**Object Oriented Programming**

Q1. Program to create a generic stack and do the Push and Pop operations.

class Stack {

// store elements of stack

private int arr[];

// represent top of stack

private int top;

// total capacity of the stack

private int capacity;

// Creating a stack

Stack(int size) {

// initialize the array

// initialize the stack variables

arr = new int[size];

capacity = size;

top = -1;

}

// push elements to the top of stack

public void push(int x) {

if (isFull()) {

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

// terminates the program

System.exit(1);

}

// insert element on top of stack

System.out.println("Inserting " + x);

arr[++top] = x;

}

// pop elements from top of stack

public int pop()

{

// if stack is empty

// no element to pop

if (isEmpty()) {

System.out.println("STACK EMPTY");

// terminates the program

System.exit(1);

}

// pop element from top of stack

return arr[top--];

}

// return size of the stack

public int getSize() {

return top + 1;

}

// check if the stack is empty

public Boolean isEmpty() {

return top == -1;

}

// check if the stack is full

public Boolean isFull() {

return top == capacity - 1;

}

// display elements of stack

public void printStack() {

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

System.out.print(arr[i] + ", ");

}

}

public static void main(String[] args) {

Stack stack = new Stack(5);

stack.push(5);

stack.push(6);

stack.push(7);

System.out.print("Stack: ");

stack.printStack();

// remove element from stack

stack.pop();

System.out.println("\nAfter popping out");

stack.printStack();

}

}

**OUTPUT**

**![Text

Description automatically generated]()**

Q2. Using generic method perform Bubble sort.

public class BubbleSort {

static void bubbleSort(int[] arr) {

int n = arr.length;

int temp = 0;

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

for(int j=1; j < (n-i); j++) {

if(arr[j-1] > arr[j]) {

temp = arr[j-1];

arr[j-1] = arr[j];

arr[j] = temp;

}

}

}

}

public static void main(String[] args) {

int arr[] = { 8,5,-1,-3,4,7,-7,-8,0,11,-12 };

System.out.println("Array Before Bubble Sort");

for(int i = 0; i < arr.length; i++) {

System.out.print(arr[i] + " ");

}

System.out.println();

bubbleSort(arr);

System.out.println("Array After Bubble Sort");

for(int i = 0; i < arr.length; i++) {

System.out.print(arr[i] + " ");

}

}

}

**OUTPUT**

![Text

Description automatically generated]()

Q3. Maintain a list of Strings using ArrayList from collection framework, perform built-in operations.

import java.util.\*;

public class arraylist

{

public static void main(String args[])

{

ArrayList<String> list=new ArrayList<String>();

list.add("Mango");

list.add("Apple");

list.add("Banana");

list.add("Grapes");

list.add("Orange");

list.add("Pappaya");

list.add("Kiwi");

System.out.println(list);

} }

**OUTPUT**

**![Text

Description automatically generated]()**

Q4. Program to remove all the elements from a linked list

import java.util.\*;

public class removelinkedlist {

public static void main(String args[])

{

LinkedList<String> l\_list = new LinkedList<String>();

l\_list.add("APPLE");

l\_list.add("MANGO");

l\_list.add("GRAPES");

l\_list.add("PINEAPPLE");

l\_list.add("ORANGE");

System.out.println("The Original linked list: " + l\_list);

l\_list.clear();

System.out.println("The New linked list: " + l\_list);

}

}

**OUTPUT**

**![Graphical user interface, text

Description automatically generated]()**

Q5. Program to demonstrate the addition and deletion of elements in deque.

import java.util.\*;

public class DequeExample {

public static void main(String[] args)

{

Deque<String> deque

= new LinkedList<String>(); deque.add("Element 1 (Tail)");

deque.addFirst("Element 2 (Head)");

deque.addLast("Element 3 (Tail)");

deque.push("Element 4 (Head)");

deque.offer("Element 5 (Tail)");

deque.offerFirst("Element 6 (Head)");

System.out.println(deque + "\n");

deque.removeFirst();

deque.removeLast();

System.out.println("Deque after removing "+ "first and last: "+ deque);

}

}

**OUTPUT**

**![Text

Description automatically generated]()**

Q6. Program to demonstrate the working of Map interface by adding, changing and removing

import java.util.\*;

class HashMapDemo {

public static void main(String args[])

{

Map<String, Integer> hm

= new HashMap<String, Integer>();

hm.put("a", new Integer(100));

hm.put("b", new Integer(200));

hm.put("c", new Integer(300));

hm.put("d", new Integer(400));

// Traversing through the map

for (Map.Entry<String, Integer> me : hm.entrySet()) {

System.out.print(me.getKey() + ":");

System.out.println(me.getValue());

}

}

}

**OUTPUT**

**![Text

Description automatically generated]()**