

case 5:

if(myStack.isFull()){

System.out.println("The Stack full");

}else{

System.out.println("The Stack is not full");

}

break;

case 6:

if(myStack.isEmpty()){

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

}else{

System.out.println("The Stack is not empty");

}

break;

case 7:

return;

}

}

}

}

Expected Output:

Enter the number of slots:

3

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

1

Enter the number:

1

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

1

Enter the number:

2

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

1

Enter the number:

3

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

4

3

2

1

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

3

Peak 3from the Stack

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

2

Popped 3 from the Stack

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

4

2

1

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

5

The Stack is not full

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

6

The Stack is not empty

1.Push to Stack

2.Pop from Stack

3.Peak from Stack

4.Print Stack

5.Is the Stack full

6.Is the Stack empty

7.Exit

Enter your choice:

7

Result:

Thus, java application for implementation of Stack ADT with neccessary exception handling has been written, executed and output is verified.