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.assignment;

**public** **class** Date

{

Date()

{

**this**.day="01";

**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*();

}

}

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

**package** com.zensar.assignment;

**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());

}

}

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

**package** com.zensar.assignment;

**import** java.util.Scanner;

**public** **class** Account {

Scanner sc= **new** Scanner(System.***in***);

**long** balance=10000, amt;

**void** withraw() {

System.***out***.println("Enter the amount to withdraw:" +amt);

amt=sc.nextLong();

**if** (balance >= amt) {

balance=balance-amt;

System.***out***.println("Balance after withdrawal: " + balance);

} **else** {

System.***out***.println("Your balance is less than " + amt + "\tTransaction failed...!!" );

}

}

**void** deposit() {

**long** amt;

System.***out***.println("Enter the amount you want to deposit: ");

amt = sc.nextLong();

balance = balance + amt;

System.***out***.println("Balance after deposit" +balance);

}

**void** print() {

System.***out***.println("Available balance:" +balance);

}

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

Account acc=**new** Account();

acc.print();

acc.withraw();

acc.deposit();

}

}

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

Class Stacked Array {

int ary[];

push (--) { }

pop () {--) {}

}

**package** com.zensar.assignment;

**import** java.util.\*;

**import** java.util.Arrays;

**public** **class** Stack {

**int** array[]=**new** **int**[6];

**public** **void** push(){

Scanner sc= **new** Scanner(System.***in***);

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

array[i]=sc.nextInt();

}

System.***out***.println("Stack Element after push" +Arrays.*toString*(array));

pop(array);

}

**public** **void** pop(**int** array[]) {

**this**.array=array;

**if**(array.length!=0) {

**int** newArr[] = Arrays.*copyOf*(array, array.length - 1);

System.***out***.println("Elemnts after pop" +Arrays.*toString*(newArr));

pop(newArr);

}

**else** {

System.***out***.println("Empty Stack after pop");

}

}

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

{

Stack st=**new** Stack();

System.***out***.println("Enter the Elements of Stack");

st.push();

}

}

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

class Queued Array {

int ary[];

push(--) { }

pop() {--) {}

}

**package** com.zensar.assignment;

**import** java.util.\*;

**public** **class** Queuee {

**int** ary[]=**new** **int**[6];

**public** **void** push(){

Scanner sc= **new** Scanner(System.***in***);

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

ary[i]=sc.nextInt();

}

System.***out***.println("Queue Element after push" +Arrays.*toString*(ary));

pop(ary);

}

**public** **void** pop(**int** ary[]) {

**this**.ary=ary;

**if**(ary.length!=0) {

**int**[] newArr = Arrays.*copyOfRange*(ary,1,ary.length);

System.***out***.println("Elemnts after pop of index" +Arrays.*toString*(newArr));

pop(newArr);

}

**else** {

System.***out***.println("Empty Queue after pop");

}

}

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

{

Queue q=**new** Queue();

System.***out***.println("Enter the Elements of Queue");

q.push();

}

}

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

**package** com.zensar.assignment;

**public** **class** Singleton

{

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

SingletonExample object = SingletonExample.*getObject*();

}

}

**final** **class** SingletonExample

{

**private** **static** SingletonExample *single\_object* = **null**;

**public** String s ;

**static** SingletonExample *object* = **new** SingletonExample();

**private** SingletonExample()

{

s = "Demo of SingletonExample";

}

**public** **static** SingletonExample getObject()

{

**if**(*single\_object* == **null**)

*single\_object* = **new** SingletonExample();

System.***out***.println("Singleton object is created");

**return** *single\_object*;

}

}

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;

}

**package** com.zensar.assignment;

**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();

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

}

}