CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

cspit
Chandubhai S Patel Institute of Technology

Faculty of Technology and Engineering

Department of Computer Science & Engineering

Date :10/10/ 2022

Academic Year	:	2022-23	Semester	:	3
Course code	:	CE251	Course name	:	Java Programming

Practical Assignment

Github link:

Question :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:	

```
import java.util.*;

class Account {
    static public int id;
    static public double balance;
    final static private double annualInterestRate = 7;//keeping
intrest rate constant
    static public String dateCreated;

    public Account() {
        id = 0;
        balance = 500;
        dateCreated = "06/11/2003";//construtor to making an
default account
    }

    static Scanner s = new Scanner(System.in);

    public Account(int Ac, double bal, String d) {
        id = Ac;
        balance = bal;
        dateCreated = d;//construtor to making an user definrd
account
    }

    public void Accessor() {
        System.out.println("Your Account  :" + id);
        System.out.println("Total balance in your account is  :" +
balance + " Rupees");
        System.out.println("The intrest given by the bank is  :" +
annualInterestRate);
        System.out.println("The at which your account was created
is  :" + dateCreated);//method for printing the account
    }

    public void mutator(int ac, double bal, String d) {
        id = ac;
        balance = bal;
        dateCreated = d;//method for using different account
    }

    public double getMonthlyInterestRate() {
        return annualInterestRate / 12;//method returning monthly
intrest rate
    }

    public double getMonthlyInterest() {
```

```
        return (annualInterestRate / 12) * balance / 100;//method
returing monthly intrest ruppee
    }

    public void withdraw(double draw) {
        balance =  balance - draw;//method editing balance after
withdrawing
    }

    public void deposit(double dep) {
        balance = balance + dep;//method editing balance after
depositing
    }

    @Override//overriding to string method for printing account
details.
    public String toString() {
        String res = "";
        res += "Account number : " + id + "\n";
        res += "Balance in account is : " + balance + "\n";
        res += "Annual Interest Rate given by bank is : " +
annualInterestRate + "\n";
        res += "Date of creation of account is : " + dateCreated +
"\n";
        return res;
    }
}




// Name :- Aswani Darsh
// • 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 7%). 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.
```

```java
// • 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.
// Roll-no :-21ce006
// Aim :-Design a class named Account that contains:
// • A private int data field named id for the account (default
0).
// Git-hub repository: https://github.com/006Darsh/java-
Assaignment-2
package Darsh2_2;
import java.util.*;

public class Darsh2_2main {
    public static void main(String[] args) {
        System.out.println("An example for you to to create a
proper account :");
        Account d2_1 =new Account();//calling and printing default
constructor
        d2_1.Accessor();
        Scanner s = new Scanner(System.in);

        int id;
        double balance,withdraw,deposit,monintrate,monint;
        String date;
        System.out.println("Enter the Account number of your
account  :");
        id = s.nextInt();
        System.out.println("Enter the initial balance your
account  :");
        balance = s.nextDouble();
        System.out.println("Enter the date at which you created
your account  :");
        date = s.next();

        Account d2_2 = new Account(id, balance, date);
        d2_2.Accessor();//calling and printing parameterized
construstor
        monintrate = d2_2.getMonthlyInterestRate();//getting
monthly intrest rate and rupees
        monint = d2_2.getMonthlyInterest();
        System.out.println("Bank give "+monintrate+"% monthly
intrest rate.");//printing monthly intrest rate and rupees
```

```
        System.out.println("Your monthly intrest is "+monint+"
Rupees");
        int i;

        System.out.println("Enter 1 to withdraw and 2 to
deposit.");
        i = s.nextInt();//giving choice to the costomer to
withdraw or to deposit the money.
        switch(i)
        {
            case 1 :
            {
                System.out.println("Enter amount to be
withdrawn  :");
                withdraw = s.nextDouble();
                d2_2.withdraw(withdraw);//withdrawing withdraw
ammount of money from account
                System.out.println("The amount remained in your
account after withdrawal is   :"+d2_2.balance);
                break;
            }
            case 2 :
            {
                System.out.println("Enter amount to be
deposited  :");
                deposit = s.nextDouble();
                d2_2.deposit(deposit);//depositing deposit ammount
of money to account
                System.out.println("The amount remained in your
account after deposit is   :"+d2_2.balance);
                break;
            }
            default :
            {
                System.out.println("You have changed
anything  :");
                break;
            }
        }
        System.out.println("Your account afer withdrawal or
deposit  :");
        d2_2.Accessor();//printing account details after
withdrawing or depositing.

        int p=1;
        while(p==1)
        {
```

```
            System.out.println("Enter 1 use another account and 2
to not.");//continuing it again and again until account holder
gives input 2
            i = s.nextInt();//giving choice to account holder to
change the account or not
            if(i==1)//if account holder changes the account the
repeating above procedure again
            {
                System.out.println("Enter the Account number of
your account  :");
                id = s.nextInt();
                System.out.println("Enter the initial balance your
account  :");
                balance = s.nextDouble();
                System.out.println("Enter the date at which you
created your account  :");
                date = s.next();

                d2_2.mutator(id, balance, date);
                d2_2.Accessor();
                monintrate = d2_2.getMonthlyInterestRate();
                monint = d2_2.getMonthlyInterest();
                System.out.println("Bank give "+monintrate+"%
monthly intrest rate.");
                System.out.println("Your monthly intrest is
"+monint+" Rupees");
                System.out.println("Enter 1 to withdraw and 2 to
deposit.");
                i = s.nextInt();

                switch(i)
                {
                    case 1 :
                    {
                        System.out.println("Enter amount to be
withdrawn  :");
                        withdraw = s.nextDouble();
                        d2_2.withdraw(withdraw);
                        System.out.println("The amount remained in
your account after withdrawal is   :"+d2_2.balance);
                        break;
                    }
                    case 2 :
                    {
                        System.out.println("Enter amount to be
deposited  :");
                        deposit = s.nextDouble();
                        d2_2.deposit(deposit);
```

```
                        System.out.println("The amount remained in
your account after deposit is   :"+d2_2.balance);
                        break;
                    }
                    default :
                    {
                        System.out.println("You have changed
anything  :");
                        break;
                    }
                }
            }
            else
            {
                System.out.println("--------------thanks for
coming---------------------");
                break;
            }
        }
    }
}
```

Output:

```
PS E:\Darsh\java\Darsh2_2> cd "e:\Darsh\java\Darsh2_2\" ; if ($?) { javac Darsh2_2main.java } ; if ($?) { java Darsh
An example for you to to create a proper account :
Your Account  :0
Total balance in your account is  :500.0 Rupees
The intrest given by the bank is  :7.0
The at which your account was created is  :06/11/2003
Enter the Account number of your account  :
111
Enter the initial balance your account  :
1500.265
Enter the date at which you created your account  :
06-11-2003
Your Account  :111
Total balance in your account is  :1500.265 Rupees
The intrest given by the bank is  :7.0
The at which your account was created is  :06-11-2003
Bank give 0.5833333333333334% monthly intrest rate.
Your monthly intrest is 8.751545833333335 Rupees
Enter 1 to withdraw and 2 to deposit.
2
Enter amount to be deposited  :
1500
The amount remained in your account after deposit is   :3000.2650000000003
Your account afer withdrawal or deposit  :
Your Account  :111
Total balance in your account is  :3000.2650000000003 Rupees
The intrest given by the bank is  :7.0
The at which your account was created is  :06-11-2003
Enter 1 use another account and 2 to not.
2
--------------thanks for coming---------------------
```

Github Link:

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.

<table>
<tr><td></td><td>

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:

</td></tr>
<tr><td>Code:</td><td>RegularPolygon.java
<pre>
// Name :- Aswani Darsh
// Roll no :-21ce006
/*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:
 */
import java.math.*;
</pre>
</td></tr>
</table>

```java
public class RegularPolygon {

    static double pi = 3.14;
    private int nos;
    private double sides;
    private double a;
    private double b;

    public RegularPolygon(){
        nos = 3;
        sides = 1;
        a = 0;
        b = 0;
    }

    public int getN() {
        return nos;
    }

    public void setN(int nos) {
        this.nos = nos;
    }

    public double getSide() {
        return sides;
    }

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

    public double getX() {
        return a;
    }

    public void setX(double x) {
        this.a = x;
    }

    public double getY() {
        return b;
    }

    public void setY(double y) {
        this.b = y;
    }

    public RegularPolygon(int nos, double sides){
```

```
        this.nos = nos;
        this.sides = sides;
        a = 0;
        b = 0;
    }

    public RegularPolygon(int nos, double sides, double x,
double y){
        this.nos = nos;
        this.sides = sides;
        this.a = x;
        this.b = y;
    }

    public double getPerimeter() {
        return nos*sides;
    }

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

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

RegularPolygon_main.java

```
public class RegularPolygon_main {

    public static void main(String[] args) {

        RegularPolygon p1 = new RegularPolygon();
        RegularPolygon p2 = new RegularPolygon(5, 120);
        RegularPolygon p3 = new RegularPolygon(10, 600, 60,
20);

        System.out.println("Polygon 1 Default");
        p1.print();
```

```
        System.out.println("Polygon 2 without coordinate");
        p2.print();
        System.out.println("Polygon 3 with coordinate");
        p3.print();
        System.out.println("This Program is created By
Aswani Drash 21CE006");

    }

}
```

Output:

```
Polygon 1 Default
No. of sides : 3
Length of sides : 1.0
Perimeter of Polygon : 3.0
Area of Polygon : 0.43354374924141237

Polygon 2 without coordinate
No. of sides : 5
Length of sides : 120.0
Perimeter of Polygon : 600.0
Area of Polygon : 24791.477199448906

Polygon 3 with coordinate
No. of sides : 10
Length of sides : 600.0
Perimeter of Polygon : 6000.0
Area of Polygon : 2771416.9832481244

This Program is created By Aswani Drash 21CE006
PS E:\Darsh\java\Practical Assignment>
```

Github Link:

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:

```
package Darsh2_3;
public class Atm {
    private static int count;
    private final String id;
    private double balance;

    public String getId() {
        return id;//returns account no.
    }

    public double getBalance() {
        return balance;//returns balance of account after using
atm
    }

    public Atm() {
        count++;
        if (count < 10) {
            id = "AC00" + (count);
        } else {
            id = "AC0" + (count);
        }
        balance = 300;
    }//construxtor to create account no and giving all account of
minimal balance of 300 rs.

    public void withdraw(double money) {
        if (balance - money >= 300) {
            balance -= money;
            System.out.println(money + " Rs successfully
withdrawn.");
            System.out.println("Remaining Balance is : " +
balance);
        } else {
            System.out.println("Insufficient balance to withdraw
the amount.");
        }
    }//function checking if the ammount to withdrawn keeps the
minimal ammount in account of 300 rs or not
```

```
    //if yes the reduces the balance balance of the account

    public void deposit(double amount) {
        balance += amount;
        System.out.println(amount + "Rs deposited to your
account.");
        System.out.println("Current Balance is : " + balance);
    }//add the ammount to be deposited in the account balance

    public void MoneyTransfer(Atm obj, double amount) {
        if (balance - amount >= 300) {
            balance -= amount;
            obj.balance += amount;
            System.out.println(amount + " Rs successfully
Transferred.");
            System.out.println("Remaining Balance is : " +
balance);
        } //function checking if the ammount to withdrawn from
account from which money is to be transfered keeps the minimal
ammount in account of 300 rs or not
        //if yes the reduces the balance balance of the account

        else {
            System.out.println("Insufficient balance to transfer
the amount.");
        }
    }
```

→Main file

```
// Name :- Aswani Darsh
// Roll-no :-21ce006
// Aim :-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.
package Darsh2_3;

import java.util.*;

    public class Darsh2_3main
```

```
    {
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            String id = "";
            String id2 = "";
            boolean flag = true;
            int choice;
            double amt;
            ArrayList<Atm> people = new
ArrayList<Atm>();//creating an arraylist named people and using it
to acccess the class Atm's constructor creating the default
account
            for (int i = 1; i <= 10; i++) {
                people.add(new Atm());//creates 10 account
            }
            System.out.print("Enter Your Account Number : ");
            id = sc.next();
            int userNumber = userID(id, people);

            while (flag) {//asking the tasks of the Atm to user
until user exits.
                System.out.println();
                System.out.println("Make a choice.....");
                System.out.println("1.Balance inquiry ");
                System.out.println("2.Withdraw money ");
                System.out.println("3.Deposit money");
                System.out.println("4.Money Transfer ");
                System.out.println("5.Create Account ");
                System.out.println("6.Deactivate Account");
                System.out.println("7.Exit ");
                choice = sc.nextInt();//asking user to make choice
for using functions of the atm
                switch (choice) {//accordingly using the metods
created in the ATm class.
                    case 1 -> {
                        System.out.println("Account Number : " +
id);
                        System.out.println("Current Balance : " +
people.get(userNumber).getBalance());
                    }
                    case 2 -> {
                        System.out.print("Enter Amount To Withdraw
: ");
                        amt = sc.nextDouble();
                        people.get(userNumber).withdraw(amt);
                    }
                    case 3 -> {
```

```
                        System.out.print("Enter Amount To Deposit
: ");
                        amt = sc.nextInt();
                        people.get(userNumber).deposit(amt);
                    }
                    case 4 -> {
                        System.out.print("Enter Account Number To
Transfer Money :");
                        id2 = sc.next();
                        int u2 = userID(id2, people);
                        System.out.print("Enter Amount To Transfer
: ");
                        amt = sc.nextInt();
                        people.get(userNumber).MoneyTransfer(peopl
e.get(u2), amt);
                    }
                    case 5 -> {
                        people.add(new Atm());
                        System.out.println("Account Created
Successfully.");
                        System.out.println("The New Account Number
Is :" + people.get(people.size() - 1).getId());
                    }
                    case 6 -> {
                        people.remove(userNumber);
                        System.out.println("Account Deleted
Successfully.");
                        flag = false;
                    }
                    case 7 ->{
                        flag = false;
                        System.out.println("----------------------
-----Thank you------------------------------");
                    }
                    default -> System.out.println("Make a valid
choice..");
                }
            }

        }

        public static int userID(String id, ArrayList<Atm>people)
{//checks if the entered userId exists or not if yes the return
the an number which assigned to a specific entered

 //userId and helps to use the account accordingly.
            Scanner s = new Scanner(System.in);
            int user = 10000;
```

```
            int i;
            for (i = 0; i < people.size(); i++) {
                if (id.equals(people.get(i).getId())) {
                    user = i;
                    break;
                }
            }
            if (i == people.size()) {
                System.out.println("No Such Account Exists.\nTry
Again..");
                System.out.print("Enter your account id :");
                id = s.next();
                return userID(id, people);
            }
            else
            return user;
        }
    }
```

Output:	

```
Enter Your Account Number (ex:AC00x): AC005

Make a choice......
1.Balance inquiry
2.Withdraw money
3.Deposit money
99.Exit
1
Account Number : AC005
Current Balance : 300.0

Make a choice......
1.Balance inquiry
2.Withdraw money
3.Deposit money
99.Exit
2
Enter Amount To Withdraw : 150
Insufficient balance to withdraw the amount.

Make a choice......
1.Balance inquiry
2.Withdraw money
3.Deposit money
99.Exit
3
Enter Amount To Deposit : 780
780.0Rs deposited to your account.
Current Balance is : 1080.0

Make a choice......
1.Balance inquiry
2.Withdraw money
3.Deposit money
99.Exit
99
----------------------------Thank you-------------------------------
```

Github Link:	

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:	

```java
//→Stack.java
// package Practical Assignment;

import java.util.*;

public 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;
        // System.out.println(a[j-1]);
    }
    public void print()
```

```
    {
        System.out.println(Arrays.toString(a));
        // if(a!=null)
        // {
            // a = null;

        // }
    }
    public void dequeue()
    {
        a = null;
        // a = new int[8];
    }
    public boolean empty()
    {
        if(a==null)
        return true;
        else
        return false;
    }
    public int getSize()
    {
        return s;
    }
    //last in first out
}

//→Queue.java

public 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));

    }
}

//→Main file
// package Practical Assignment;
//This program is created by Aswani Darsh 21CE006
//Github link:-
/*Aim:-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.
*/
import java.util.*;

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

        Queue q = new Queue();
        q.enqueue(1);
        q.enqueue(2);
        q.enqueue(3);
        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(1);
        s.enqueue(2);
        s.enqueue(3);
        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("This program is created by Aswani
Darsh 21CE006");
    }

}
```

Output:

```
[150, 100, 50, 0, 0, 0, 0, 0]
Size of the queue is :  8
true
false
[0, 0, 0, 0, 0, 150, 100, 50]
Size of the stack is :  8
true
false
This program is created by Aswani Darsh 21CE006
```

Github Link:

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
Code:	→Account.java

```
import java.util.*;

class Account {
    static public int id;
    static public double balance;
    final static private double annualInterestRate = 7;//keeping
intrest rate constant
    static public String dateCreated;

    public Account() {
        id = 0;
        balance = 500;
        dateCreated = "06/11/2003";//construtor to making an
default account
    }

    static Scanner s = new Scanner(System.in);

    public Account(int Ac, double bal, String d) {
        id = Ac;
        balance = bal;
        dateCreated = d;//construtor to making an user definrd
account
    }

    public void Accessor() {
        System.out.println("Your Account  :" + id);
        System.out.println("Total balance in your account is  :" +
balance + " Rupees");
        System.out.println("The intrest given by the bank is  :" +
annualInterestRate);
        System.out.println("The at which your account was created
is  :" + dateCreated);//method for printing the account
    }

    public void mutator(int ac, double bal, String d) {
```

```
        id = ac;
        balance = bal;
        dateCreated = d;//method for using different account
    }

    public double getMonthlyInterestRate() {
        return annualInterestRate / 12;//method returning monthly
intrest rate
    }

    public double getMonthlyInterest() {
        return (annualInterestRate / 12) * balance / 100;//method
returing monthly intrest ruppee
    }

    public void withdraw(double draw) {
        balance =  balance - draw;//method editing balance after
withdrawing
    }

    public void deposit(double dep) {
        balance = balance + dep;//method editing balance after
depositing
    }

    @Override//overriding to string method for printing account
details.
    public String toString() {
        String res = "";
        res += "Account number : " + id + "\n";
        res += "Balance in account is : " + balance + "\n";
        res += "Annual Interest Rate given by bank is : " +
annualInterestRate + "\n";
        res += "Date of creation of account is : " + dateCreated +
"\n";
        return res;
    }
}
```

→checking accont class

```
public class CheckingAccount extends Account {
    protected double OVERDRAFT_LIMIT = -100;//putting a limit for
overdrafting

    public CheckingAccount(int id, double balance,String date) {
        super(id, balance,date);//creating a checking account
    }
```

```
    @Override
    public void withdraw(double amount) {//withdrawing and
overdrafting money from checking account.
        if (balance - amount >= OVERDRAFT_LIMIT) {//if the
overdrafting limit is passsed the no money is withdrawn if not the
money is allowed to be withdrawn.
            super.withdraw(amount);
        }
        else
        System.out.println("Over drawing is passing the given
limit  :");
    }

    @Override
    public String toString() {
        return "CheckingAccount{"+ "mBalance=" + balance
+'}';//overriding the to string method for printing the checking
account balance
    }
}
```

→saving account class

```
public class SavingsAccount extends Account {
    protected double OVERDRAFT_LIMIT = 0;

    public SavingsAccount(int id, double balance,String date) {
        super(id, balance, date);//creates the saving account
    }

    @Override
    public void withdraw(double amount) {//as their is no
overdraft limit so method to withdraw any ammount of money.
        if (balance - amount >= OVERDRAFT_LIMIT || balance -
amount <= OVERDRAFT_LIMIT)
            super.withdraw(amount);
        }


    @Override
    public String toString() {
```

```
        return "SavingsAccount{" + "net Balance =" + balance
+'}';//overriding the to string method for printing the savings
account balance
    }
}
```

Main file

```
// Name :- Aswani Darsh
// Roll-no :-21ce006
// Aim :-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
import java.util.Scanner;

import Darsh2_2.*;
public class Darsh2_4main {
    public static void main(String[] args) {
        Account account = new Account(111, 200,"06-11-2003");
        System.out.println("simple account");//creates a simple
account
        System.out.println(account);//calls the account classes to
string overide
        System.out.println("----------------------------------------
-----------------------------------------------");
        CheckingAccount checkingAccount = new CheckingAccount(112,
250,"05-11-2003");
        System.out.println("Checking account");//creates a
checking account
        System.out.println(account);//calls the account classes to
string overide
        System.out.println("Enter ammount for withdrawing in
checking account  :");
        Scanner s = new Scanner(System.in);
        double ammount = s.nextDouble();
        checkingAccount.withdraw(ammount);
        System.out.println(checkingAccount);//call the checking
account to string override
        System.out.println(account);
        SavingsAccount savingsAccount = new SavingsAccount(113,
10000,"04-11-2003");
```

```
        System.out.println("----------------------------------------
------------------------------------------------");
        System.out.println("Saving account");//creates a checking
account
        System.out.println(account);//calls the account classes to
string overide
        System.out.println("Enter ammount for withdrawing in
saving account  :");
        double ammounts = s.nextDouble();
        savingsAccount.withdraw(ammounts);

        System.out.println(savingsAccount);//call the checking
account to string override
        System.out.println(account);
    }
}
```

Output:

```
simple account
Account number : 111
Balance in account is : 200.0
Annual Interest Rate given by bank is : 7.0
Date of creation of account is : 06-11-2003

--------------------------------------------------------------------------------
Checking account
Account number : 112
Balance in account is : 250.0
Annual Interest Rate given by bank is : 7.0
Date of creation of account is : 05-11-2003

Enter ammount for withdrawing in checking account  :
400
Over drawing is passing the given limit  :
CheckingAccount{mBalance=250.0}
--------------------------------------------------------------------------------
Saving account
Account number : 113
Balance in account is : 10000.0
Annual Interest Rate given by bank is : 7.0
Date of creation of account is : 04-11-2003

Enter ammount for withdrawing in saving account  :
400
SavingsAccount{net Balance =9600.0}
```

Github Link:

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:	Triangle.java

```java
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;
    }



    @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());
    }

}
```

//→GeometericObject.java

```
public abstract class GeometricObject {

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

}
```

//→triangle_main.java

```
//This Program Is Created By Aswani Darsh 21CE006
//https:
/*AiM: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
*/
```

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

        Triangle t1 = new Triangle();
        Triangle t2 = new Triangle(25.256, 68.546, 86.46);

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

        System.out.println(t2.toString());
        t2.toString();
        t2.print();
        System.out.println("This Program Is Created By Aswani Darsh 21CE006");
    }

}
```

Output:

```
Triangle: side1 = 1.0, side2 = 1.0, side3 = 1.0
Area: 0.4330127018922193
Perimeter: 3.0

Triangle: side1 = 25.256, side2 = 68.546, side3 = 86.46
Area: 680.6815810830825
Perimeter: 180.262
This Program Is Created By Aswani Darsh 21CE006
```

Github Link:

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
//This program is created by Aswani Darsh 21CE006
/*Aim:-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*/
//Github link:-
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
class file1 {
    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 = "Hello everyone I am Aswani Darsh Hemrajbhai
" +
                " And I am a Computer Engineer Studying at CSPIT
";

        // take a file to FileWriter
        FileWriter writeInFile = new FileWriter("XYZ.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("XYZ.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);
            System.out.println("\nThis program is created by
Aswani Darsh 21CE006");
        // close the file while no longer use
```

```
        readInFile.close();
    }
}
```

Output:

```
Writting mode open Successfully
Hello everyone I am Aswani Darsh Hemrajbhai  And I am a Computer Engineer Studyin
This program is created by Aswani Darsh 21CE006
```

Practical Assignment > XYZ.txt

```
1   Hello everyone I am Aswani Darsh Hemrajbhai  And I am a Computer Engineer Studying at CSPIT
```

Github Link:

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:

```java
//This Program Is Created Aswani Darsh 21CE006
//Github Link:-
//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.
import java.util.Scanner;
import java.io.File;

public class charWordCounter {
    public static void main(String[] args) throws Exception {
      //  if in Commandline Argument we didn't give file it will not execute this program
        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);
        }
    //here we initialized all value zero
        Scanner in = new Scanner(file);
        long charCount = 0L;
        int lines = 0;
        int words = 0;

        while(in.hasNext()) {
            String line = in.nextLine();
            //here we Applied logic for counting the lines,words etc....
            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");
```

```
        System.out.println("This Program Is Created Aswani Darsh
21CE006");
    }
}
```

Output:

```
PS E:\Darsh\java\Practical Assignment> javac charWordCounter.java
PS E:\Darsh\java\Practical Assignment> java charWordCounter file.txt
File file.txt has 44 characters 6 words 1 lines
This Program Is Created Aswani Darsh 21CE006
```

```
ractical Assignment > file.txt
 1    Hello Everyone I am Aswani Darsh!.....
```

Github Link:

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
Software Requirement:	VS CODE
Code:	

```java
public 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());
    }

}
```

```
public abstract class GeometricObject {

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

}


interface colorable {
    public void howToColor();
}
```

```
//This Program is created By Aswani Darsh 21CE006
//Github Link:-
/*AIM: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 */
```

```java
public class Square_main {

    public static void main(String[] args) {

        Square s1 = new Square();
        Square s2 = new Square(45.4632);

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

        System.out.println(s2.toString());
        s2.print();
        s2.howToColor();
        System.out.println("This Program is created By Aswani darsh 21CE006");
    }
}
```

Output:

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

Square: side = 45.4632
Perimeter of Square: 181.8528
Area of Square: 2066.90255424
Color all four sides
This Program is created By Aswani darsh 21CE006
```

Github Link:

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:	

```java
//This program is created by Aswani Darsh 21CE006
//Github link:-
// //Aim:- 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.

interface Comparable{
    public void CompareTo(double Area1,double Area2);
}
class Square{
    private double s1;
    double Area = 0.0;
    Square(){
        s1 = 5.0;
    }
    Square(double s1){
        this.s1 = s1;
    }
    public double getArea(double s1){
        return s1*s1;
    }
    public void setside(double s1){
        this.s1 = s1;
    }
    public double getSide(){
        return s1;
    }
}
class comparableSquare extends Square implements Comparable{

    @Override
    public void CompareTo(double Area1,double Area2){
        if(Area1 == Area2){
            System.out.println("Both squares are same");
        }
        else{
            System.out.println("Both squares are different");
        }

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

```
        comparableSquare c1 = new comparableSquare();
        comparableSquare c2 = new comparableSquare();
        c1.setside(10);
        c2.setside(10.1);
        System.out.println("side of square 1:"+c1.getSide());
        System.out.println("side of square 2:"+c2.getSide());
        System.out.println("area of square
1:"+c1.getArea(c1.getSide()));
        System.out.println("area of square
2:"+c2.getArea(c2.getSide()));
        c1.CompareTo(c1.getArea(c1.getSide()),c2.getArea(c2.getS
ide()));
        System.out.println("This program is created by Darsh
Aswani 21CE006");
    }
}
```

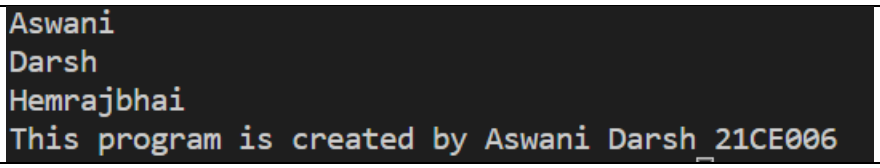
Output:

```
side of square 1:10.0
side of square 2:10.1
area of square 1:100.0
area of square 2:102.00999999999999
Both squares are different
This program is created by Darsh Aswani 21CE006
```

Github Link:

Question:11	Create a Triplet class that encapsulates three objects of the same data type in a given instance of Triplet.
Code:	

```java
//This program is created by Aswani Darsh 21CE006
//Github link:-
// //Aim:- Create a Triplet class that encapsulates three objects of the same data type in a given instance of Triplet.
class Triplet<T> {
    private T obj1;
    private T obj2;
    private T obj3;

    public Triplet(T obj1, T obj2, T obj3) {
        this.obj1 = obj1;
        this.obj2 = obj2;
        this.obj3 = obj3;
    }

    public T getObj1() {
        return obj1;
    }

    public T getObj2() {
        return obj2;
    }

    public T getObj3() {
        return obj3;
    }

}
public class Triplet_Main {
    public static void main(String[] args) {
        Triplet<String> triplet = new Triplet<>("Aswani ", "Darsh ", "Hemrajbhai");
        System.out.println(triplet.getObj1());
        System.out.println(triplet.getObj2());
        System.out.println(triplet.getObj3());
        System.out.println("This program is created by Aswani Darsh 21CE006");
    }
}
```

Output:	
Github Link:	

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.
Code:	

```
//This program is created by Aswani Darsh 21CE006
//Github link:-
// //Aim:- 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 Association<T1, T2> {
    T1 object1;
    T2 object2;

    public Association(T1 object1, T2 object2) {
      this.object1 = object1;
      this.object2 = object2;
    }
  }

  class Transition<T1, T2, T3> {
    T1 object1;
    T2 object2;
    T3 object3;

    public Transition(T1 object1, T2 object2, T3 object3) {
      this.object1 = object1;
      this.object2 = object2;
      this.object3 = object3;
    }
  }

  public class Asso_Tran_main {
    public static void main(String[] args) {
      Association<String, Integer> asso1 = new Association<String, Integer>("One", 1);
      Association<String, Integer> asso2 = new Association<String, Integer>("Two", 2);
      Association<String, Integer> asso3 = new Association<String, Integer>("Three", 3);

      System.out.println(asso1.object1 + " " + asso1.object2);
      System.out.println(asso2.object1 + " " + asso2.object2);
      System.out.println(asso3.object1 + " " + asso3.object2);
```

```
      Transition<String, Integer, String> tran1 = new
Transition<String, Integer, String>("One", 1, "One");
      Transition<String, Integer, String> tran2 = new
Transition<String, Integer, String>("Two", 2, "Two");
      Transition<String, Integer, String> tran3 = new
Transition<String, Integer, String>("Three", 3, "Three");

      System.out.println(tran1.object1 + " " + tran1.object2
+ " " + tran1.object3);
      System.out.println(tran2.object1 + " " + tran2.object2
+ " " + tran2.object3);
      System.out.println(tran3.object1 + " " + tran3.object2
+ " " + tran3.object3);
    }
  }
```

Output:

```
One 1
Two 2
Three 3
One 1 One
Two 2 Two
Three 3 Three
```

Github Link:

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
Software Requirement:	VS CODE
Code:	

```
//This program is created by Aswani Darsh 21CE006
//Github link:-
/*Aim:-(Display nonduplicate names in ascending order) Given
one or more text files, each
representing a day's attendance in a course and containing
the names of 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(s */
import java.io.File;
import java.io.IOException;
import java.util.*;

public class Stud_Attendence {
    // function to sort the array of students
    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;
                }
            }
        }
        // print the student array in ascending order
        for (int i = 0; i < k; i++) {
            System.out.println(student[i]);
        }
    }

    public static void main(String[] args) {
        // an array to store the names
```

```
        String student[] = new String[500];
        int counter = 0;
        try {
            // Here we use the loop for reading the names
            for (int i = 0; i < args.length; i++) {
                File file1 = new File(args[i]);
                Scanner scnr = new Scanner(file1);
                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:");
            // call the sortArray() to sort the student
array and print it
            sortArray(student, counter);
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        System.out.println("This program is created by
Aswani darsh 21CE006");
    }
}
```

Output:

```
PS E:\Darsh\java\Practical Assignment> javac Stud_Attendence.java
PS E:\Darsh\java\Practical Assignment> java Stud_Attendence file1.txt
the list of students in ascending order:
abhi
darsh
krunal
prince
priyanshu
yashu
This program is created by Aswani darsh 21CE006
```

	Practical Assignment > file1.txt 1 darsh 2 prince 3 yashu 4 krunal 5 abhi 6 priyanshu 7	
Github Link:		