Name-Anurag Singh

Class-AIML A1

PRN-21070126016

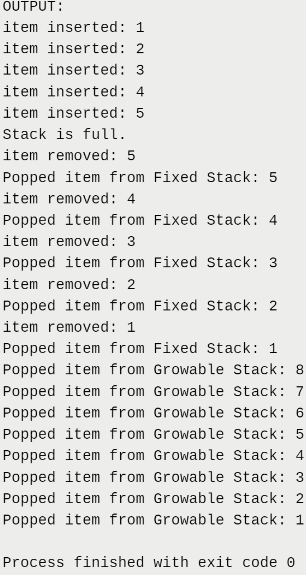
Assignment 6

Code-

/\*  
  
  
PROBLEM STATEMENT:  
Part 1: An implementation of IntStack (integer stack) that uses fixed storage as well as "growable"  
using interface.  
Create a user defined package “pkg\_Stack” where the interface is stored. The other two complete  
classes will need to import the package ‘pkg\_Stack’ and then use it.  
Part 2: Program to implement the following Multiple Inheritance.  
 \*/  
  
  
  
package com.College;  
import java.util.ArrayList;  
  
public class Assignment\_6 {  
 public static void main(String[] arg){  
  
 Fixed\_stk fixedStack = new Fixed\_stk(5);  
 growable\_stk growableStack = new growable\_stk();  
  
 // Push items to the fixed stack  
 fixedStack.push(1);  
 fixedStack.push(2);  
 fixedStack.push(3);  
 fixedStack.push(4);  
 fixedStack.push(5);  
  
 // Try to push an additional item to the fixed stack (which is full)  
 fixedStack.push(6); // Output: Stack is full.  
  
 // Pop items from the fixed stack  
 while (!fixedStack.isEmpty()) {  
 System.*out*.println("Popped item from Fixed Stack: " + fixedStack.pop());  
 }  
  
 // Push items to the growable stack  
 growableStack.push(1);  
 growableStack.push(2);  
 growableStack.push(3);  
 growableStack.push(4);  
 growableStack.push(5);  
  
 // Push more items to the growable stack (which will trigger its growth)  
 growableStack.push(6);  
 growableStack.push(7);  
 growableStack.push(8);  
  
 // Pop items from the growable stack  
 while (!growableStack.isEmpty()) {  
 System.*out*.println("Popped item from Growable Stack: " + growableStack.pop());  
 }  
 }  
}  
  
  
  
class Fixed\_stk implements Interface\_STK {  
 private int[] stack;  
 private int top;  
  
 public Fixed\_stk(int size) {  
 stack = new int[size];  
 top = -1;  
 }  
  
 public void push(int item) {  
 if (isFull()) {  
 System.*out*.println("Stack is full.");  
 } else {  
 stack[++top] = item;  
 System.*out*.println("item inserted: " + item);  
 }  
 }  
  
 public int pop() {  
 if (isEmpty()) {  
 System.*out*.println("Stack is empty.");  
 return -1;  
 } else {  
 int popped = stack[top--];  
 System.*out*.println("item removed: " + popped);  
 return popped;  
 }  
 }  
  
 public int peek() {  
 if (isEmpty()) {  
 System.*out*.println("Stack is empty.");  
 return -1;  
 } else {  
 return stack[top];  
 }  
 }  
  
 public boolean isEmpty() {  
 return top == -1;  
 }  
  
 public boolean isFull() {  
 return top == stack.length - 1;  
 }  
  
 public void size(){  
 System.*out*.println(stack.length);  
 }  
}  
  
  
  
class growable\_stk implements Interface\_STK {  
 private ArrayList<Integer> stack;  
 private int top;  
  
 public growable\_stk() {  
 stack = new ArrayList<Integer>();  
 top = -1;  
 }  
  
 public void push(int item) {  
 stack.add(++top, item);  
 }  
  
 public int pop() {  
 if (isEmpty()) {  
 System.*out*.println("Stack is empty.");  
 return -1;  
 } else {  
 return stack.remove(top--);  
 }  
 }  
  
 public int peek() {  
 if (isEmpty()) {  
 System.*out*.println("Stack is empty.");  
 return -1;  
 } else {  
 return stack.get(top);  
 }  
 }  
  
 public boolean isEmpty() {  
 return top == -1;  
 }  
  
 public boolean isFull() {  
 System.*out*.println("Not valid for growable stack.");  
 return false;  
 }  
  
 public void size(){  
 System.*out*.println(stack.size());  
 }  
}

Interface stk-

package com.College;  
  
public interface Interface\_STK {  
 int *max* = 10;  
 int *top* = 0;  
 void push(int item); // add item to the stack  
 int pop(); // remove and return the top item from the stack  
 int peek(); // return the top item from the stack without removing it  
 boolean isEmpty(); // check if the stack is empty  
 boolean isFull(); // check if the stack is full  
 void size();  
}

output-

github link-<https://github.com/Boomsnipa/Java_assignments>