Assignement for core java classes and objects.

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.project;

**public** **class** Date {

**int** day,month,year;

**public** **int** getDay() {

//Getter used for day//

**return** day;

}

**public** **int** setday(**int** day) {

//setter used for day//

**return** **this**.day=day;

}

**public** **int** getMonth() {

//Getter used for month//

**return** month;

}

**public** **int** setMonth(**int** month) {

//setter used for month//

**return** **this**.month=month;

}

**public** **int** getYear() {

//Getter used for year//

**return** year;

}

**public** **int** setYear(**int** year) {

//setter used for year//

**return** **this**.year=year;

}

Date(){

}

Date(**int** a,**int** b, **int** c){

day=a;

month=b;

year=c;

}

**void** show\_date() {

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

}

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

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

d.setday(10);//calling setter function of day to set day//

d.setMonth(11);//calling setter function of month to set month//

d.setYear(1999);//calling setter function of year to set year//

d.show\_date();//printing date//

Date d1=**new** Date(3,12,1998);//calling parameterized constructor of Date with arguments//

d1.show\_date();//printing date of parameterized constructor//

}

}

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

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**package** com.project;

**public** **class** ComplexNumber {

**double** real,imaginary,sum;

ComplexNumber(**double** real,**double** imaginary){

**this**.real=real;

**this**.imaginary=imaginary;

}

**static** ComplexNumber add(ComplexNumber n1,ComplexNumber n2) {

ComplexNumber res = **new** ComplexNumber(0, 0);

res.real=n1.real+n2.real;

res.imaginary=n1.imaginary+n2.imaginary;

**return** res;

}

**static** ComplexNumber sub(ComplexNumber n1,ComplexNumber n2) {

ComplexNumber res1 = **new** ComplexNumber(0, 0);

res1.real=n1.real-n2.real;

res1.imaginary=n1.imaginary-n2.imaginary;

**return** res1;

}

**static** ComplexNumber mul(ComplexNumber n1,ComplexNumber n2) {

ComplexNumber res2 = **new** ComplexNumber(0, 0);

res2.real=n1.real\*n2.real;

res2.imaginary=n1.imaginary\*n2.imaginary;

**return** res2;

}

//

**void** showComplexNumber() {

System.***out***.println(**this**.real+ "+i"+**this**.imaginary);

}

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

ComplexNumber c1=**new** ComplexNumber(3.5,4.5);

ComplexNumber c2=**new** ComplexNumber(3.7,4.6);

ComplexNumber res = *add*(c1, c2);

res.showComplexNumber();

ComplexNumber res1 = *sub*(c1, c2);

res1.showComplexNumber();

ComplexNumber res2 = *mul*(c1, c2);

res2.showComplexNumber();

}

}

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

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

**package** com.project;

**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();

//acc.print();

}

}

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

class StackedArray {

int ary[];

push(--) { }

pop() {--) {}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**package** com.project;

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

**import** java.util.Arrays;

**public** **class** Stackk {

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

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

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

// int ary[]= {2, 4, 6, 8};

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

ary[i]=sc.nextInt();

}

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

pop(ary);

}

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

**this**.ary=ary;

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

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

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

pop(newArr);

}

// if(newArr.length==-1) {

**else** {

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

}

}

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

{

Stackk st=**new** Stackk();

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

st.push();

}

}

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

class QueuedArray {

int ary[];

push(--) { }

pop() {--) {}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**package** com.project;

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

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

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

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

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

// int ary[]= {2, 4, 6, 8};

**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);

// int newArr[] = Arrays.copyOf(ary, ary.length- 1);

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

pop(newArr);

}

// if(newArr.length==-1) {

**else** {

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

}

}

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

{

Queuee q=**new** Queuee();

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.project;

**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.project;

**import** java.util.Scanner;

**public** **class** Node {

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

**public** **class** Node1{

**int** node;

Node1 previous;

Node1 next;

**public** Node1(**int** node) {

**this**.node = node;

} }

Node1 h, t = **null**;

**public** **void** add\_Node(**int** node) {

//Create a new node

Node1 newNode = **new** Node1(node);

**if**(h == **null**) {

h = t = newNode;

h.previous = **null**;

t.next = **null**;

}

**else** {

t.next = newNode;

newNode.previous = t;

t = newNode;

t.next = **null**;

}

}

**public** **void** print\_list() {

Node1 current = h;

**if**(h == **null**) {

System.***out***.println("List is empty");

**return**;

}

System.***out***.println("Nodes of doubly linked list: ");

**while**(current != **null**) {

System.***out***.print(current.node + " ");

current = current.next;

}

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

}

**public** **void** countNodes() {

**int** counter = 0;

//Node current will point to head

Node1 current = h;

**while**(current != **null**) {

//Increment the counter by 1 for each node

counter++;

current = current.next;

}

System.***out***.println("Total no of nodes are:" +counter);

}

**public** **void** del\_node(Node1 del) {

**if**(h == **null** )

{

**return**;

}

**if**(h == del) {

h = del.next;

}

**if**(del.next != **null**) {

del.next.previous = del.previous;

}

**if**(del.previous != **null**) {

del.previous.next = del.next;

}

**return**;

}

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

Node n=**new** Node();

n.add\_Node(1);

n.add\_Node(4);

n.add\_Node(5);

n.add\_Node(6);

n.add\_Node(7);

n.add\_Node(8);

n.add\_Node(12);

n.print\_list();

n.countNodes();

}

}