Assignment 3

Comp 2230\_02

Colton isles and kaylee crocker

2024



**COMP 2230 – Data Structures and Algorithm Analysis**

Assignment #3: Linked Stacks

## Due Date: Section 01 - Sept. 26th 2024 Section 02 Sept 27th 2024

**Chapter 13**

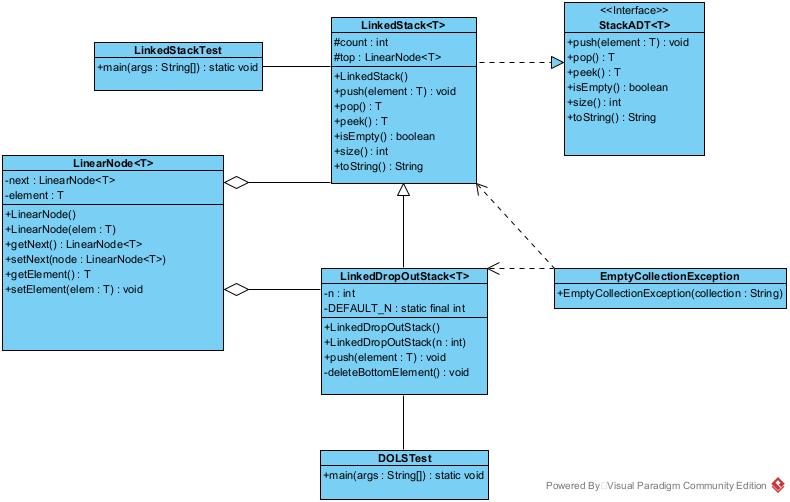
**Problem 1**:

Implementation of the Size(), isEmpty(), and toString() methods into the LinkedStack class.

**Problem 2**:

Override the push method to allow for a drop out mechanism with a linked stack

**UML:**



**Problem 1 Code**

|  |
| --- |
| **LinkedStack.java**  package Ass3\_2230;  import Ass3\_2230.exceptions.\*;  /\*\*  \* Represents a linked implementation of a stack.  \*  \* @author Java Foundations  \* @version 4.0  \*/  public class LinkedStack<T> implements StackADT<T>  {  protected int count;  protected LinearNode<T> top;  /\*\*  \* Creates an empty stack.  \*/  public LinkedStack()  {  count = 0;  top = null;  }  /\*\*  \* Adds the specified element to the top of this stack.  \* @param element element to be pushed on stack  \*/  public void push(T element)  {  LinearNode<T> temp = new LinearNode<T>(element);  temp.setNext(top);  top = temp;  count++;  }  /\*\*  \* Removes the element at the top of this stack and returns a  \* reference to it.  \* @return element from top of stack  \* @throws EmptyCollectionException if the stack is empty  \*/  public T pop() throws EmptyCollectionException  {  if (isEmpty())  throw new EmptyCollectionException("stack");  T result = top.getElement();  top = top.getNext();  count--;  return result;  }  /\*\*  \* Returns a reference to the element at the top of this stack.  \* The element is not removed from the stack.  \* @return element on top of stack  \* @throws EmptyCollectionException if the stack is empty  \*/  public T peek() throws EmptyCollectionException  {  if(isEmpty()){  throw new EmptyCollectionException("Linked Stack");  }    return top.getElement(); // temp  }  /\*\*  \* Returns true if this stack is empty and false otherwise.  \* @return true if stack is empty  \*/  public boolean isEmpty()  {  return count == 0;  }  /\*\*  \* Returns the number of elements in this stack.  \* @return number of elements in the stack  \*/  public int size()  {  return count;  }  /\*\*  \* Returns a string representation of this stack.  \* @return string representation of the stack  \*/  @Override  public String toString()  {  String result = "";  LinearNode<T> current = top;  for (int i = 1; i <= count; i++) {  result += current.getElement() + ",";  current = current.getNext();  }    return result;  }  } |

|  |
| --- |
| **LinkedStackTest.java**  package Ass3\_2230;  import Ass2\_2230.ArrayStack;  import Ass2\_2230.exceptions.EmptyCollectionException;  public class LinkedStackTest {  public static void main(String[] args){  LinkedStack<Integer> linkStack = new LinkedStack<>();  //empty stack initialization  System.out.println("Fresh initialized stack: " + "Top -> " + linkStack.toString() + " <- Bottom");  for(int i = 1; i < 6; i++){  linkStack.push(i);  }  System.out.println("Filled Stack: " + "Top -> " + linkStack.toString() + " <- Bottom");  //test pop and peek  System.out.println("-----pop() & peek() Test-----");  System.out.println("Peek Top Value: " + linkStack.peek());  for (int i = 1; i < 5; i++) {  linkStack.pop();  }  System.out.println("Stack after pop test: " + "Top -> " + linkStack.toString() + " <- Bottom");  System.out.println("Top Value after pop test: " + linkStack.peek());  linkStack.pop();  //test peek with empty method  System.out.println("-----peek() with empty stack test-----");  try{  linkStack.peek();  } catch (EmptyCollectionException e) {  System.out.println("peek() throws empty collection exception correctly");  }  //test pop with empty stack  System.out.println("-----pop() with empty stack test-----");  try {  linkStack.pop();  } catch (EmptyCollectionException e) {  System.out.println("pop() throws empty collection exception correctly");  }  System.out.println("Is the stack empty: " + linkStack.isEmpty());  System.out.println("stack size: " + linkStack.size());  }  } |

|  |
| --- |
| **Test Output** |

**Problem 2 Code**

|  |
| --- |
| **LinkedDropOutStack.java**  package Ass3\_2230;  public class LinkedDropOutStack<T> extends LinkedStack<T> {    private int n;  private static final int DEFAULT\_N = 100;    LinkedDropOutStack() {  this(DEFAULT\_N);  }    LinkedDropOutStack(int n) {  super();  this.n = n;  }    public void push(T element) {  super.push(element);    if (size() > n) {  deleteBottomElement();  }  }    private void deleteBottomElement() {    LinearNode<T> newBottomNode = top;  for (int i = 1; i < size()-1; i++) {  newBottomNode = newBottomNode.getNext();  }  newBottomNode.setNext(null);  count--;  }  } |

|  |
| --- |
| **DOLSTest.java**  package Ass3\_2230;  import Ass2\_2230.ArrayStack;  import Ass2\_2230.DropOutArrayStack;  import Ass2\_2230.exceptions.EmptyCollectionException;  public class DOLSTest {  public static void main(String[] args){  LinkedDropOutStack<Integer> dols = new LinkedDropOutStack<>(5);  //initialization with null values and capacity 5  System.out.println("Current stack: " + "Top -> " + dols.toString() + " <- Bottom");  //populate stack to fill the initial capacity  for (int i = 1; i < 6; i++) {  dols.push(i);  }  System.out.println("Full stack: " + "Top -> " + dols.toString() + " <- Bottom");  dols.push(6);  dols.push(7);  System.out.println("Full stack after dropout occurs : " + "Top -> " + dols.toString() + " <- Bottom");    //test pop and peek  System.out.println("-----pop() & peek() Test-----");  System.out.println("Peek Top Value: " + dols.peek());  for (int i = 1; i < 5; i++) {  dols.pop();  }  System.out.println("Stack after pop test: " + "Top -> " + dols.toString() + " <- Bottom");  System.out.println("Top Value after pop test: " + dols.peek());  dols.pop();  //test peek with empty method  System.out.println("-----peek() with empty stack test-----");  try{  dols.peek();  } catch (EmptyCollectionException e) {  System.out.println("peek() throws empty collection exception correctly");  }  //test pop with empty stack  System.out.println("-----pop() with empty stack test-----");  try {  dols.pop();  } catch (EmptyCollectionException e) {  System.out.println("pop() throws empty collection exception correctly");  }  System.out.println("Is the stack empty: " + dols.isEmpty());  System.out.println("stack size: " + dols.size());  }  } |

|  |
| --- |
| **Test Output** |