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.

Date.java

package com.zensar;

public class Date {

private int day,month,year,temp;

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

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 swap(int d,int m,int y){

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

System.out.println("Date 1= "+this.getDay()+"-"+this.getMonth()+"-"+this.getYear());

System.out.println("Date 2= "+d+"-"+m+"-"+y);

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

temp=this.day;

this.day=d;

d=temp;

temp=this.month;

this.month=m;

m=temp;

temp=this.year;

this.year=y;

y=temp;

System.out.println("Date 1= "+this.getDay()+"-"+this.getMonth()+"-"+this.getYear());

System.out.println("Date 2= "+d+"-"+m+"-"+y);

}

}

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

ComplexNumber.java

public class ComplexNumber {

private double real,imaginary;

private double temp=0;

public ComplexNumber(double real,double imaginary){

this.real=real;

this.imaginary=imaginary;

}

public double getReal() {

return real;

}

public void setReal(double real) {

this.real = real;

}

public double getImaginary() {

return imaginary;

}

public void setImaginary(double imaginary) {

this.imaginary = imaginary;

}

public void add(double real,double imaginary){

this.real+=real;

this.imaginary+=imaginary;

}

public void subtract(double real,double imaginary){

this.real-=real;

this.imaginary-=imaginary;

}

public void multiply(double real,double imaginary){

this.real\*=real;

this.imaginary\*=imaginary;

}

public void swap(){

temp=this.real;

this.real=this.imaginary;

this.imaginary+=temp;

}

}

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

Account.java

package com.zensar;

public class Account {

private int accountNumber;

private double currentBalance;

private String name;

public Account(int accountNumber, double currentBalance, String name) {

this.accountNumber = accountNumber;

this.currentBalance = currentBalance;

this.name = name;

}

public int getAccountNumber() {

return accountNumber;

}

public void setAccountNumber(int accountNumber) {

this.accountNumber = accountNumber;

}

public double getCurrentBalance() {

return currentBalance;

}

public void setCurrentBalance(double currentBalance) {

this.currentBalance = currentBalance;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public void deposit(double amount){

this.currentBalance+=amount;

System.out.println("Amount Deposited = "+amount);

}

public void withdraw(double amount){

if(this.currentBalance<amount){

System.out.println("Insufficient Balance");

}

else{

this.currentBalance-=amount;

System.out.println("Amount withdrawn = "+amount);

}

}

public void print(){

System.out.println("Name = "+this.name);

System.out.println("Account Number = "+this.accountNumber);

System.out.println("Current Balance = "+this.currentBalance);

}

}

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

class StackedArray {

int ary[];

push(--) { }

pop() {--) {}

}

StackedArray.java

package com.zensar;

public class StackedArray {

static final int max = 1000;

int top=-1;

int[] a = new int[max];

public void push(int x){

if(top> max-1){

System.out.println("Stack Overflow");

}

else{

a[++top]=x;

System.out.println(x +" is inserted into stack");

}

}

public void pop(){

if(top<0){

System.out.println("Stack Underflow");

}

else{

int x=a[top--];

System.out.println(x+" is popped from stack");

}

}

public void print(){

for(int i=top;i>-1;i--){

System.out.println(" "+a[i]);

}

}

}

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

class QueuedArray {

int ary[];

push(--) { }

pop() {--) {}

}

QueuedArray.java

package com.zensar;

public class QueuedArray {

private int front=0;

private int back=0;

private int capacity=100;

int[] q=new int[capacity];

public void push(int x){

if(capacity==back)

System.out.println("Queue is full");

else{

q[back]=x;

System.out.println(x+" is push into Queue");

back++;

}

}

public void pop(){

if(front==back)

System.out.println("Queue is empty");

else{

int x=q[0];

for(int i=0;i<back;i++){

q[i]=q[i+1];

}

back--;

System.out.println(x+" is removed from the Queue");

}

}

public void display(){

System.out.println("Elements in queue are:");

for(int i=0;i<back;i++){

System.out.println(q[i]);

}

}

}

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

Singleton.java

public class Singleton {  
  
 private static Singleton *instance*;  
  
 protected Singleton() {  
 }  
  
 public static synchronized Singleton getInstance() {  
 if (*instance* == null) {  
 *instance* = new Singleton();  
 System.*out*.println("Parent Instance Created");  
 }  
  
 return *instance*;  
 }  
}  
  
class ChildSingleton extends Singleton {  
  
 private static ChildSingleton *instance*;  
  
 public static synchronized Singleton getInstance() {  
 if (*instance* == null) {  
 *instance* = new ChildSingleton();  
 System.*out*.println("Child Instance is Created");  
 }  
  
 return *instance*;  
 }  
}

Output

E:\Inheritence and Polymorphism\src\main.java:4:52

java: incompatible types: Singleton cannot be converted to ChildSingleton.

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;

}

DoublyLinkedList.java

public class DoublyLinkedList {  
 class Node{  
 int data;  
 Node previous;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 }  
 }  
 Node head,tail=null;  
 public void addNode(int data){  
 Node n=new Node(data);  
 if(head==null){  
 head=tail=n;  
 head.previous=null;  
 tail.next=null;  
 }  
 else{  
 tail.next=n;  
 n.previous=tail;  
 tail=n;  
 tail.next=null;  
 }  
 }  
 public void display(){  
 Node current=head;  
 if(head==null){  
 System.*out*.println("Empty List");  
 return;  
 }  
 System.*out*.println("Nodes of Linked List");  
 while(current!=null){  
 System.*out*.println(current.data+" ");  
 current=current.next;  
 }  
 }  
}