1. 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.

Code :-

package com.zensar;

public class Date {

public int day;

public int month;

public int year;

public Date() {

super();

this.day = 9;

this.month = 2;

this.year = 2022;

}

public Date(int day, int month, int year) {

super();

this.day = day;

this.month = month;

this.year = 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 void Print() {

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

}

public void Swap() {

Date day1 = new Date();

int day = day1.getDay();

day1.setDay(10);

int day2 = day1.getDay();

System.out.println("Before Swapping :" + day + " " + "&" + " " + day2);

int temp;

temp = day;

day = day2;

day2 = temp;

System.out.println("After Swapping :" + day + " " + "&" + " " + day2);

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Date date = new Date();

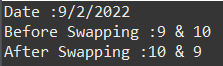
date.Print();

date.Swap();

}

}

Output :-



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

Code :-

package com.zensar;

public class ComplexNumber

{

double real,img;

ComplexNumber()

{

}

ComplexNumber(double real,double img)

{

this.real=real;

this.img=img;

}

public String getValue()

{

return real+"+"+img+"i";

}

public static ComplexNumber swap( ComplexNumber c1, ComplexNumber c2)

{

ComplexNumber complexNumber = new ComplexNumber();

System.out.println("Before swaping-1 =" + c1.real + "+" + c1.img + "i");

complexNumber.real=c1.real;

c1.real=c1.img;

c1.img=complexNumber.real;

System.out.println("after swaping-1= " + c1.real + "+" + c1.img + "i");

complexNumber.real=c2.real;

c2.real=c2.img;

c2.img=complexNumber.real;

System.out.println("Before swaping-2 = " + c2.real + "+" + c2.img + "i");

System.out.println("after swaping-2= " + c2.real + "+" + c2.img + "i");

return complexNumber;

}

public static ComplexNumber add( ComplexNumber c1,ComplexNumber c2)

{

ComplexNumber complexNumber=new ComplexNumber();

complexNumber.real=c1.real+c2.real;

complexNumber.img=c1.img+c2.img;

return complexNumber;

}

public static ComplexNumber sub( ComplexNumber c1,ComplexNumber c2)

{

ComplexNumber complexNumber=new ComplexNumber();

complexNumber.real=c1.real-c2.real;

complexNumber.img=c1.img-c2.img;

return complexNumber;

}

public static ComplexNumber mul( ComplexNumber c1,ComplexNumber c2)

{

ComplexNumber complexNumber=new ComplexNumber();

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

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

return complexNumber;

}

public static void main(String[] args)

{

ComplexNumber a=new ComplexNumber(5,8);

ComplexNumber b=new ComplexNumber(7,9);

System.out.println(" Swapping ");

swap(a,b);

ComplexNumber c=add(a,b);

System.out.println("\*\*\*\*Adding\*\*\*\*");

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

ComplexNumber c1=sub(a,b);

System.out.println("\*\*\*\*Substraction\*\*\*\*");

System.out.println(c1.getValue());

ComplexNumber c2=mul(a,b);

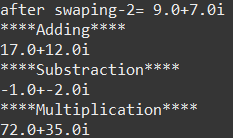
System.out.println("\*\*\*\*Multiplication\*\*\*\*");

System.out.println(c2.getValue());

}

}

Output :-



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

Code :-

**package** com.zensar;

**public** **class** Account

{

**private** **int** accountNumber;

**private** **double** currentBal;

Account(){

}

Account(**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 fund");

}

**public** **static** **void** main(String[] args) {

Account account = **new** Account();

account.setAccountNumber(543742);

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

account.setCurrentBal(15000.0);

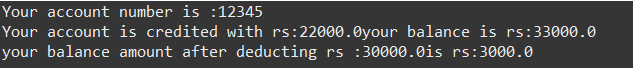
account.deposit(25000.0);

account.Withdraw(35000.0);

}

}

Output :-



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

class StackedArray {

int ary[];

push(--) { }

pop() {--) {}

}

Code :-

package com.zensar;

public class Stack

{

int arry[]=new int[5];

int top=0;

public void push(int data)

{

arry[top]=data;

top++; }

public void pop()

{

int data;

top--;

arry[top]=0;

}

public void display()

{

System.out.println("array elements are : " );

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

{

System.out.println("arr[" + i +"] ="+arry[i]);

}

}

public static void main(String[] args)

{

Stack s = new Stack();

s.push(3);

s.push(4);

s.push(5);

s.push(6);

s.push(7);

s.display();

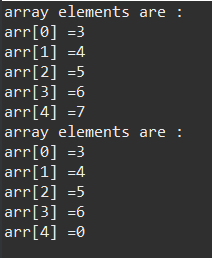
s.pop();

s.display();

}

}

Output :-



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

class QueuedArray {

int ary[];

push(--) { }

pop() {--) {}

}

Code :-

package com.zensar;

public class Queue

{

int n = 200;

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(4);

s.enQueue(2);

s.enQueue(8);

s.enQueue(5);

s.enQueue(6);

s.display();

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

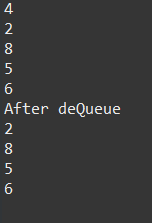
s.deQueue();

s.display();

}

}

Output :-



1. Write a single tone class. Confirm that single tone class cannot be inherited.

Code :-

package com.zensar;

public class Singletone

{

private static Singletone singletone=null;

public static Singletone singletoneMethod()

{

if(singletone == null)

{

System.out.println("object is null---create object");

singletone=new Singletone();

return singletone;

}

else

{

System.out.println("object cannot be created");

return singletone;

}

}

public static void main(String[] args) {

Singletone singletone=Singletone.singletoneMethod();

Singletone singletone1=Singletone.singletoneMethod();

}

}

Output :-



1. 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;

}

Code :-

package com.zensar;

public class Doublylinkedlist

{

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 int countNodes() {

int counter = 0;

Node current = head;

while(current != null) {

counter++;

current = current.next;

}

return counter;

}

public static void main(String[] args) {

Doublylinkedlist dl = new Doublylinkedlist();

dl.insert(3);

dl.insert(8);

dl.insert(4);

dl.insert(5);

dl.printNodes();

dl.delete(dl.head.next);

dl.delete(dl.tail.prev);

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

dl.printNodes();

System.out.println("No. of nodes: "+ dl.countNodes());

}

}

Output :-

