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.

**package** com.zensar;

**public** **class** Date

{

Date()

{

**this**.day="13";

**this**.month="01";

**this**.year="2022";

}

Date(String day,String month,String year)

{

**this**.day=day;

**this**.month=month;

**this**.year=year;

}

**private** String day;

**private** String month;

**private** String year;

**public** **void** setDay(String day)

{

**this**.day=day;

}

**public** **void** setmonth(String month)

{

**this**.month=month;

}

**public** **void** setyear(String year)

{

**this**.year=year;

}

**public** String getDay()

{

**return** day;

}

**public** String getMonth()

{

**return** month;

}

**public** String getyear()

{

**return** year;

}

**public** **static** **void** printDate()

{

Date d = **new** Date();

String day=d.getDay();

String month=d.getMonth();

String year=d.getyear();

System.***out***.println("stored date : " + day+ "/" + month+"/" + year);

}

**public** **static** **void** swapDate()

{

Date d = **new** Date();

String day=d.getDay();

String month=d.getMonth();

String year=d.getyear();

d.setDay("Monday");

String day2=d.getDay();

System.***out***.println(" before swapping = "+ day + " " + day2);

String temp;

temp=day;

day=day2;

day2=temp;

System.***out***.println(" after swapping ="+ day + " " + day2);

}

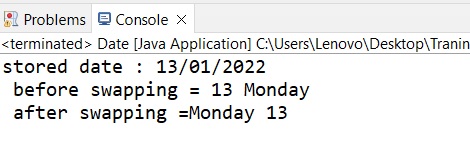
**public** **static** **void** main(String[] args) {

*printDate*();

*swapDate*();

}}

OUTPUT:



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

**package** com.zensar;

**public** **class** Complex

{

**double** real,img;

Complex()

{

}

Complex(**double** real,**double** img)

{

**this**.real=real;

**this**.img=img;

}

**public** String getValue()

{

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

}

**public** **static** Complex swap( Complex c1, Complex c2)

{

Complex complex = **new** Complex();

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

complex.real=c1.real;

c1.real=c1.img;

c1.img=complex.real;

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

complex.real=c2.real;

c2.real=c2.img;

c2.img=complex.real;

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

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

**return** complex;

}

**public** **static** Complex add( Complex c1,Complex c2)

{

Complex complex=**new** Complex();

complex.real=c1.real+c2.real;

complex.img=c1.img+c2.img;

**return** complex;

}

**public** **static** Complex sub( Complex c1,Complex c2)

{

Complex complex=**new** Complex();

complex.real=c1.real-c2.real;

complex.img=c1.img-c2.img;

**return** complex;

}

**public** **static** Complex mul( Complex c1,Complex c2)

{

Complex complex=**new** Complex();

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

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

**return** complex;

}

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

{

Complex a=**new** Complex(5,6);

Complex b=**new** Complex(7,9);

System.***out***.println(" Swapping ");

*swap*(a,b);

Complex c=*add*(a,b);

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

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

Complex c1=*sub*(a,b);

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

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

Complex c2=*mul*(a,b);

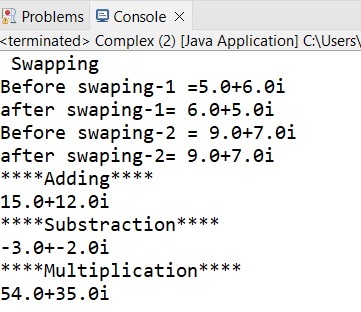
System.***out***.println("\*\*\*\*Multiplication\*\*\*\*");

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

}

}

OUTPUT:



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

**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 Balance");

}

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

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

account.setAccountNumber(101213);

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

account.setCurrentBal(10000);

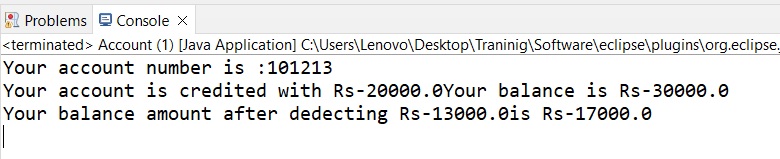
account.deposit(20000);

account.Withdraw(13000);

}

}

OUTPUT:



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

Class Stacked Array {

int ary[];

push (--) { }

pop () {--) {}

}

**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** show()

{

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(21);

s.push(22);

s.push(23);

s.push(24);

s.push(25);

s.show();

s.pop();

s.show();

s.pop();

s.show();

s.pop();

s.show();

s.pop();

s.show();

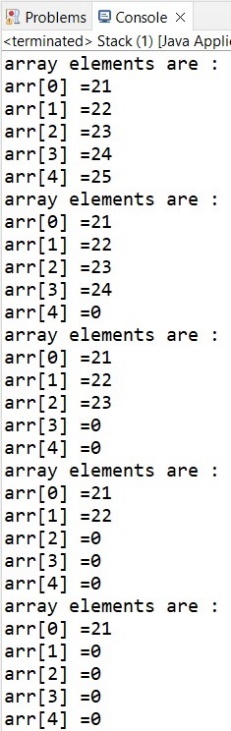
s.pop();

s.show();

}

}

OUTPUT:



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

class Queued Array {

int a ry[];

push(--) { }

pop() {--) {}

}

**package** com.zensar;

**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(25);

s.enQueue(200);

s.enQueue(45);

s.enQueue(36);

s.enQueue(15);

s.enQueue(99);

s.display();

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

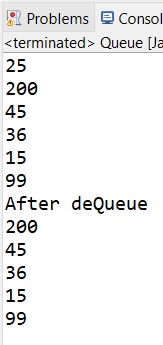
s.deQueue();

s.display();

}

}

OUTPUT:



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

**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) {

// **TODO** Auto-generated method stub

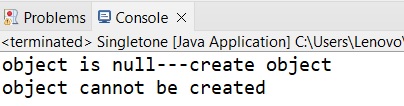
Singletone singletone=Singletone.*singletoneMethod*();

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;

}

**package** com.zensar;

**class** DLL

{

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) {

DLL dl = **new** DLL();

dl.insert(21);

dl.insert(22);

dl.insert(23);

dl.insert(24);

dl.insert(25);

dl.printNodes();

dl.delete(dl.head.next);

dl.delete(dl.tail.prev);

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

dl.printNodes();

//number of nodes in the list

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

}

}

OUTPUT: 