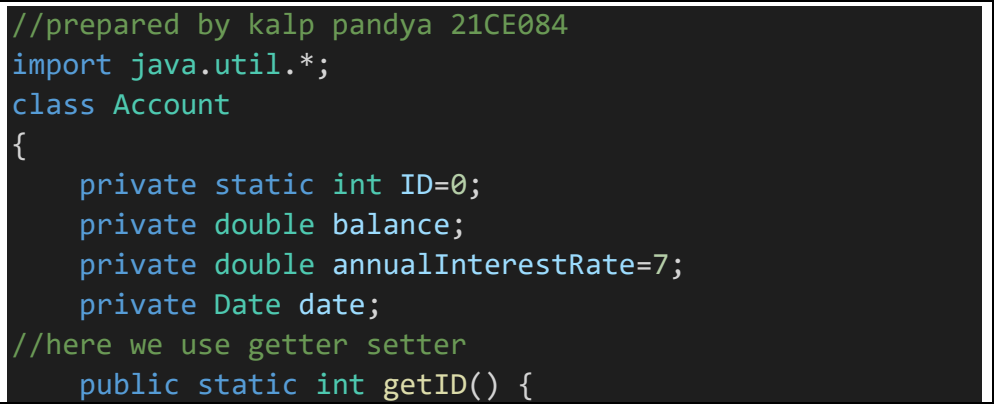
Practical Assignment

Github link:

https://github.com/kalp104/java_assignment.git

Qustion 1 :	Design a class named Account that contains: • A private int data field named id for the account (default 0). • A private double data field named balance for the account (default 500₹). • A private double data field named annualInterestRate that stores the current interest rate (default 0%). Assume all accounts have the same interest rate. • A private Date data field named dateCreated that stores the date when the account was created. • A no-arg constructor that creates a default account. • A constructor that creates an account with the specified id and initial balance. • The accessor and mutator methods for id, balance, and annualInterestRate. • The accessor method for dateCreated. • A method named getMonthlyInterestRate() that returns the monthly interest rate. • A method named getMonthlyInterest() that returns the monthly interest. • A method named withdraw that withdraws a specified amount from the account. • A method named deposit that deposits a specified amount to the account.
Code:	

```java
//prepared by kalp pandya 21CE084
import java.util.*;
class Account
{
    private static int ID=0;
    private double balance;
    private double annualInterestRate=7;
    private Date date;
//here we use getter setter
    public static int getID() {
```

```
        return ID;

    }
    public double getBalance() {
        return this.balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public double getAnnualInterestRate() {
        return this.annualInterestRate;
    }

    public void setAnnualInterestRate(double
annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }

    public Date getDate() {
        return this.date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

//here we use Default constructor
    public Account()
    {
        ID = 0;
        balance = 0;
        annualInterestRate = 0;
    }
//here we use paramiterrized constructor
    public Account(int id,double b)
    {
        ID = id;
        balance = b;
    }
//here we make some methods for application
    public double getMonthlyInterestRate()
    {
        return ((float)annualInterestRate/12);
```

```
    }

    public double getMonthlyInterest()
    {
        return ((float)annualInterestRate/100)/12;
    }

    public void withdraw(double a)
    {
        balance-=a;
        System.out.println("Your current balance is:-
"+balance);
    }

    public String toString()
    {
        Date d = new Date();
        System.out.println(d);
        setDate(d);
        return ("Account id is :- " + getID() + "\nYour
Balance is:- " + getBalance() + "\n Account created on
date:- " + d + "\nMonthly Interest
is:-  "+getMonthlyInterest());
    }
}
```

```
//prepared by kalp pandya 21CE084
import java.util.*;
public class TestAccount {
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        Date D = new Date();


        //call the Static function directly by the class
name.
        double b;
        System.out.println("Enter Your Balance:-");
        b = sc.nextDouble();
        Account a = new Account(Account.getID(),b);
        a.setDate(D);
        System.out.println("Date is :- "+ a.getDate());
        double amount;
```

```
        System.out.println("enter the amount to be
withdrawn:-");
        amount = sc.nextDouble();
        a.withdraw(amount);
        b = a.getMonthlyInterestRate();
        System.out.println("Your monthly interest rate
is:-"+b);
        b = a.getMonthlyInterest();
        System.out.println("Your monthly interest is:-
"+b);

        sc.close();
    }
}
```

output

```
Enter Your Balance:-
10000
Date is :- Sun Oct 09 23:26:53 IST 2022
enter the amount to be withdrawn:-
4000
Your current balance is:-6000.0
Your monthly interest rate is:-0.5833333134651184
Your monthly interest is:-0.005833333358168602
```

Question 2

In an n-sided regular polygon, all sides have the same length and all angles have the same degree (i.e., the polygon is both equilateral and equiangular). Design a class named RegularPolygon that contains:

- A private int data field named n that defines the number of sides in the polygon with default value 3.
- A private double data field named side that stores the length of the side with default value 1.
- A private double data field named x that defines the x-coordinate of the polygon's center with default value 0.
- A private double data field named y that defines the coordinate of the polygon's center with default value 0.
- A no-arg constructor that creates a regular polygon with default values. A constructor that creates a regular polygon with the specified number of sides and length of side, centered at (0, 0).
- A constructor that creates a regular polygon with the specified number of sides, length of side, and x- and y-coordinates.
- The accessor and mutator methods for all data fields.
- The method getPerimeter() that returns the perimeter of the polygon.

The method getArea() that returns the area of the polygon. The formula for computing the area of a regular polygon is:

code

```java
// prepared by kalp pandya 21CE084
import java.math.*;

public class aasignment_prac2{

    static double pi = 3.14;
    private int a;
    private double l;
    private double b;
    private double c;

    public aasignment_prac2(){
        a= 3;
        l = 1;
        b = 0;
        c = 0;
    }

    public int geta() {
        return a;
    }

    public void seta(int n) {
        a = n;
    }

    public double getl() {
        return l;
    }

    public void setSide(double l) {
        this.l = l;
    }

    public double getb() {
        return b;
    }

    public void setb(double x) {
        this.b = b;
    }

    public double getc() {
        return c;
    }
```

```
    public void setc(double c) {
        this.c = c;
    }

    public aasignment_prac2(int n, double l){
        a = n;
        this.l=l;
        b = 0;
        c = 0;
    }

    public aasignment_prac2(int n, double l, double x,
double y){
        a = n;
        this.l=l;
        b = x;
        c = y;
    }

    public double getPerimeter() {
        return a*a;
    }

    public double getArea() {
        return (a*l*l)/(4*Math.tan(pi/a));
    }

    public void print() {
        System.out.println("No. of sides : " + a);
        System.out.println("Length of sides : " + l);
        System.out.println("Perimeter of Polygon : " +
getPerimeter());
        System.out.println("Area of Polygon : " +
getArea());
        System.out.println();
    }

}
```

```
public class prac2_main{

    public static void main(String[] args) {

        aasignment_prac2 obj1 = new aasignment_prac2();
```

```
        aasignment_prac2 obj2 = new aasignment_prac2(3,
12);
        aasignment_prac2 obj3 = new aasignment_prac2(20,
1000, 12, 10);

        System.out.println("Polygon 1");
        obj1.print();
        System.out.println("Polygon 2");
        obj2.print();
        System.out.println("Polygon 3");
        obj3.print();


    }

}
```

output

```
Area of Polygon : 0.43354374924141237

Polygon 2
No. of sides : 3
Length of sides : 12.0
Perimeter of Polygon : 9.0
Area of Polygon : 62.43029989076338

Polygon 3
No. of sides : 20
Length of sides : 1000.0
Perimeter of Polygon : 400.0
Area of Polygon : 3.15850360912823 75E7
```

Question 3

Use the Account class created in Programming Exercise 1 to simulate an ATM machine.

- Create 10 accounts in an array with id 0, 1, . . . , 9, and an initial balance of $100.
- The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id.
- Once an id is accepted, the main menu is displayed.
- You can enter choice 1 for viewing the current balance, 2 for withdrawing money, 3 for depositing money, and 4 for exiting the main menu.
- Once the system starts, it will stop by entering number 99.

Code

```
// prepared by kalp pandya 21CE084
import java.util.Scanner;
```

```
import java.util.ArrayList;
import java.util.Iterator;

public class prac3
{
    public static void main(String[] args)
    {


        ArrayList<AC> l = new ArrayList<>();
        for(int i=0;i<10;i++)
        {
            l.add(new  AC(AC.ID));
            AC.ID++;
        }
        int n;
        final boolean i=true;
        Scanner sc = new Scanner(System.in);

        while(i)
        {



            System.out.println("Enter your ID:-");
            n = sc.nextInt();
            if(n>(AC.ID-1) || n==0)
            {
                System.out.println("Incorrect ID");
                continue;
            }
            else
            {
                AC b = l.get(--n);

                System.out.println("Press 1 for Balance
Inquiry \n Press 2 to Withdraw Money \n Press 3 to
Deposit money \n Press 99 to Exit \n Enter Your Choice :-
");
                int choice = sc.nextInt();
                switch(choice)
                {
                    case 1:
                    b.balanceInquiry();
                    break;
```

```
                    case 2:
                    System.out.println("Enter the amount
you want to Withdraw : ");
                    double money = sc.nextDouble();
                    b.withdrawMoney(money);
                    break;

                    case 3:
                    b.deposit();
                    break;

                    case 99:
                    System.exit(0);

                    default:
                    System.out.println("Invalid
Input..!!");
                }
            }
        }

    }
}
```

```
//prepared by kalp pandya 21CE084
import java.util.*;
class AC
{
    static int ID=1;
    int id;

    AC(int id)
    {
        this.id = id;

    }

    public int getId() {
        return this.id;
    }

    public void setId(int id) {
        this.id = id;
```

```
    }

    public double balance=10000;

    public double getBalance() {
        return this.balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public static void printID() {
        System.out.printf("Your Account ID is
:-  "+"AC"+"%03d\n",ID);
         ID++;
    }

    public void balanceInquiry()
    {
        System.out.println("Balance of account id "+id+"
is:- "+balance);
    }
    public void withdrawMoney(double money)
    {
        if(money>balance)
        {

            System.out.println("Not sufficient Balance");
        }
        else if(balance<=300)
        {

            System.out.println("Soryy..You can not
Withdraw money.. \n Maintain Proper Balance");
        }
        else
        {
            System.out.println("You have withdrawn
"+money+" Rupees");
            balance = balance - money;
            System.out.println("Now you have
"+balance+"  balance left");
        }
    }
    public void deposit()
```

```
    {
        double money;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter amount you want to
deposit:-");
        money = sc.nextDouble();
        balance = balance + money;
        System.out.println("Your balance is:-"+balance);

    }
    public void moneyTransfer(ArrayList<AC> l)
    {
        double money;
        int id;
        Scanner sc = new Scanner(System.in);
        System.out.println("To  which account number do
you want to transfer money?");
        id = sc.nextInt();
        System.out.println("Enter the amount of money you
want to transfer to id "+id);
        money = sc.nextDouble();
        AC a = l.get(--id);
        if(money>=this.balance)
        {
            System.out.println("Can't Transfer Money \n
Insufficient Balance...");
        }
        else if(this.balance<=300)
        {
            System.out.println("Soryy..You can not
transfer money.. \n Maintain Proper Balance");
        }
        else
        {
        a.balance = a.balance + money;
        this.balance = this.balance - money;
        System.out.println(money+" Money has been
Transfered from ID "+this.id+" to ID "+a.id);
        System.out.println("Remaining balance in ID
"+this.id+" is "+this.balance+" Doller");
        System.out.println("\n Balance in ID "+a.id+" is
"+a.balance);
        }

    }
    public void createAccount(ArrayList<AC> a)
```

```
    {
        a.add(new AC(ID));
        AC.printID();
    }
    public void Deactivate(ArrayList<AC> a,int ID)
    {
        a.remove(--ID);
        AC.ID--;
    }
}
```

output

```
Enter your ID:-
10
Press 1 for Balance Inquiry
 Press 2 to Withdraw Money
 Press 3 to Deposit money
 Press 3 to Deposit money
 Press 99 to Exit
 Enter Your Choice :-
3
Enter amount you want to deposit:-
1000
Your balance is:-1100.0
Enter your ID:-
4
Press 1 for Balance Inquiry
 Press 2 to Withdraw Money
 Press 3 to Deposit money
 Press 99 to Exit
 Enter Your Choice :-
99
```

Question 4

Create a class named Stack. Design a class named Queue for storing integers. Like a stack, a queue holds elements. In a stack, the elements are retrieved in a last-in firstout fashion. In a queue, the elements are retrieved in a first-in first-out fashion. The

class contains:

- An int[] data field named elements that stores the int values in the queue.
- A data field named size that stores the number of elements in the queue.
- A constructor that creates a Queue object with default capacity 8.
- The method enqueue(int v) that adds v into the queue.
- The method dequeue() that removes and returns the element from the queue.

	• The method empty() that returns true if the queue is empty. The method getSize() that returns the size of the queue.
code	

```
// 21CE084 kalp pandya

import java.util.*;

class stack {
    static int j = 0;
    int size;
    int s;
    int a[] = null;

    stack() {
        size = 8;
        s = size;
        a = new int[size];
    }

    stack(int size) {
        this.size = size;
        s = size;
        a = new int[size];
    }

    public void enqueue(int v) {
        a[--size] = v;

    }

    public void print() {
        System.out.println(Arrays.toString(a));


    }

    public void dequeue() {
        a = null;

    }

    public boolean empty() {
        if (a == null)
            return true;
        else
            return false;
```

```
    }

    public int getSize() {
        return s;
    }
    // last in first out
}

class queue {
    static int j = 0;
    // first in first out
    int size;
    int a[] = null;

    queue() {
        size = 8;
        a = new int[size];
    }

    queue(int size) {
        this.size = size;
        a = new int[size];
    }

    public void enqueue(int v) {
        a[j++] = v;
        // System.out.println(a[j-1]);
    }

    public void dequeue() {
        a = null;
        // a = new int[8];
    }

    public boolean empty() {
        if (a == null) {

            return true;
        } else
            return false;
    }

    public int getSize() {
        return size;
    }
```

```
    public void print() {
        System.out.println(Arrays.toString(a));

    }
}
```

```
public class stack_que {
    public static void main(String[] args) {

        queue q = new queue();
        q.enqueue(1);
        q.enqueue(3);
        q.enqueue(5);
        q.enqueue(6);
        q.print();
        System.out.println("Size of the queue is :  " +
q.getSize());
        q.dequeue();
        System.out.println(q.empty());
        q = new queue();
        System.out.println(q.empty());

        stack s = new stack();
        s.enqueue(5);
        s.enqueue(2);
        s.enqueue(4);
        s.print();
        System.out.println("Size of the stack is :  " +
s.getSize());
        s.dequeue();
        System.out.println(s.empty());
        s = new stack();
        System.out.println(s.empty());
        System.out.println(" 21CE084 kalp pandya ");
    }
}
```

```
[1, 3, 5, 6, 0, 0, 0, 0]
Size of the queue is :  8
true
false
[0, 0, 0, 0, 0, 4, 2, 5]
Size of the stack is :  8
true
false
 21CE084 kalp pandya
```

Question:5

According to question no 1, the Account class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn

Account.java

```java
// 21CE084 kalp pandya
    import java.util.Date;


public class Account {


    private int id=0;
    private double balance=500;
    private double annualInterestRate=0;
    private Date dateCreated= new Date();


    Account(){}
    public double getAnnualInterestRate() {
        return annualInterestRate;
    }
    public void setAnnualInterestRate(double
annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }
    public int  setDateCreated() {
        return setDateCreated();
    }
    public void setDateCreated(Date dateCreated) {
        this.dateCreated = dateCreated;
```

```
    }
    public double getBalance() {
        return balance;
    }
    public void setBalance(double balance) {
        this.balance = balance;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    Account(int a,double b,double c){
        id=a;
        balance=b;
        annualInterestRate=c;
    }
    Account(Date x){
        dateCreated=x;
    }

    public Date AccDate(){
        return dateCreated;
    }

    public void getMonthlyInterestRate(){

        double m_IR=annualInterestRate;

        System.out.println("Monthly interest rate:
"+m_IR/(12));
    }
```

```
    public void getMonthlyInterest(){
        double m_=annualInterestRate;
        double  b2=balance;
        double m_in=(m_/(100*12));
        b2=b2*m_in;
        System.out.println("Monthly interest :"+ b2);
    }

    public void withdraw(double a){
        double b1=balance;
        b1=b1-a;
        System.out.println("Withdraw amount :"+a);
        System.out.println("After withdraw remaining
balance:"+b1);
    }
   public void  deposit(double b){
    double b2=balance;
       b2=b2+b;
       System.out.println("Deposit amount :"+b);
       System.out.println("After Deposit balance: "+b2);
   }


}
```

```
public class Checking extends Account {

    private double overdraftLimit;


    public Checking() {
        super();
        overdraftLimit = 0;
    }
```

```
    public Checking(int id, double balance,double
overdraftLimit) {
        super(id, balance);
        this.overdraftLimit = overdraftLimit;
    }

    public void setOverdraftLimit(double overdraftLimit) {
        this.overdraftLimit = overdraftLimit;
    }

    public double getOverdraftLimit() {
        return overdraftLimit;
    }


    public void withdraw(double amount) {
        if (getBalance() - amount > overdraftLimit) {
            setBalance(getBalance() - amount);
        }
        else
            System.out.println("Amount exceeds overdraft
limit.");
    }


    public String toString() {
        return super.toString() + "\nOverdraft limit: " +
        String.format("%.2f", overdraftLimit);
    }
}
```

```
public class SavingAccount extends Account {
```

```
    public SavingAccount() {
        super();
    }

    public SavingAccount(int id, double balance) {
        super(id, balance);
    }

    public void withdraw(double amount) {
        double a = amount;
        if (a < getBalance()) {
            setBalance(getBalance() - a);
        } else
            System.out.println("Savings account overdrawn
transtaction rejected");
    }
}
```

test.java

```
public class test {



    public static void main(String[] args) {


        Account account = new Account(2001, 25000);
        SavingAccount savings = new SavingAccount(1002,
20000);
        Checking checking = new Checking(1023, 20000,500);
        //Set annual interest rate of 8%
        account.setAnnualInterestRate(8);
        savings.setAnnualInterestRate(8);
        checking.setAnnualInterestRate(8);
```

```
        // Withdraw 500
        account.withdraw(500);
        savings.withdraw(500);
        checking.withdraw(500);

        // Deposit 2000
        account.deposit(2000);
        savings.deposit(2000);
        checking.deposit(2000);



        // System.out.println(account.toString());
        // System.out.println(savings.toString());
        System.out.println(checking.toString());

    }

}
```

Output

```
Withdraw amount :500.0
After withdraw remaining balance:24500.0
Deposit amount :2000.0
After Deposit balance: 27000.0
Deposit amount :2000.0
After Deposit balance: 21500.0
Deposit amount :2000.0
After Deposit balance: 21500.0
Checking@1be6f5c3
Overdraft limit: 500.00
```

Question:6

Design a class named Triangle that extends GeometricObject.

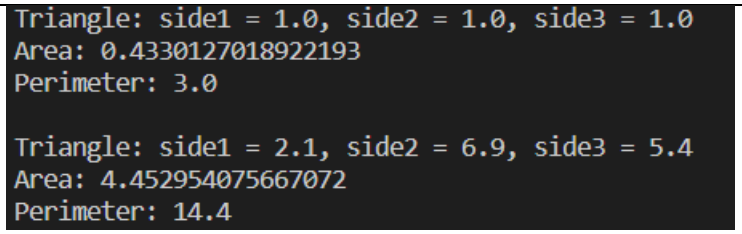
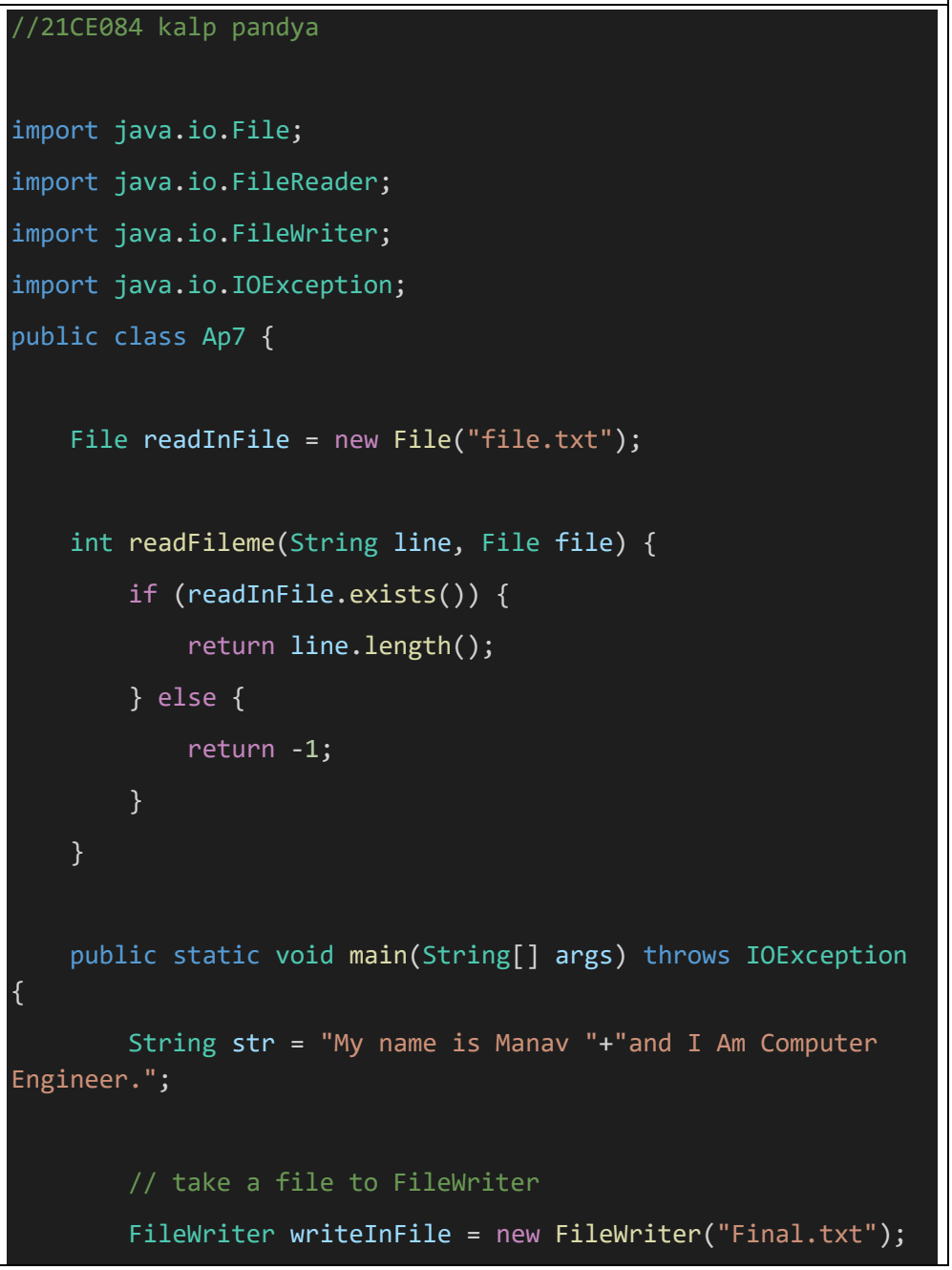
- The class contains: Three double data fields named side1, side2, and side3
- with default values 1.0 to denote three sides of a triangle.
- A no-arg constructor that creates a default triangle.
- A constructor that creates a triangle with the specified side1, side2, and side3.
- The accessor methods for all three data fields.
- A method named getArea() that returns the area of this triangle.

	• A method named getPerimeter() that returns the perimeter of this triangle. • A method named toString() that returns a string description for the triangle. • return "Triangle: side1 = " + side1 + " side2 = " + side2 + " side3 = " + side3
Code:	

```
// 21CE084 kalp pandya

import java.math.*;

public class Triangle  extends GeometricObject{

    private double side1;
    private double side2;
    private double side3;

    public Triangle() {
        side1 = 1.0;
        side2 = 1.0;
        side3 = 1.0;
    }

    public Triangle(double side1, double side2, double side3)
{
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    public double getSide1() {
        return side1;
    }

    public void setSide1(double side1) {
```

```
        this.side1 = side1;
    }

    public double getSide2() {
        return side2;
    }

    public void setSide2(double side2) {
        this.side2 = side2;
    }

    public double getSide3() {
        return side3;
    }

    public void setSide3(double side3) {
        this.side3 = side3;
    }

    @Override
    public double getPerimeter() {
        return (side1+side2+side3);
    }

    @Override
    public double getArea() {
        double s = (side1+side2+side3)/2;
        double area = Math.sqrt(s*(s-side1)*(s-side2)*(s-
side3));
        return area;
    }

    @Override
    public String toString() {
```

```
        return "Triangle: side1 = " + side1 + ", side2 = " +
side2 + ", side3 = " + side3;
    }

    public void print() {
        System.out.println("Area: " + getArea());
        System.out.println("Perimeter: " + getPerimeter());
    }

}




public abstract class GeometricObject {

    public  abstract double getPerimeter();
    public abstract double getArea();


    public static void main(String[] args) {

        Triangle t1 = new Triangle();
        Triangle t2 = new Triangle(2.1, 6.9, 5.4);

        System.out.println(t1.toString());
        t1.print();
        System.out.println();

        System.out.println(t2.toString());
        t2.toString();
        t2.print();
    }
}
```

output	
Question:7	Write a method public static int readInFile(String line, File file) that returns the position number of a line in the file filename or −1 if there is no such line or file. Assume that this file contains names of people with a name per line. Names (and hence lines) are listed in ascending alphabetical order in the file. We can not find the same line twice
Code:	

```java
//21CE084 kalp pandya


import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class Ap7 {


    File readInFile = new File("file.txt");


    int readFileme(String line, File file) {
        if (readInFile.exists()) {
            return line.length();
        } else {
            return -1;
        }
    }


    public static void main(String[] args) throws IOException
{
        String str = "My name is Manav "+"and I Am Computer
Engineer.";


        // take a file to FileWriter
        FileWriter writeInFile = new FileWriter("Final.txt");
```

```
        for (int i = 0; i < str.length(); i++)
            writeInFile.write(str.charAt(i));

        System.out.println("Writting mode open Successfully");
        // close the file while no longer use
        writeInFile.close();



        int ch;
        // check if File exists or not
        FileReader readInFile = new FileReader("Final.txt");
       // System.out.println("File created SucessFully");



        // read from FileReader till the end of file
        while ((ch = readInFile.read()) != -1)
            System.out.print((char) ch);
        // close the file while no longer use
        readInFile.close();
    }
}
```

Output

```
Writting mode open Successfully
My name is Manav and I Am Computer Engineer.
```

Question:8 Write a program that will count the number of characters, words, and lines in a file. Words are separated by whitespace characters. The file name should be passed as a command-line argument.

Code:

```
// 21CE084 kalp pandya


import java.util.Scanner;
import java.io.File;
```

```java
public class count {
    public static void main(String[] args) throws Exception {

        if (args.length < 1) {
            System.out.println("You Have not Given Path for File, Please specify the path");
            System.exit(1);
        }

        File file = new File(args[0]);
        if (!file.exists()) {
            System.out.println("File Does not exist!!");
            System.exit(2);
        }

        Scanner in = new Scanner(file);
        long charCount = 0L;
        int lines = 0;
        int words = 0;

        while(in.hasNext()) {
            String line = in.nextLine();

            String[] wordArray = line.split(" ");

            charCount += line.length();
            lines += 1;
            words += wordArray.length;
        }
        System.out.println("File "+args[0]+" has "+ charCount +" characters " + words + " words " + lines + " lines");
    }
}
```

Output

```
M:\java>java count Final.txt
File Final.txt has 44 characters 9 words 1 lines
```

Question:9	Design an interface named Colorable with a void method named howToColor(). Every class of a colorable object must implement the Colorable interface. Design a class named Square that extends GeometricObject and implements Colorable. Implement howToColor to display the message Color all four sides. The Square class contains a data field side with getter and setter methods, and a constructor for constructing a Square with a specified side. The Square class has a private double data field named side with its getter and setter methods. It has a no-arg constructor to create a Square with side 0, and another constructor that creates a Square with the specified side
Code:	

```java
// 21CE084 kalp pandya

interface Colorable {

    public void howToColor();

}

abstract class GeometricObject {

    public abstract double getPerimeter();

    public abstract double getArea();

}

class Square extends GeometricObject implements Colorable {

    private double side;

    @Override
    public void howToColor() {
        System.out.println("Color all four sides");
    }

    public Square(double side) {
```

```
        this.side = side;
    }

    public double getSide() {
        return side;
    }

    public Square() {
        side = 0;
    }

    public void setSide(double side) {
        this.side = side;
    }

    @Override
    public double getPerimeter() {
        return 4 * side;
    }

    @Override
    public double getArea() {
        return side * side;
    }

    @Override
    public String toString() {
        return "Square: side = " + side;
    }

    public void print() {
        System.out.println("Perimeter of Square: " +
getPerimeter());
        System.out.println("Area of Square: " + getArea());
```

```
    }


}
```

Square_main.java

```
public class square_main {


    public static void main(String[] args) {


        Square s1 = new Square();
        Square s2 = new Square(8.9);


        // Default Square
        System.out.println(s1.toString());
        s1.print();
        s1.howToColor();
        System.out.println();


        System.out.println(s2.toString());
        s2.print();
        s2.howToColor();
    }


}
```

Output

```
Square: side = 0.0
Perimeter of Square: 0.0
Area of Square: 0.0
Color all four sides

Square: side = 5.0
Perimeter of Square: 20.0
Area of Square: 25.0
Color all four sides
```

Question:10 Define a class named ComparableSquare that extends Square and implements Comparable. Implement the compareTo method to compare the Squares on the

	basis of area. Write a test class to find the larger of two instances of ComparableSquareobjects.
Code:	

```
// 21CE084 kalp pandya


interface comparable {


    default public void compare(double side1, double side2) {


        if (side1 == side2) {
            System.out.println();
            System.out.println("The Square and ColorableSquare
is Same...");
        } else {
            System.out.println("The Square and ColorableSquare
is NOT Same...");
        }


    }
}


class squre {


    protected double side1;


    public double gets() {
        return side1;
    }


    public void sets(double side1) {
        this.side1 = side1;
    }


    public void area() {
```

```
        System.out.println("The area of square is : " +
this.side1 * this.side1);
    }
}

class ComparableSquare extends squre implements comparable {
    protected double side2;

    public double gets2() {
        return side2;
    }

    public void sets2(double side2) {
        this.side2 = side2;
    }

    public void area1() {
        System.out.println();

        System.out.println("The area of Colorablesquare is : "
+ this.side2 * this.side2);
    }
}

public class Ap10 {

    public static void main(String[] args) {
        ComparableSquare t = new ComparableSquare();
        t.sets(8);
        t.gets();
        t.area();
        t.sets2(12);
        t.gets2();
        t.area1();
        t.compare(8, 12);
```

```
    }
}
```

Output

```
The area of square is : 64.0

The area of Colorablesquare is : 144.0
The Square and ColorableSquare is NOT Same...
```

Question:11

Create a Triplet class that encapsulates three objects of the same data type in a given instance of Triplet.

Code:

```
// 21CE084 kalp pandya
public class Triplet<S1, S2, S3> {
    private S1 first;
    private S2 second;
    private S3 third;

    Triplet(S1 first, S2 second, S3 third){
        this.first = first;
        this.second = second;
        this.third = third;
    }

    @Override
    public String toString() {
        return "[" + first + "," + second + "," + third +
"]";
    }
    public static void main(String[] args) {
        Triplet<Integer,String,String> t = new
Triplet<Integer,String,String>(Integer.valueOf(84),"kalp"
,"pandya");
        System.out.println(t);
    }
}
```

output

```
[97,kalp,pandya]
```

Question:12

Create an Association class that encapsulates two objects of different types. Similar to Exercise above, create a Transition class that does the same of Association class with three objects.

```
class Bank {

    private String name;
```

```
    Bank(String name) {

        this.name = name;
    }

    public String getBankName() {
        return this.name;
    }
}
```

```
class Employee {
    private String name;

    Employee(String name) {
        this.name = name;
    }

    public String getEmployeeName() {
        return this.name;
    }
}
```

```
public class AssociationExample {
    public static void main(String[] args) {
        Bank bank = new Bank("bob");
        Employee emp = new Employee("kalp pandya");
        System.out.println(emp.getEmployeeName() + " is
employee of " + bank.getBankName());
        System.out.println("prepared by kalp pandya
21ce084");
    }
}
```

Output 1

```
kalp pandya is employee of bob
prepared by kalp pandya 21ce084
```

```
class Branch{
    private String name;

    Branch(String name){
        this.name = name;
```

```
    }

    public String getBranceName() {
        return this.name;
    }
}
```

```
public class TransitionExample {
    public static void main(String[] args) {
        Bank bank = new Bank("BOB");
        Employee emp = new Employee("kalp pandya");
        Branch brc = new Branch("vaodadara");
        System.out.println(emp.getEmployeeName() + " is
employee of " + bank.getBankName() + " in " +
brc.getBranceName() + " branch");
    System.out.println("prepared by kalp pandya
21CE084");
    }
}
```

Output 2

```
kalp pandya is employee of BOB in vaodadara branch
prepared by kalp pandya 21CE084
```

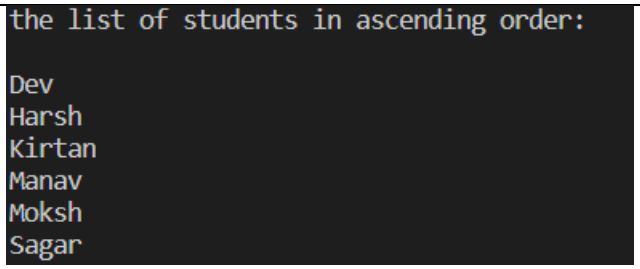
Question:13

(Display nonduplicate names in ascending order) Given one or more text files, each representing a day's attendance in a course and containing the namesof the students who attended the course on that particular day, write a program that displays, in ascending order, the names of those students who have attended at least one day of the course. The text file(s) is/are passed as command-line argument

Code

```
//21CE084 kalp pandya
import java.util.*;
import java.io.*;

public class Prac_Ass_13 {

    public static void sortArray(String[] student, int k)
{
        for (int i = 0; i < k - 1; i++) {
            for (int j = 0; j < k - i - 1; j++) {
                if (student[j].compareTo(student[j + 1])
> 0) {
                    String temp = student[j];
                    student[j] = student[j + 1];
```

```
                    student[j + 1] = temp;
                }
            }
        }
        for (int i = 0; i < k; i++) {
            System.out.println(student[i]);
        }
    }

    public static void main(String[] args) {
    String student[] = new String[500];
        int counter = 0;
        try {
            for (int i = 0; i < args.length; i++) {
                File file = new File(args[i]);
                Scanner scnr = new Scanner(file);
                while (scnr.hasNextLine()) {
                    String line = scnr.nextLine();
                    int flag = 0;
                    for (int j = 0; j < counter; j++) {
                        if (line.compareTo(student[j]) ==
0) {
                            flag = 1;
                            break;
                        }
                    }
                    if (flag == 0) {
                        student[counter] = line;
                        counter++;
                    }
                }
            }
            System.out.println("the list of students in
ascending order:");
            sortArray(student, counter);
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

Output	 the list of students in ascending order: Dev Harsh Kirtan Manav Moksh Sagar