1. **Write a class ComplexNumber having attributes real & imaginary. Add functions like add, subtract, multiply & swap**

public class ComplexNumber{

double real, img;

ComplexNumber(double r, double i){

this.real = r;

this.img = i;

}

public static ComplexNumber sum(ComplexNumber c1, ComplexNumber c2)

{

ComplexNumber temp = new ComplexNumber(0, 0);

temp.real = c1.real + c2.real;

temp.img = c1.img + c2.img;

return temp;

}

public static ComplexNumber sub(ComplexNumber c1, ComplexNumber c2)

{

ComplexNumber temp1 = new ComplexNumber(0, 0);

temp1.real = c1.real - c2.real;

temp1.img = c1.img - c2.img;

return temp1;

}

public static ComplexNumber mul(ComplexNumber c1, ComplexNumber c2)

{

ComplexNumber temp2 = new ComplexNumber(0, 0);

temp2.real = c1.real \* c2.real;

temp2.img = c1.img \* c2.img;

return temp2;

}

public static ComplexNumber div(ComplexNumber c1, ComplexNumber c2)

{

ComplexNumber temp3 = new ComplexNumber(0, 0);

temp3.real = c1.real / c2.real;

temp3.img = c1.img / c2.img;

return temp3;

}

public static void main(String args[]) {

ComplexNumber c1 = new ComplexNumber(5.5, 4);

ComplexNumber c2 = new ComplexNumber(1.2, 3.5);

ComplexNumber temp = sum(c1, c2);

ComplexNumber temp1 = sub(c1, c2);

ComplexNumber temp2 = mul(c1, c2);

ComplexNumber temp3 = div(c1, c2);

System.out.print("Sum is: "+ temp.real+" + "+ temp.img +"i");

System.out.printf("\n");

System.out.println("Sub is: "+ temp1.real+" + "+ temp1.img +"i");

System.out.println("mul is: "+ temp2.real+" +"+ temp2.img +"i");

System.out.println("div is: "+ temp3.real+" + "+ temp3.img +"i");

}

}

**2.Write the class Date having attributes like day, month & year. Add default & parameterized constructors. Add getters & setters. Add method to print the date. Add method to swap two dates.**

class DateIm

{

int day,month,year;

DateIm()

{

//default

}

DateIm(int day,int month,int year)

{

this.day = day;

this.month = month;

this.year = year;

}

void printDate(int day,int month,int year)

{

System.out.println(day + " " + month + " " + year);

}

public int getDay() {

return day;

}

public void setDay(int day) {

this.day = day;

}

public int getMonth() {

return month;

}

public void setMonth(int month) {

this.month = month;

}

public int getYear() {

return year;

}

public void setYear(int year) {

this.year = year;

}

}

public class Date {

public static void main(String[] args) {

DateIm a = new DateIm();

a.printDate(03, 01, 2022);

}

}

**3.Write a class Account & add methods like deposit, withdraw, print etc.**

public class BankAcount{

private int accountNumber;

private double currentBal;

BankAcount()

{

}

BankAcount(int accountNumber)

{

this.accountNumber=accountNumber;

}

public void setAccountNumber(int accountNumber)

{

this.accountNumber=accountNumber;

}

public int getAccountNumber()

{

return accountNumber;

}

public void setCurrentBal(double currentBal)

{

this.currentBal=currentBal;

}

public double getCurrentBal()

{

return currentBal;

}

public void deposit(double amount)

{

this.currentBal += amount;

System.out.println("Your account is credited with Rs." +amount+"Your balance is Rs." +

currentBal );

}

public void Withdraw(double amount)

{

if(currentBal>=amount)

{

this.currentBal-=amount;

System.out.println("Your balance amount after dedecting Rs." + amount + " is Rs." +currentBal);

}

else

System.out.println("Insufficient Balance");

}

public static void main(String[] args) {

BankAcount account1 = new BankAcount();

account1.setAccountNumber(229810000);

System.out.println("Your account number is :" + account1.getAccountNumber());

account1.setCurrentBal(50000);

account1.deposit(10000);

account1.Withdraw(15000);

}

}

**4.Write a program to implement a Stack using arrays as follows**-

class StackedArray {

int ary[];

push(--) { }

pop() {--) {}

}

public class Stack

{

int n = 100;

int arr[] = new int[n];

int top = -1;

void push(int value)

{

if(top == (n-1))

System.out.println("stack is full");

arr[top+1] = value;

top++;

}

void pop()

{

if(top < 0)

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

top = top-1;

}

void display()

{

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

{

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

}

}

public static void main(String[] args) {

Stack s1 = new Stack();

s1.push(20);

s1.push(35);

s1.push(150);

s1.push(256);

s1.push(88);

s1.push(200);

s1.display();

System.out.println("After pop");

s1.pop();

s1.pop();

s1.display();

}

}

**5.Write a program to implement a Queue using arrays as follows-**

**class QueuedArray {**

**int ary[];**

**push(--) { }**

**pop() {--) {}**

**}**

public class Queue

{

int n = 100;

int arr[] = new int[n];

int rear=-1, front = 0;

void enQueue(int value)

{

if(rear == (n-1))

System.out.println("Queue is full");

arr[rear+1] = value;

rear++;

}

void deQueue()

{

if(rear == front)

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

for(int i=0;i<=rear;i++)

arr[i] = arr[i+1];

--rear;

}

void display()

{

for(int i = 0; i<=rear ; i++)

{

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

}

}

public static void main(String[] args) {

Queue s = new Queue();

s.enQueue(55);

s.enQueue(95);

s.enQueue(66);

s.enQueue(40);

s.enQueue(74);

s.enQueue(300);

s.display();

System.out.println("After DeQueue");

s.deQueue();

s.deQueue();

s.display();

}

}

**6.Write a single tone class. Confirm that single tone class cannot be inherited**

public class Singleton

{

public static void main(String[] args) {

Example obj = Example.getObject();

}

}

final class Example

{

private static Example single\_obj = null;

public String str ;

static Example obj = new Example();

private Example()

{

str = "Demo";

}

public static Example getObject()

{

if(single\_obj == null)

single\_obj = new Example();

System.out.println("Singleton obj created");

return single\_obj;

}

}

7. **Write java classes to build doubly linked list. Add functionalities like add new node, insert node, delete node, count nodes & print linked list.**

**class Node {**

**Node previous;**

**Node next;**

**Int data;**

}

class Double

{

Node head,tail = null;

class Node

{

int data;

Node prev;

Node next;

Node(int d)

{

data = d;

}

}

void insert(int data)

{

Node new\_node = new Node(data);

if(head==null)

{

head = tail = new\_node;

head.prev = null;

tail.next = null;

}

tail.next = new\_node;

new\_node.prev = tail;

tail = new\_node;

new\_node.next = null;

}

void delete(Node del)

{

if(head == null )

{

return;

}

if(head == del) {

head = del.next;

}

if(del.next != null) {

del.next.prev = del.prev;

}

if(del.prev != null) {

del.prev.next = del.next;

}

return;

}

void printNodes()

{

Node curr = head;

if(head == null)

{

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

return;

}

while(curr!=null)

{

System.out.print(curr.data + "->" );

curr = curr.next;

}

System.out.println(" ");

}

}

public class DoublyLinkedList

{

public static void main(String[] args) {

Double dl = new Double();

dl.insert(100);

dl.insert(200);

dl.insert(300);

dl.insert(400);

dl.insert(500);

dl.printNodes();

dl.delete(dl.head.next);

dl.delete(dl.tail.prev);

System.out.println("after deletion");

dl.printNodes();

}

}