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.

//Creating DateClass to Implement properties and methods

**package** com.zensar;

**public** **class** DateClass {

**private** **int** day;

**private** String month;

**private** **int** year;

**public** DateClass(){

day = 0;

month = **null**;

year = 0;

}

**public** DateClass(**int** day, String 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** String getMonth() {

**return** month;

}

**public** **void** setMonth(String month) {

**this**.month = month;

}

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

**return** year;

}

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

**this**.year = year;

}

**public** **void** SwapTwoDates(DateClass obj)

{

**int** day\_Of\_Current\_Object, year\_Of\_Current\_Object;

String month\_Of\_Current\_Object;

day\_Of\_Current\_Object = **this**.day;

month\_Of\_Current\_Object = **this**.month;

year\_Of\_Current\_Object = **this**.year;

**this**.day = obj.day;

**this**.month = obj.month;

**this**.year = obj.year;

obj.day = day\_Of\_Current\_Object;

obj.month = month\_Of\_Current\_Object;

obj.year = year\_Of\_Current\_Object;

}

}

//Creating Starter Class

**package** com.zensar;

**public** **class** class\_n\_objects {

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

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

//Creating objects

DateClass dateObject = **new** DateClass();

DateClass dateObject1 = **new** DateClass();

//With Set Method Setting Day, Month, Year

dateObject.setDay(9);

dateObject.setMonth("Feb");

dateObject.setYear(2022);

// Printing Day, Month, Year

System.***out***.println("The Date is : " + dateObject.getDay() + "/"+ dateObject.getMonth()+ "/" + dateObject.getYear());

//With Set Method Setting Day, Month, Year

dateObject1.setDay(11);

dateObject1.setMonth("Jan");

dateObject1.setYear(2022);

// Printing Day, Month, Year

System.***out***.println("The Date is : " + dateObject1.getDay() + "/" + dateObject1.getMonth() + "/" + dateObject1.getYear());

//Swapping two objects properties by calling SwapTwoDates Method

dateObject1.SwapTwoDates(dateObject);

//Printing out Day,Month,Year for dateObject

System.***out***.println("The Date is : " + dateObject.getDay() + "/" + dateObject.getMonth() + "/" + dateObject.getYear());

//Printing out Day,Month,Year for dateObject1

System.***out***.println("The Date is : " + dateObject1.getDay() + "/" + dateObject1.getMonth() + "/" + dateObject1.getYear());

}

}

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

**package** com.zensar;

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

**private** **int** real\_number;

**private** **int** imaginary\_number;

**public** ComplexNumber(**int** real\_number, **int** imaginary\_number) {

**super**();

**this**.real\_number = real\_number;

**this**.imaginary\_number = imaginary\_number;

}

**public** **void** Add(ComplexNumber obj)

{

**this**.real\_number += obj.real\_number;

**this**.imaginary\_number += obj.imaginary\_number;

}

**public** **void** Substract(ComplexNumber obj)

{

**this**.real\_number -= obj.real\_number;

**this**.imaginary\_number -= obj.imaginary\_number;

}

**public** **void** Multiple(ComplexNumber obj)

{

**int** real\_number\_for\_an\_object = 0, imaginary\_number\_for\_an\_object = 0;

**boolean** negative = **false**;

**if**(**this**.imaginary\_number < 0 && obj.imaginary\_number < 0)

negative = **true**;

real\_number\_for\_an\_object = (**this**.real\_number \* obj.real\_number) + ((negative == **true**)?((-1) \* **this**.imaginary\_number \* obj.imaginary\_number): **this**.imaginary\_number \* obj.imaginary\_number);

imaginary\_number\_for\_an\_object =(**this**.real\_number \* obj.imaginary\_number) + (**this**.imaginary\_number \* obj.real\_number);

**this**.real\_number = real\_number\_for\_an\_object;

**this**.imaginary\_number = imaginary\_number\_for\_an\_object;

}

**public** **void** Swap(ComplexNumber obj)

{

**int** real\_number\_for\_an\_object = 0, imaginary\_number\_for\_an\_object = 0;

real\_number\_for\_an\_object = **this**.real\_number;

imaginary\_number\_for\_an\_object = **this**.imaginary\_number;

**this**.real\_number = obj.real\_number;

**this**.imaginary\_number = obj.imaginary\_number;

obj.real\_number = real\_number\_for\_an\_object;

obj.imaginary\_number = imaginary\_number\_for\_an\_object;

}

**public** **void** display()

{

**if**(**this**.imaginary\_number > 0)

System.***out***.println("The Complex number is : " + **this**.real\_number + "+" + **this**.imaginary\_number+"i");

**else**

System.***out***.println("The Complex number is : " + **this**.real\_number + **this**.imaginary\_number+"i");

}

}

//Creating Starter Class

**package** com.zensar;

**public** **class** class\_n\_objects {

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

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

//Creating objects

ComplexNumber complexnumber\_1 = **new** ComplexNumber(4,-7);

ComplexNumber complexnumber\_2 = **new** ComplexNumber(7,5);

//Printing out complexnumber\_1 properties

complexnumber\_1.display();

//Printing out complexnumber\_2 properties

complexnumber\_2.display();

//Adding complexnumber\_1 with complexnumber\_2

complexnumber\_1.Add(complexnumber\_2);

//Printing out complexnumber\_1 properties

complexnumber\_1.display();

complexnumber\_1.Multiple(complexnumber\_2);

//Printing out complexnumber\_1 properties

complexnumber\_1.display();

}

}

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

**package** com.zensar;

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

**private** **int** amount;

**public** Account(**int** amount) {

**super**();

**this**.amount = amount;

}

**public** **int** withdraw(**int** amount)

{

**return** **this**.amount -= amount;

}

**public** **int** deposit(**int** amount)

{

**return** **this**.amount += amount;

}

**public** **void** display()

{

System.***out***.println("The Balance left in Account : " + **this**.amount);

}

**public** **void** display(**int** amount)

{

System.***out***.println("The Balance left in Account : " + amount);

}

}

//Creating Starter Class

**package** com.zensar;

**public** **class** class\_n\_objects {

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

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

//Creating objects

Account accountObject = **new** Account(14000);

//Displaying balance in account

accountObject.display();

//Displaying balance in account after getting amount from deposit function

accountObject.display(accountObject.deposit(5000));

//Displaying balance in account after getting amount from withdraw function

accountObject.display(accountObject.withdraw(10000));

}

}

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

**package** com.zensar;

**public** **class** StackedArray {

**private** **int** n = 100;

**private** **int**[] arr = **new** **int**[n];

**private** **int** top = -1;

**public** **void** push(**int** value)

{

**if**(top == n)

{

System.***out***.println("Stack Overflow because stack is full now ");

**return**;

}

++top;

arr[top] = value;

}

**public** **void** pull()

{

**if**(top < 0)

{

System.***out***.println("Stack Underflow because Stack is empty already");

top = -1;

**return**;

}

arr[top] = 0;

--top;

}

**public** **void** display()

{

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

{

System.***out***.print(arr[i] + " ");

}

**if**(top < 0)

{

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

}

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

}

}

//Creating Starter Class

**package** com.zensar;

**public** **class** class\_n\_objects {

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

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

//Creating objects;

StackedArray stackObject = **new** StackedArray();

stackObject.push(9);

System.***out***.println("After Pushing Elements Into Stack");

stackObject.display();

System.***out***.println("After Pulling out Elements from Stack");

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

stackObject.pull();

stackObject.pull();

stackObject.display();

stackObject.push(7);

stackObject.push(10);

stackObject.push(19);

stackObject.push(32);

stackObject.push(78);

System.***out***.println("After Pushing Elements Into Stack");

stackObject.display();

stackObject.pull();

stackObject.pull();

stackObject.pull();

System.***out***.println("After Pulling out Elements from Stack");

stackObject.display();

}

}

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

**package** com.zensar;

**public** **class** QueuedArray {

**private** **int** n = 100;

**private** **int**[] arr = **new** **int**[n];

**private** **int** enqueue = -1 , dequeue = -1;

**public** **void** push(**int** value)

{

**if**(enqueue == n)

{

System.***out***.println("Stack Overflow because stack is full now ");

**return**;

}

++enqueue;

arr[enqueue] = value;

}

**public** **void** pull()

{

**if**(enqueue == dequeue)

{

System.***out***.println("Stack Underflow because Stack is empty already");

dequeue = -1;

**return**;

}

++dequeue;

System.***out***.print(arr[dequeue] + " ");

arr[dequeue] = 0;

}

**public** **void** display()

{

**for**(**int** i=0;i<=enqueue;i++)

{

System.***out***.print(arr[i] + " ");

}

**if**(enqueue < 0)

{

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

}

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

}

}

//Creating Starter Class

**package** com.zensar;

**public** **class** class\_n\_objects {

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

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

//Creating objects

QueuedArray queueObject = **new** QueuedArray();

//Adding Element in Queue

queueObject.push(9);

System.***out***.println("After Pushing Elements Into Queue");

queueObject.display();

System.***out***.println("After Pulling out Elements from Queue");

//Pull out Elements from Queue

queueObject.pull();

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

queueObject.pull();

//Displaying Available Element in Queue

queueObject.display();

//Adding Elements into Queue

queueObject.push(7);

queueObject.push(10);

queueObject.push(19);

queueObject.push(32);

queueObject.push(78);

System.***out***.println("After Pushing Elements Into Queue");

queueObject.display();

//Removing Elements from Queue

queueObject.pull();

queueObject.pull();

queueObject.pull();

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

System.***out***.println("After Pulling out Elements from Queue");

queueObject.display();

}

}

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

**package** com.zensar;

**public** **class** SingletonClass {

**private** **static** SingletonClass *instance* = **null**;

**public** String str;

**private** SingletonClass()

{

}

**public** **static** SingletonClass getInstance() {

**if** (*instance* == **null**){

*instance* = **new** SingletonClass();

}

**return** *instance*;

}

}

**package** com.zensar;

**public** **class** class\_n\_objects {

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

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

//creating objects for singletonClass

SingletonClass singletonReference\_1 = SingletonClass.*getInstance*();

singletonReference\_1.str = "This is singletonReference\_1";

SingletonClass singletonReference\_2 = SingletonClass.*getInstance*();

singletonReference\_2.str = "This is singletonReference\_2";

SingletonClass singletonReference\_3 = SingletonClass.*getInstance*();

singletonReference\_3.str = "This is singletonReference\_3";

System.***out***.println(singletonReference\_1.str);

System.***out***.println(singletonReference\_2.str);

System.***out***.println(singletonReference\_3.str);

}

}

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

**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(" ");

}

}

**package** com.zensar;

**public** **class** class\_n\_objects {

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

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

//Creating Doubly Objects

DoublyLinkedList dl = **new** DoublyLinkedList();

dl.insert(10);

dl.insert(20);

dl.insert(30);

dl.insert(40);

dl.insert(50);

dl.printNodes();

dl.delete(dl.head.next);

dl.delete(dl.tail.prev);

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

dl.printNodes();

}

}