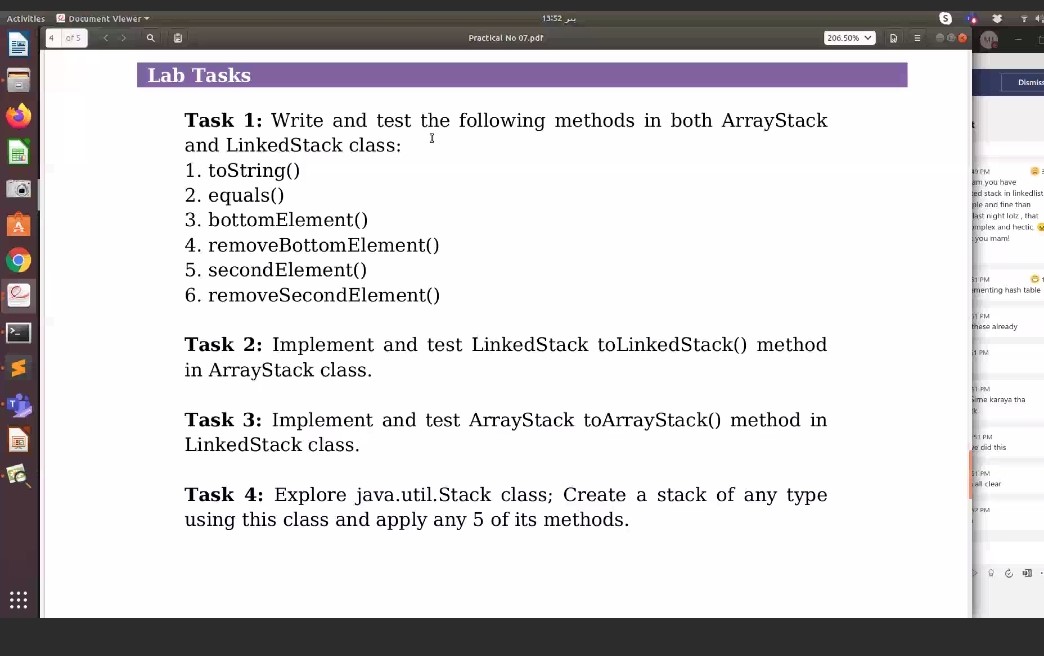
Name: **ZOHAIB HASSAN SOOMRO**

RollNo#: **19SW42**

Subject: **DSA**





Task#1, Task#2, Task3:

**ANS:**

1. **Stack:**
2. public interface Stack {
3. public Object peek();
4. public Object pop();
5. public boolean push(Object object);
6. public int size();
7. public boolean isEmpty();
8. public String toString();
9. }
10. **ArrayStack:**

import java.util.Arrays;

import java.util.NoSuchElementException;

public class ArrayStack implements Stack {

private int size;

private Object array[];

public ArrayStack() {

}

public ArrayStack(int capacity) {

array = new Object[capacity];

}

*@Override*

public Object peek() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

return array[size - 1];

}

*@Override*

public Object pop() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

Object obj = array[--size];

array[size] = null;

return obj;

}

*@Override*

public boolean push(Object object) {

if (size == array.length)

array = Arrays.*copyOf*(array, array.length \* 2);

array[size++] = object;

return true;

}

*@Override*

public int size() {

return size;

}

*@Override*

public boolean isEmpty() {

return size == 0;

}

///////////////////Method to Convert the ArrayStack to an string

*@Override*

public String toString() {

if (this.isEmpty())

return "[]";

String buffer = "[";

for (int i = size - 1; i >= 0; i--)

buffer += array[i] + ",";

buffer = buffer.substring(0, buffer.length() - 1);

return (buffer + "]");

}

///////////////////Method to check if two ArrayStacks are equal

public boolean equals(ArrayStack stack) {

if (this.isEmpty() && stack.isEmpty())

return true;

if (this.isEmpty() || stack.isEmpty())

return false;

if (this.size() != stack.size())

return false;

for (int i = 0; i < size(); i++) {

if (!array[i].equals(stack.array[i]))

return false;

}

return true;

}

///////////////////Method to return bottom element

public Object bottomElement() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

return array[0];

}

///////////////////Method to remove+return bottom element

public Object removeBottomElement() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

Object object = array[0];

System.*arraycopy*(array, 1, array, 0, --size);

return object;

}

///////////////////Method to return second element

public Object secondElement() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

return array[size - 2];

}

///////////////////Method to remove+return second element

public Object removeSecondElement() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

Object object = array[size - 2];

array[size - 2] = array[size - 1];

array[--size] = null;

return object;

}

///////////////////Method to convert current ArrayStack to equivalent LinkedStack

public LinkedStack toLinkedStack() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

LinkedStack stack = new LinkedStack();

for (int i = 0; i < size(); i++) {

stack.push(array[i]);

}

return stack;

}

public static void main(String[] args) {

ArrayStack stack = new ArrayStack(4);

stack.push(2);

stack.push(50);

stack.push("Hello");

stack.push(20);

ArrayStack stack2 = new ArrayStack(4);

stack2.push(2);

stack2.push(50);

stack2.push("Hello");

stack2.push(20);

System.***out***.println("stack.toString() : " + stack.toString());

System.***out***.println("stack2.toString() : " + stack2.toString());

System.***out***.println("\nstack.equals(stack2) : " + stack.equals(stack2));

System.***out***.println("\nstack.removeBottomElement() : " + stack.removeBottomElement());

System.***out***.println("stack.bottomElement() : " + stack.bottomElement());

System.***out***.println("\nstack.removeSecondElement() : " + stack.removeSecondElement());

System.***out***.println("stack.secondElement() : " + stack.secondElement());

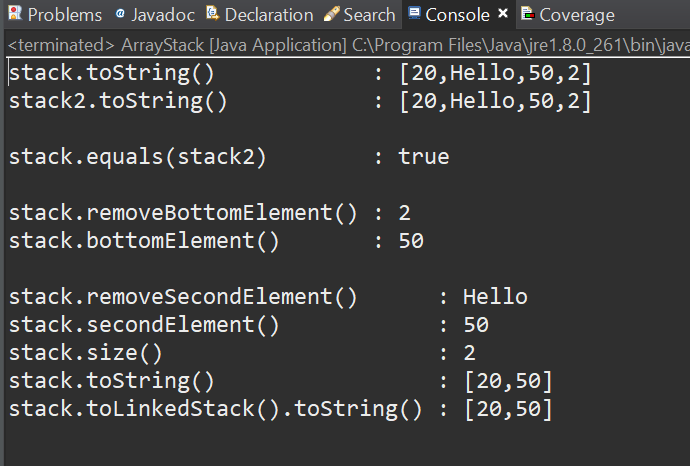
System.***out***.println("stack.size() : " + stack.size());

System.***out***.println("stack.toString() : " + stack.toString());

System.***out***.println("stack.toLinkedStack().toString() : " + stack.toLinkedStack().toString());

}

}



1. **LinkedStack:**

import java.util.NoSuchElementException;

public class LinkedStack implements Stack {

private int size;

private Node top;

private class Node {

private Object data;

private Node next;

public Node(Object data, Node next) {

this.data = data;

this.next = next;

}

}

*@Override*

public Object peek() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

return top.data;

}

*@Override*

public Object pop() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

Object object = top.data;

top = top.next;

size--;

return object;

}

*@Override*

public boolean push(Object object) {

top = new Node(object, top);

size++;

return true;

}

*@Override*

public int size() {

return size;

}

*@Override*

public boolean isEmpty() {

return size == 0;

}

/////////////////// Method to Convert the LinkedStack to an string

*@Override*

public String toString() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

Node start = top;

String buffer = "[";

while (start != null) {

buffer += start.data + ",";

start = start.next;

}

buffer = buffer.substring(0, buffer.length() - 1);

return buffer + "]";

}

///////////////////Method to check if two ArrayStacks are equal

public boolean equals(LinkedStack stack) {

if (this.isEmpty() && stack.isEmpty())

return true;

if (this.isEmpty() || stack.isEmpty())

return false;

if (this.size() != stack.size())

return false;

Node p1 = this.top;

Node p2 = stack.top;

while (p1 != null) {

if (!p1.data.equals(p2.data))

return false;

p1 = p1.next;

p2 = p2.next;

}

return true;

}

///////////////////Method to return bottom element

public Object bottomElement() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

Node p = top;

while (p.next != null) {

p = p.next;

}

return p.data;

}

///////////////////Method to remove+return bottom element

public Object removeBottomElement() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

Node p = top;

while (p.next.next != null) {

p = p.next;

}

Object obj = p.next.data;

p.next = null;

--size;

return obj;

}

///////////////////Method to return second element

public Object secondElement() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

return top.next.data;

}

///////////////////Method to remove+return second element

public Object removeSecondElement() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

Object object = top.next.data;

top.next = top.next.next;

--size;

return object;

}

///////////////////Method to convert current ArrayStack to equivalent LinkedStack

public ArrayStack toArrayStack() {

if (this.isEmpty())

throw new NoSuchElementException("Stack is empty!");

ArrayStack stack = new ArrayStack(this.size());

Object array[] = new Object[size()];

int index = 0;

for (Node i = top; i != null; i = i.next)

array[index++] = i.data;

while (--index >= 0)

stack.push(array[index]); // for preserving same order that's why storing elements in an Object array

return stack;

}

public static void main(String[] args) {

LinkedStack stack = new LinkedStack();

stack.push(2);

stack.push(50);

stack.push("Hello");

stack.push(20);

stack.push(60);

stack.push(45);

stack.push(80);

LinkedStack stack2 = new LinkedStack();

stack2.push(2);

stack2.push(50);

stack2.push("Hello");

stack2.push(20);

System.***out***.println("stack.toString() : " + stack.toString());

System.***out***.println("stack2.toString() : " + stack2.toString());

System.***out***.println("\nstack.equals(stack2) : " + stack.equals(stack2));

System.***out***.println("\nstack.removeBottomElement() : " + stack.removeBottomElement());

System.***out***.println("stack.bottomElement() : " + stack.bottomElement());

System.***out***.println("\nstack.removeSecondElement() : " + stack.removeSecondElement());

System.***out***.println("stack.secondElement() : " + stack.secondElement());

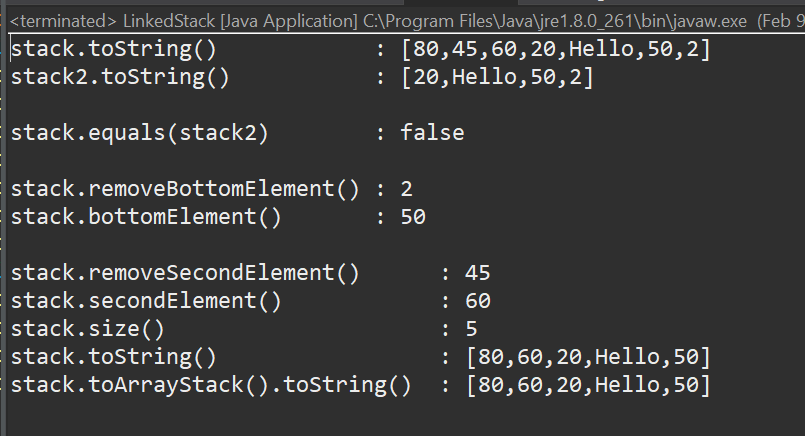
System.***out***.println("stack.size() : " + stack.size());

System.***out***.println("stack.toString() : " + stack.toString());

System.***out***.println("stack.toArrayStack().toString() : " + stack.toArrayStack().toString());

}

}



**Task#4**

**Code:**

import java.util.Stack;

public class StackTask2 {

public static void main(String[] args) {

Stack<String> nameStack = new Stack<String>();

nameStack.push("Zohaib"); // #1

nameStack.push("Syed Ahmad");

nameStack.push("Uzair");

nameStack.push("Noman");

System.***out***.println("nameStack.toString(): " + nameStack.toString()); // #2

System.***out***.println("nameStack.peek() : " + nameStack.peek()); // #3

System.***out***.println("nameStack.pop() : " + nameStack.pop()); // #4

System.***out***.println("nameStack.capacity(): " + nameStack.capacity()); // #5

System.***out***.println("nameStack.isEmpty() : " + nameStack.isEmpty()); // #6

}

}

**OUTPUT:**

