SDM College of Engineering and Technology

Dhavalagiri, Dharwad-580 002. Karnataka State. India.

Email: principal@sdmcet.ac.in, cse.sdmcet@gmail.com

Ph: 0836-2447465/ 2448327 Fax: 0836-2464638 Website: sdmcet.ac.in

Department

of

COMPUTER SCIENCE AND ENGINEERING

LABORATORY REPORT

[18UCSL405- OBJECT ORIENTED PROGRAMMING LABORATORY]

Even Semester: Jan-June-2022

Course Teacher: Dr. U.P.Kulkarni and Prof. R.G.Yadwad

2021- 2022

Submitted by

By

Mr. Meeth Sakaria

2SD20CS054

4th Semester A division

Table of Contents

Termwork-1: Prepare the requirement specifications from the given class diagram and write a Java program to implement the given design.

Problem Statement:

Solve the given UML diagram and implement it.

Theory:

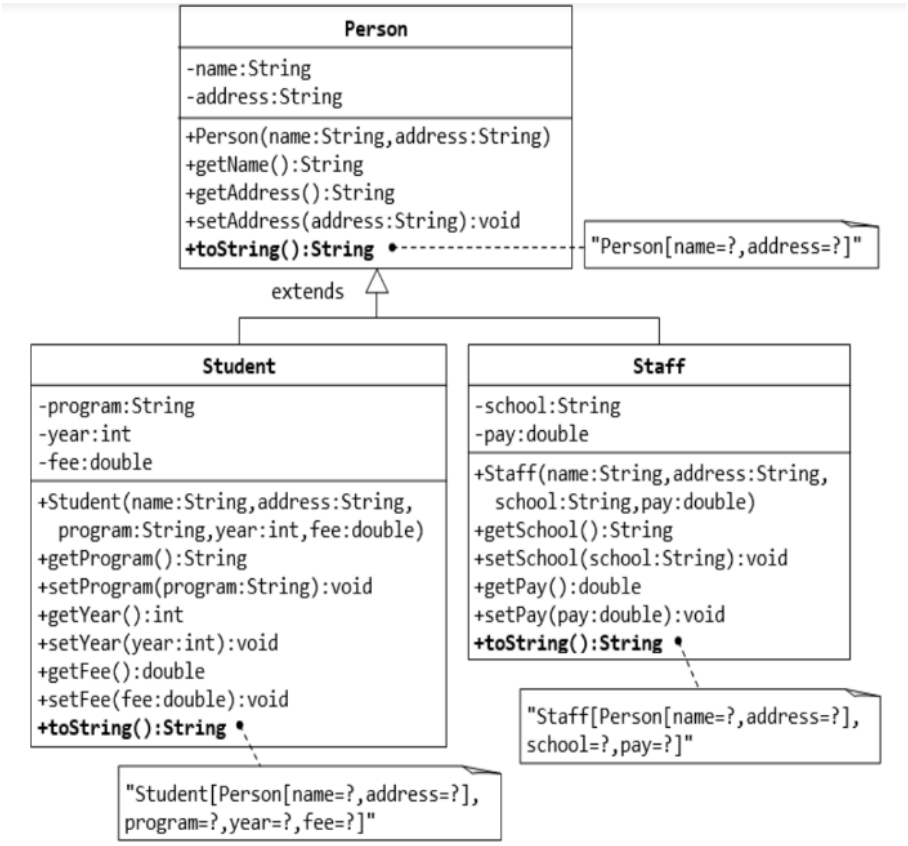
-> Abstract class: It is used to put architectural constraints on design. We cannot create instances for an abstract class, but we can inherit from the abstract class.

-> toString: It is a method present in object class. Every class is initially extended from the object class, hence we can override the toString method. it is used to convert any input to string.

-> 'This': It is used to refer to the current object in a method. It will differentiate between parameters and variables of present class.

-> 'super': It is a reference variable which is used to refer immediate parent class object.

Design:



Program:

```
import java.util.Scanner;

abstract class Person {

    private String name;
    private String address;

    public Person(String name, String address) {
        this.name = name;
        this.address = address;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String toString(){
        return "\n Name: " + this.name + "\n Address: " +
this.address;
    }
}

class Student extends Person {

    private String program;
    private int year;
    private double fee;

    public Student(String name, String address,String program,int
year, double fee) {
        super(name,address);
        this.program = program;
        this.year = year;
```

```
        this.fee = fee;
    }

    public String getProgram() {
        return program;
    }

    public void setProgram(String program) {
        this.program = program;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public double getFee(){
        return fee;
    }

    public void setFee(Double fee){
        this.fee = fee;
    }

    public String toString(){
        return (super.toString()+"\n  Program: " + this.program +
"\n  Year: " + this.year + "\n  Fee: " + this.fee);
    }
}

class Staff extends Person{

    private String school;
    private double pay;

    public Staff(String name, String address,String school,double pay)
{
        super(name, address);
        this.school = school;
        this.pay = pay;
    }

    public String getSchool() {
        return school;
```

```
    }

    public void setSchool(String school) {
        this.school = school;
    }

    public double getPay() {
        return pay;
    }

    public void setPay(double pay) {
        this.pay = pay;
    }

    public String toString(){
        return (super.toString()+"\n  School: " + this.school +
"\n  Pay: " + this.pay);
    }
}

public class termwork_1 {

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

    public static void main(String[] args){

        String name,address,program,school,name1,address1;
        int year;
        double pay,fee;

        System.out.println("  Enter the details of the students:\n");

        System.out.print("  Enter the name of the Student: ");
        name = in.nextLine();

        System.out.print("  Enter the Address of the Student: ");
        address = in.nextLine();

        System.out.print("  Enter the program of the Student: ");
        program = in.nextLine();

        System.out.print("  Enter the year in which the Student is
studying: ");
        year = in.nextInt();

        System.out.print("  Enter the Fees of the Student: ");
        fee = in.nextDouble();
```

```
        in.nextLine();
        System.out.println("\n  Enter the details of the staff:\n");

        System.out.print("  Enter the name of the Staff: ");
        name1 = in.nextLine();

        System.out.print("  Enter the Address of the Staff: ");
        address1 = in.nextLine();

        System.out.print("  Enter the School of the Staff: ");
        school = in.nextLine();

        System.out.print("  Enter the Pay of the Staff: ");
        pay = in.nextDouble();

        Student s = new Student(name,address,program,year,fee);
        Staff s1 = new Staff(name1,address1,school,pay);

        System.out.println("\n  The Details are as follows: \n");

        System.out.println(s.toString());
        System.out.println();
        System.out.println(s1.toString());
    }
}
```

Sample input and output:

Enter the details of the students:

Enter the name of the Student: Rahul
Enter the Address of the Student: Hubli
Enter the program of the Student: CSE
Enter the year in which the Student is studying: 2
Enter the Fees of the Student: 75000

Enter the details of the staff:

Enter the name of the Staff: Rahul
Enter the Address of the Staff: Hubli
Enter the School of the Staff: SDM
Enter the Pay of the Staff: 25000

The Details are as follows:

Name: Rahul
Address: Hubli
Program: CSE
Year: 2
Fee: 75000.0

Name: Rahul
Address: Hubli
School: SDM
Pay: 25000.0

References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

Termwork-2: Prepare object oriented design specification for the given problem description and implement in Java language.

Problem Statement:

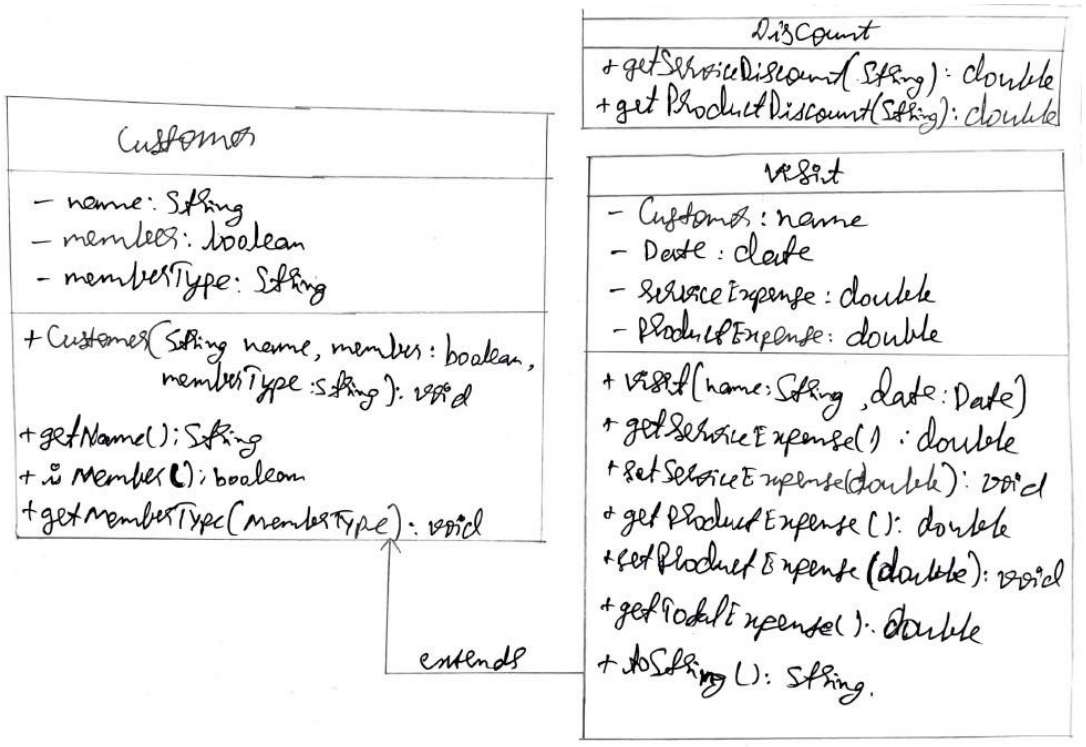
you are asked to write a discount system for a beauty salon, which provides services and sells beauty products. It offers 3 types of memberships: Premium, Gold and Silver. Premium, gold and silver members receive a discount of 20%, 15%, and 10%, respectively, for all services provided. Customers without membership receive no discount. All members receive a flat 10% discount on products purchased. Your system shall consist of three classes: Customer, Discount and Visit. It shall compute the total bill if a customer purchases x of products and y of services, for a visit.

Theory:

-> toString: It is a method present in object class. Every class is initially extended from the object class, hence we can override the toString method. it is used to convert any input to string.

-> 'This': It is used to refer to the current object in a method. It will differentiate between parameters and variables of present class.

Design:



Program:

```
import java.util.*;

class Customer {

    private String name;
    private boolean member;
    private int memberType;
    private int Id;

    public Customer() {
        this.member = false;
    }

      public Customer(int Id,String name, boolean member, int
memberType) {
        this.Id = Id;
        this.name = name;
        this.member = member;
        this.memberType = memberType;
    }

    public String getName() {
        return name;
    }

    public boolean isMember() {
        return member;
    }

    public int getMemberType() {
        return memberType;
    }

    public void setMemberType(int memberType) {
        this.memberType = memberType;
    }

    public int getCustomerId(){
        return Id;
    }

    public String printMemberType(int type){
        if(type==1)
            return "Premium";
        else if(type==2)
```

```
            return "Gold";
        else if(type==3)
            return "Silver";
        else
            return "No Membership";
    }
}

class Visit extends Customer {

    private Customer name;
    private Date date;
    private double serviceExpense;
    private double productExpense;

    public Visit(Customer name, Date date) {
        this.name = name;
        this.date = date;
    }

    public double getServiceExpense() {
        return serviceExpense;
    }

    public void setServiceExpense(double serviceExpense) {
        this.serviceExpense = this.serviceExpense + serviceExpense;
    }

    public double getProductExpense() {
        return productExpense;
    }

    public void setProductExpense(double productExpense) {
        this.productExpense = this.productExpense + productExpense;
    }

    public double getTotalExpense() {
        return  (serviceExpense - (serviceExpense *
Discount.getServiceDiscount(name.getMemberType()))) +
                (productExpense - (productExpense *
Discount.getProductDiscount(name.getMemberType())));

    }

    public String toString() {
        return "\n  customer name: " + name.getName() +"\n  customer
member: " + name.isMember() +
```

```
                "\n  customer member type: " +
name.printMemberType(name.getMemberType()) +"\n  date: " + date +
                "\n  serviceExpense: " + serviceExpense
+"\n  productExpense: " + productExpense;
    }
}

class Discount {

    public static double getServiceDiscount(int type) {
        if(type==1)
            return 0.2;         //service discount premium;
        else if(type==2)
            return 0.15;        //service discount gold;
        else if(type==3)
            return 0.1;         //service discount silver;
        else
            return 0;
    }

    public static double getProductDiscount(int type) {
        if(type==1)
                return 0.1;     //product discount;
        else if(type==2)
                return 0.1;     //product discount;
        else if(type==3)
                return 0.1;     //product discount;
        else
            return 0;
    }
}

public class termwork_2{

    private static int i=0;

    //variables to hold customer details
    private static int customerId;
    private static String customerName;
    private static boolean ismember;
    private static int membertype;
    private static Customer customerPool[] = new Customer[100];
    static int customerIndex = -1;
    static boolean customerFound = false;
    static Scanner in = new Scanner(System.in);

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

```
    int menuChoice=0;

    while(menuChoice!=3){
        try{
            System.out.printf("\n\n");
            System.out.println("  MENU    ");
            System.out.println("  --------------------    ");
            System.out.println("  1 to Create Customer");
            System.out.println("  2 to Customer visit");
            System.out.println("  3 to Exit");
            System.out.print("  Enter an your option : ");
            menuChoice = in.nextInt ();
            switch (menuChoice)
            {
                case 1 : {
                    createCustomer();
                    break;
                }
                case 2 : {
                    customerVisit();
                    break;
                }
            }
        } finally{};
    }
}

private static void createCustomer(){

    System.out.print("\n  Enter customer Id: ");
    customerId = in.nextInt();

    if(isCustomer(customerId) == true){
        System.out.println("  Customer Id already exist");
    }
    else {
        in.nextLine();

        System.out.print("  Enter Customer name: ");
        customerName = in.nextLine();

        System.out.print("  is the customer a member: ");
        ismember = in.nextBoolean();

        if(ismember){
```

```
                System.out.println("\n  1: Premium\n  2: Gold\n  3:
Silver\n  4: No membership\n");
                System.out.print("  Enter the membership type: ");
                membertype = in.nextInt();
            }
            else{
                membertype = 4;
            }

            customerPool[i] = new
Customer(customerId,customerName,ismember,membertype);
            System.out.println("  the customer created is: " +
customerPool[i].toString());
            i++;
        }

    }

    private static void customerVisit(){

        System.out.print("\n  Enter customer Id: ");
        customerId = in.nextInt();

        if(isCustomer(customerId) == true)
            computeBill(customerPool[customerIndex]);
        else
            System.out.println("  Customer not Found");
    }

    private static void computeBill(Customer c){

        double product,service;
        Visit v1 = new Visit(c, new Date());

        System.out.print("  Enter the service expense: ");
        service = in.nextDouble();
        v1.setServiceExpense(service);

        System.out.print("  Enter the product expense: ");
        product = in.nextDouble();
        v1.setProductExpense(product);

        System.out.println("\n  The details of the customer are: ");
        System.out.println(v1.toString());
        System.out.println("  Total expense made by " + c.getName() +
" = " + v1.getTotalExpense());
    }
```

```
    private static boolean isCustomer(int customerId){

        int j=0;
        customerFound=false;

        while((customerFound == false)&&(j<i)){
            if(customerId == customerPool[j].getCustomerId()){
                customerFound = true;
                customerIndex = j;
            }
            j++;
        }
        return customerFound;
    }
}
```

Sample input and output:

CASE 1:

```
MENU
--------------------
1 to Create Customer
2 to Customer visit
3 to Exit
Enter your option : 1

Enter customer Id: 15
Enter Customer name: chris
is the customer a member: true

1: Premium
2: Gold
3: Silver
4: No membership

Enter the membership type: 1
the customer created is: Customer@16b98e56
```

CASE 2:

```
MENU
--------------------
1 to Create Customer
2 to Customer visit
3 to Exit
```

```
Enter your option: 2

Enter customer Id: 15
Enter the product expense: 6000
Enter the service expense: 8000

The details of the customer are:

customer name: chris
customer member: true
customer member type: Premium
date: Sun May 01 19:30:55 IST 2022
serviceExpense: 8000.0
productExpense: 6000.0
Total expense made by chris = 11800.0
```

References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

Termwork-3: Write a Java Program to solve the given problem scenario using package features of the java language.

Problem Statement:

A package named "BasicMath" is to be created which has a class named Basic and has methods to perform following computations:

i. Addition of two numbers
ii. Subtraction of two numbers.
iii. Multiplication of two numbers
iv. Division of two numbers.

Another package named "AdvancedMath" is to be created which has a class named Advanced and has methods to perform following computations using built in features of java.

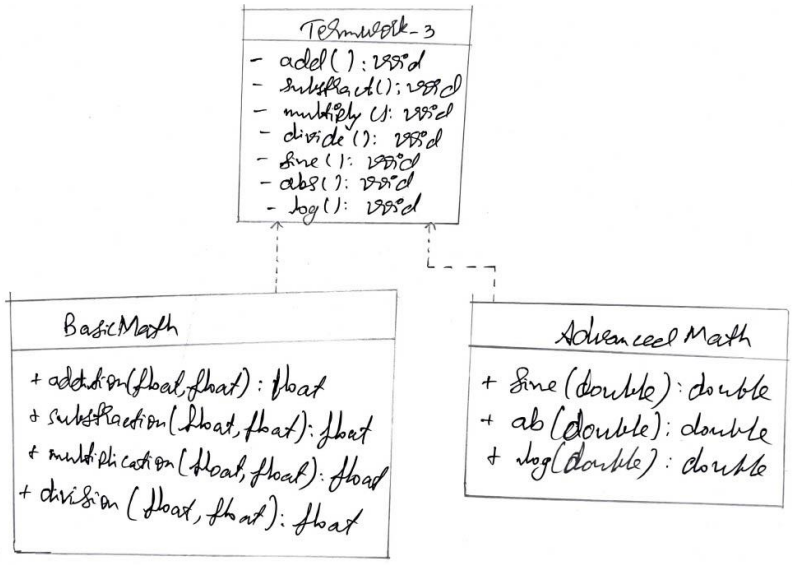
i. Find sine of an angle.
ii. Find ab.
iii. Find log10 of a number.

To test above scenes, another class named TestDemo is to be created defined inside a default package. This class must invoke all the methods of Basic and Advanced classes.

Theory:

-> Packages: Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.

Design:



Program:

```
import BasicMath.Basic;
import AdvancedMath.Advanced;
import java.util.Scanner;
import java.text.DecimalFormat;

public class termwork_3{

    private static final DecimalFormat df = new
DecimalFormat("0.0000");
    private static Scanner in = new Scanner(System.in);

    public static void main(String[] args) {

        int menuChoice=0;

        while(menuChoice!=8){
            try{
                System.out.printf("\n\n\n\n");
                System.out.println("  MENU ");
                System.out.println("  ------------------- ");
                System.out.println("  1 to ADD two numbers");
                System.out.println("  2 to Subtract two numbers");
                System.out.println("  3 to Multiply two numbers");
                System.out.println("  4 to Divide two numbers");
                System.out.println("  5 to find sine of the given
angle in degrees");
                System.out.println("  6 to find absolute value");
                System.out.println("  7 to find log10");
                System.out.println("  8 to Exit");

                System.out.print ("  Enter your option : ");
                menuChoice = in.nextInt ();

                switch (menuChoice)
                {
                    case 1 : {
                        add();
                        break;
                    }
                    case 2 : {
                        subtract();
                        break;
                    }
                    case 3 : {
                        multiply();
```

```
                        break;
                    }
                    case 4 : {
                        divide();
                        break;
                    }
                    case 5 : {
                        sine();
                        break;
                    }
                    case 6 : {
                        abs();
                        break;
                    }
                    case 7 : {
                        log();
                        break;
                    }
                }
            } finally{}
        }
    }

    private static void add() {

        float num1,num2,res;

        System.out.print("  Enter two numbers: ");
        num1=in.nextInt();
        num2=in.nextInt();

        Basic b1 = new Basic();
        res=b1.addition(num1, num2);

        System.out.println("  The addition of two numbers is: " +
df.format(res));
    }

    private static void subtract() {

        float num1,num2,res;

        System.out.print("  Enter two numbers: ");
        num1=in.nextInt();
        num2=in.nextInt();

        Basic b2 = new Basic();
```

```
        res=b2.subtraction(num1, num2);

        System.out.println("  The subtraction of two numbers is: " +
df.format(res));
    }

    private static void multiply() {

        float num1,num2,res;

        System.out.print("  Enter two numbers: ");
        num1=in.nextInt();
        num2=in.nextInt();

        Basic b3 = new Basic();
        res=b3.multiplication(num1, num2);

        System.out.println("  The multiplication of two numbers is: "
+ df.format(res));
    }

    private static void divide() {

        float num1,num2,res;

        System.out.print("  Enter two numbers: ");
        num1=in.nextInt();
        num2=in.nextInt();

        Basic b4 = new Basic();
        res=b4.division(num1, num2);

        System.out.println("  The division of two numbers is: " +
df.format(res));
    }

    private static void sine() {

        double degree,res;

        System.out.print("  Enter the angle in degrees: ");
        degree=in.nextInt();

        Advanced a1 = new Advanced();
        res=a1.sine(degree);
```

```
        System.out.println("  The sine of " + degree + "is:" +
df.format(res));
    }

    private static void abs() {

        double num,res;

        System.out.print("  Enter a number: ");
        num=in.nextInt();

        Advanced a2 = new Advanced();
        res=a2.ab(num);

        System.out.println("  The absolute value is: " +
df.format(res));
    }

    private static void log() {

        double num,res;

        System.out.print("  Enter a number: ");
        num=in.nextInt();

        Advanced a3 = new Advanced();
        res=a3.log(num);

        System.out.println("  The log10 value is: " + df.format(res));
    }
}
```

```
package BasicMath;

public class Basic {

    public float addition(float num1,float num2){
        return(num1+num2);
    }
    public float subtraction(float num1,float num2){
        return(num1-num2);
    }
    public float multiplication(float num1,float num2){
        return(num1*num2);
    }
    public float division(float num1,float num2){
        return(num1/num2);
    }
```

```
}

package AdvancedMath;

public class Advanced {

    public double sine(double num){
        double num1 = Math.toRadians(num);
        return(Math.sin(num1));
    }
    public double ab(double num){
        return(Math.abs(num));
    }
    public double log(double num){
        return(Math.log10(num));
    }
}
```

Sample input and output:

CASE 1:

MENU
---
1 to ADD two numbers
2 to Subtract two numbers
3 to Multiply two numbers
4 to Divide two numbers
5 to find sine of the given angle in degrees
6 to find absolute value
7 to find log10
8 to Exit
Enter your option : 1
Enter two numbers: 2 3
The addition of two numbers is: 5.0000

CASE 2:

MENU
---
1 to ADD two numbers
2 to Subtract two numbers0.
3 to Multiply two numbers
4 to Divide two numbers
5 to find sine of the given angle in degrees
6 to find absolute value
7 to find log10

8 to Exit
Enter your option : 2
Enter two numbers: 3 2
The subtraction of two numbers is: 1.0000

CASE 3:

MENU
---
1 to ADD two numbers
2 to Subtract two numbers
3 to Multiply two numbers
4 to Divide two numbers
5 to find sine of the given angle in degrees
6 to find absolute value
7 to find log10
8 to Exit
Enter your option : 3
Enter two numbers: 2 3
The multiplication of two numbers is: 6.0000

CASE 4:

MENU
---
1 to ADD two numbers
2 to Subtract two numbers
3 to Multiply two numbers
4 to Divide two numbers
5 to find sine of the given angle in degrees
6 to find absolute value
7 to find log10
8 to Exit
Enter your option : 4
Enter two numbers: 4 2
The division of two numbers is: 2.0000

CASE 5:

MENU
---
1 to ADD two numbers
2 to Subtract two numbers
3 to Multiply two numbers
4 to Divide two numbers
5 to find sine of the given angle in degrees

6 to find absolute value
7 to find log10
8 to Exit
Enter your option : 5
Enter the angle in degrees: 45
The sine of 45.0 is:0.7071

CASE 6:

MENU
---
1 to ADD two numbers
2 to Subtract two numbers
3 to Multiply two numbers
4 to Divide two numbers
5 to find sine of the given angle in degrees
6 to find absolute value
7 to find log10
8 to Exit
Enter your option : 6
Enter a number: -8
The absolute value is: 8.0000

CASE 7:

MENU
---
1 to ADD two numbers
2 to Subtract two numbers
3 to Multiply two numbers
4 to Divide two numbers
5 to find sine of the given angle in degrees
6 to find absolute value
7 to find log10
8 to Exit
Enter your option : 7
Enter a number: 150
The log10 value is: 2.1761

References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

Termwork-4: Write a Java Program solve the given problem scenario using string and exceptions features of the java language.

Problem Statement:

a) It takes a string as command-line argument and checks whether it is palindrome or not.

b) If the string is palindrome, then it must print its length and

c) If the string is palindrome, then it converts that string to upper case letters.

d) If the string is not palindrome, then it should generate a user defined exception StringNotPalindromeException.

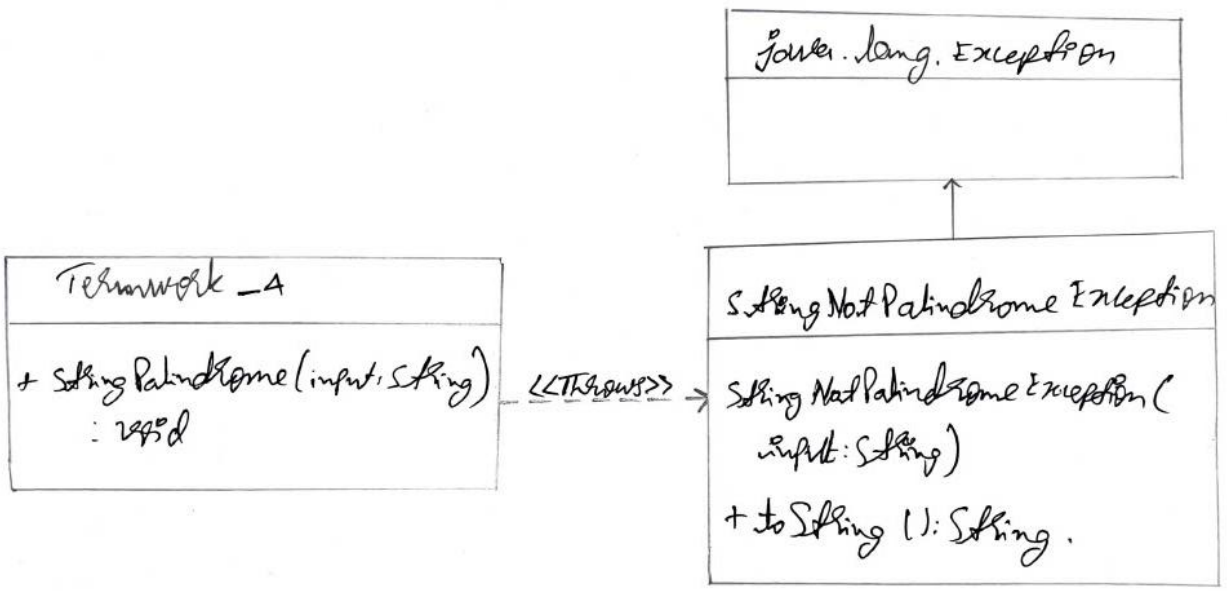
Theory:

-> Exception: A java exception is an object that describes an exceptional condition that has occurred in a piece of code

-> Try: The try statement allows user to define a block of code to be tested for errors while it is being executed.

-> Catch: The catch statement allows user to define a block of code to be executed if an error occurs in the try block.

Design:



Program:

```
class StringNotPalindromeException extends Exception {
}

public class termwork_4{
    public static void main(String[] args){

        int i=0,n;
        String s1,s2="";

        s1=args[0];
        n=s1.length();

        for (i = (n- 1); i >=0; --i) {
            s2 = s2 + s1.charAt(i);
        }

        try {
            if (s1.toLowerCase().equals(s2.toLowerCase()))
            {
                System.out.println(s1 + " is a Palindrome String.");
                System.out.println("The length of the string is " +
s1.length());
                System.out.println("The string in upper case is " +
s1.toUpperCase());
            }
            else
                throw new StringNotPalindromeException();
        } catch (StringNotPalindromeException s) {
            System.out.print("String is not Palindrome: " + s);
        }
    }
}
```

Sample input and output:

CASE 1:

PS C:\Termwork_4> java termwork_4
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds for length 0 at termwork_4.main(termwork_4.java:38)

CASE 2:

PS C:\ Termwork_4> java termwork_4 omkar
String is not Palindrome: StringNotPalindromeException

CASE 3:

```
PS C:\Termwork_4> java termwork_4 gadag
gadag is a Palindrome String.
The length of the string is 5
The string in upper case is GADAG
```

References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

Termwork-5: Write a Java Program solve the given problem scenario using threads features of the java language.

Problem Statement:

a) Creates three threads.

b) The first thread should print the prime numbers between 1 and 100;

c) Second thread should print prime numbers between 101 and 200; and

d) Third thread should print prime numbers between 201 and 300.

e) It is mandatory that first thread must print first, followed by second and third threads.

f) Output should be:

Thread-SMJ: Primes between 1 and 100

Thread-JVV: Primes between 101 and 200

Thread-Myself: Primes between 201 and 300

g) All threads must call the same method generatePrime () to print prime numbers.

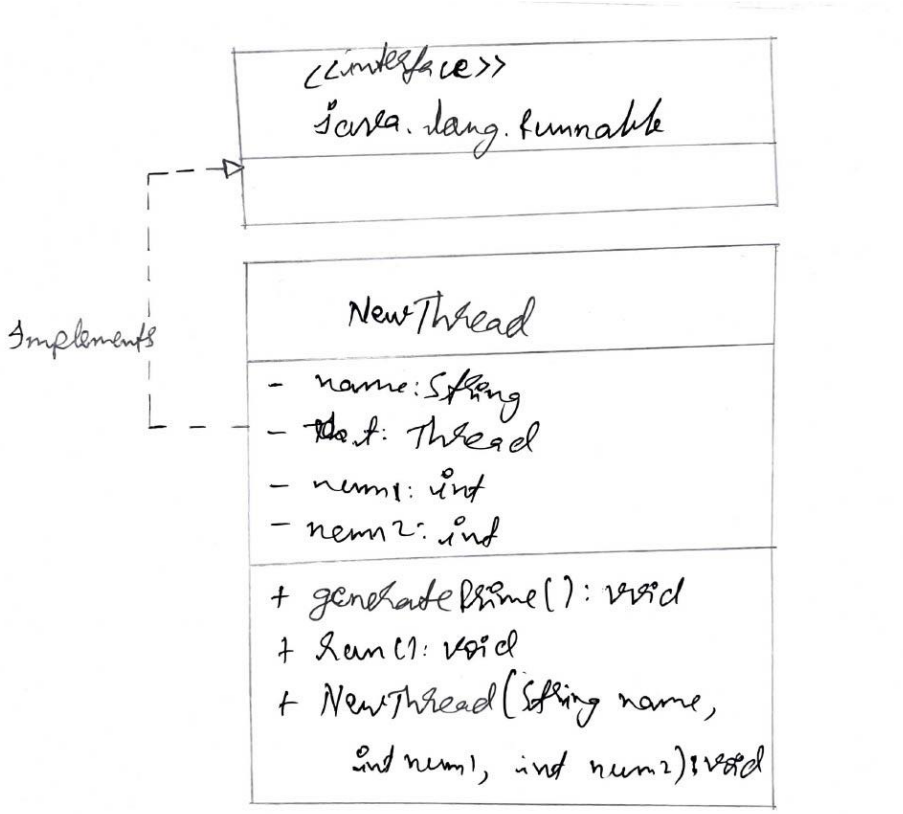
Theory:

-> Threads: Threads are subprocess with lightweight with the smallest unit of processes and also has separate paths of execution.

Threads can be created using two methods:

1) By extending Thread Class.
2) By Implementing a Runnable interface.

Design:



Program:

```
import java.util.*;

class NewThread implements Runnable {

    String name;
    Thread t;
    int num1,num2;

    public synchronized void generatePrime(String name,int num1,int
num2){

        int flag;
        System.out.println("The prime numbers between " + num1 + " and
" + num2 + " are: ");

        for(int i = num1; i<=num2; i++) {
            flag = 1;
            for(int j = 2; j<=Math.sqrt(i); j++)
            {
                if(i%j == 0)
                {
```

```
                    flag = 0;
                    break;
                }
            }
            if(flag == 1)
                System.out.println(i);
        }
    }

    NewThread(String threadname, int num1,int num2) {
        name = threadname;
        t = new Thread(this, name);
        this.num1=num1;
        this.num2=num2;
    }

    public  void run() {
        System.out.println("New thread: " + t);
        generatePrime(name, num1, num2);
    }
}

class termwork_5 {
    public static void main(String args[]) {

        NewThread t1 = new NewThread("SMJ",1,100);
        NewThread t2 = new NewThread("JVV",101,200);
        NewThread t3 = new NewThread("Myself",201,300);

        try{
            t1.t.start();
            t1.t.join();

            t2.t.start();
            t2.t.join();

            t3.t.start();
            t3.t.join();
        }catch(InterruptedException e){
            e.printStackTrace();
        }
    }
}
```

Sample input and output:

```
New thread: Thread[SMJ,5,main]
The prime numbers between 1 and 100 are:
1
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
New thread: Thread[JVV,5,main]
The prime numbers between 101 and 200 are:
101
103
107
109
113
127
131
137
139
149
151
157
163
167
```

```
173
179
181
191
193
197
199
New thread: Thread[Myself,5,main]
The prime numbers between 201 and 300 are:
211
223
227
229
233
239
241
251
257
263
269
271
277
281
283
293
```

References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

Termwork-6: Write a Java Program solve the given problem scenario using interface features of the java language.

Problem Statement:

a) Create an interface AdvancedArithmetic which contains a method signature int divisor_sum (int n).

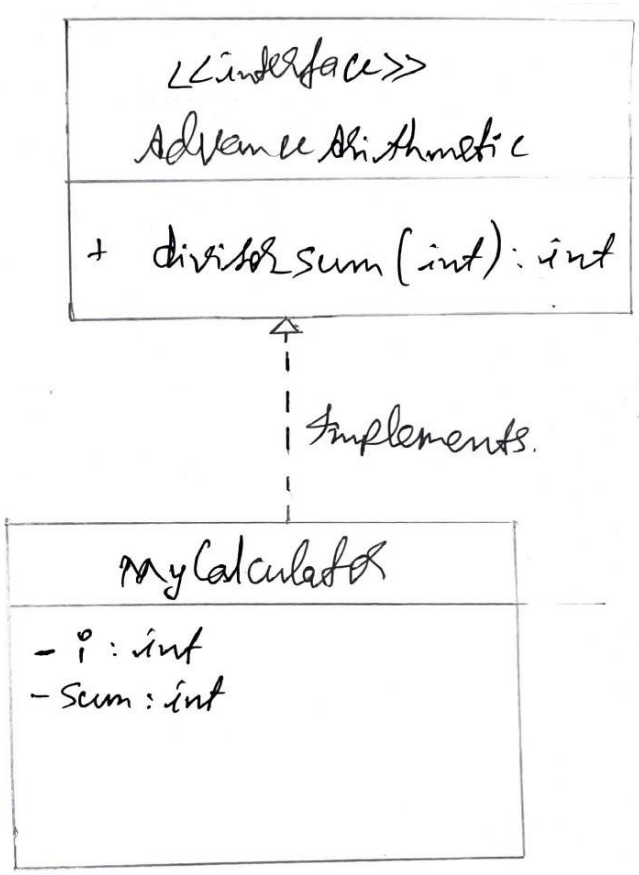
b) Create a class called MyCalculator which implements the interface.

c) The divisorSum function returns the sum of all the divisors of 'n' supplied as parameter. For example, divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 1000.

Theory:

-> interface: An interface in JAVA programming language is defined as an abstract type used to specify the behaviour of a class. An interface contains static constant and abstract methods. In the JAVA interface only abstract methods are permitted not the method body.

Design:



Program:

```
import java.util.Scanner;

interface AdvancedArithmetic{
   int divisor_sum(int n);
}

class MyCalculator implements AdvancedArithmetic{

    public int divisor_sum(int n){

        int sum=1,i;

        if(n<=1000)
        {
            for(i=2;i<=(n/2);i++)
            {
                if(n%i==0)
                    sum=sum+i;
            }
            if(n!=1)
                return sum+n;
            else
                return sum;
        }
        else
        {
            System.out.println("\nNumber is outside the range\n");
            System.exit(0);
            return(0);
        }
    }
}

public class termwork_6 {

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

    public static void main(String[] args) {

        System.out.print("\nEnter the number: ");
        int n=in.nextInt();

        AdvancedArithmetic c = new MyCalculator();
        int sum = c.divisor_sum(n);
```

```
        System.out.println("\nThe sum of all the divisors of " + n + "
is: " + sum);
    }
}
```

Sample input and output:

CASE 1:

Enter the number: 254

The sum of all the divisors of 254 is: 384

CASE 2:

Enter the number: 2585

Number is outside the range

References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

Termwork-7: Write a Java Program solve the given problem scenario using streams features of the java language.

Problem Statement:

Write a Java Program to do the following.

a) An input file named StudentList.txt, which contains first names of students belonging to a batch.

b) Program should extract the first appearing name in the file and find the length of the string extracted from the file using appropriate String method.

c) Program should find whether character “a” is present in extracted string or not, if yes, print the positions of all the occurrences of “a” in the extracted string.

Theory:

-> Streams: A stream is an abstraction that can either produces or consumes information. A stream is linked to a physical device by the Java I/O device.

Java defines two types of streams:

Byte: Byte streams provide a convenient means for handling input and output of bytes.
Character: Character provide a convenient means for handling input and output of characters.

-> StreamTokenizer: StreamTokenizer breaks up the Input Stream into tokens that are delimited by sets of characters.

Program:

```
import java.io.*;

public class termwork_7 {

    public static void main(String[] args) throws IOException{

        try{
            BufferedReader reader = new BufferedReader(new
FileReader(args[0]));
            StreamTokenizer tok = new StreamTokenizer(reader);

            while(tok.nextToken()!=TT_EOF)
            {
                String line = tok.sval;
```

```
                if(line!=null)
                {
                    System.out.println("\n" + line);
                }

                line.length();
                System.out.println("\nThe length of the string is : "
+ line.length() + "\n");

                for(int i=0;i<line.length();i++)
                {
                    if(line.charAt(i)=='a')
                    {
                        System.out.println("'a' is present at index :
"+ (i+1));
                    }
                }
            }
        }catch(NullPointerException e){}
        catch(IOException e){}
    }
}
```

Sample input and output:

Meeth
The length of the string is : 5

Sakaria
The length of the string is : 7
'a' is present at index : 2
'a' is present at index : 4
'a' is present at index : 7

Basu
The length of the string is : 4
'a' is present at index : 2

Omkar
The length of the string is : 5
'a' is present at index : 4

References:

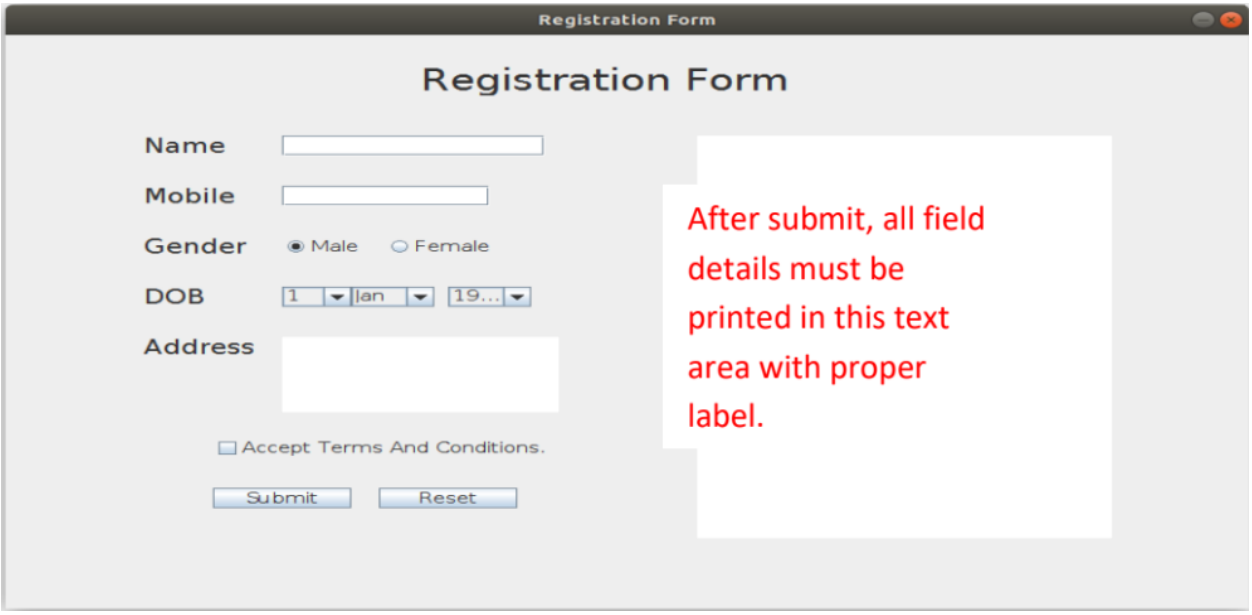
1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.

2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

Termwork-8: Write a Java Program solve the given problem scenario using events and AWR features of the java language.

Problem Statement:

Write a Java program to build the following user registration form.



Theory:

-> Java AWT (Abstract Window Toolkit): It is an API to develop Graphical User Interface (GUI) or windows-based applications in Java. Java AWT components are platform dependent i.e., components are displayed according to the view of the operating system. AWT is heavy weight i.e., it's components are using the resources of underlying operating system.

-> Events Handling: Changing the state of an object is known as an event. The java.awt.event package provides many event classes and listener interfaces for event handling.

Program:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class registrationForm extends JFrame implements ActionListener {

    private Label title;
    private Label name;
    private TextField tname;
```

```
    private Label mobileno;
    private TextField tmobileno;
    private Label gender;
    private JRadioButton male;
    private JRadioButton female;
    private ButtonGroup gengp;
    private Label dob;
    private JComboBox date;
    private JComboBox month;
    private JComboBox year;
    private Label add;
    private JTextArea tadd;
    private JCheckBox term;
    private Button submit;
    private Button reset;
    private JTextArea tout;
    private Label res;

    private String[] dates = { "1", "2", "3", "4", "5", "6", "7", "8",
"9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
"21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31" };

    private String[] months = { "Jan", "feb", "Mar", "Apr", "May",
"Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec" };

    private String[] years = { "1995", "1996", "1997", "1998","1999",
"2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007",
"2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015",
"2016", "2017", "2018", "2019", "2020", "2021", "2022" };

    public registrationForm()
    {
        setTitle("Registration Form");
        setBounds(300, 90, 900, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setResizable(false);

        setLayout(null);

        title = new Label("Registration Form");
        title.setFont(new Font("Arial", Font.PLAIN, 30));
        title.setSize(300, 30);
        title.setLocation(300, 30);
        add(title);

        name = new Label("Name");
        name.setFont(new Font("Arial", Font.PLAIN, 20));
```

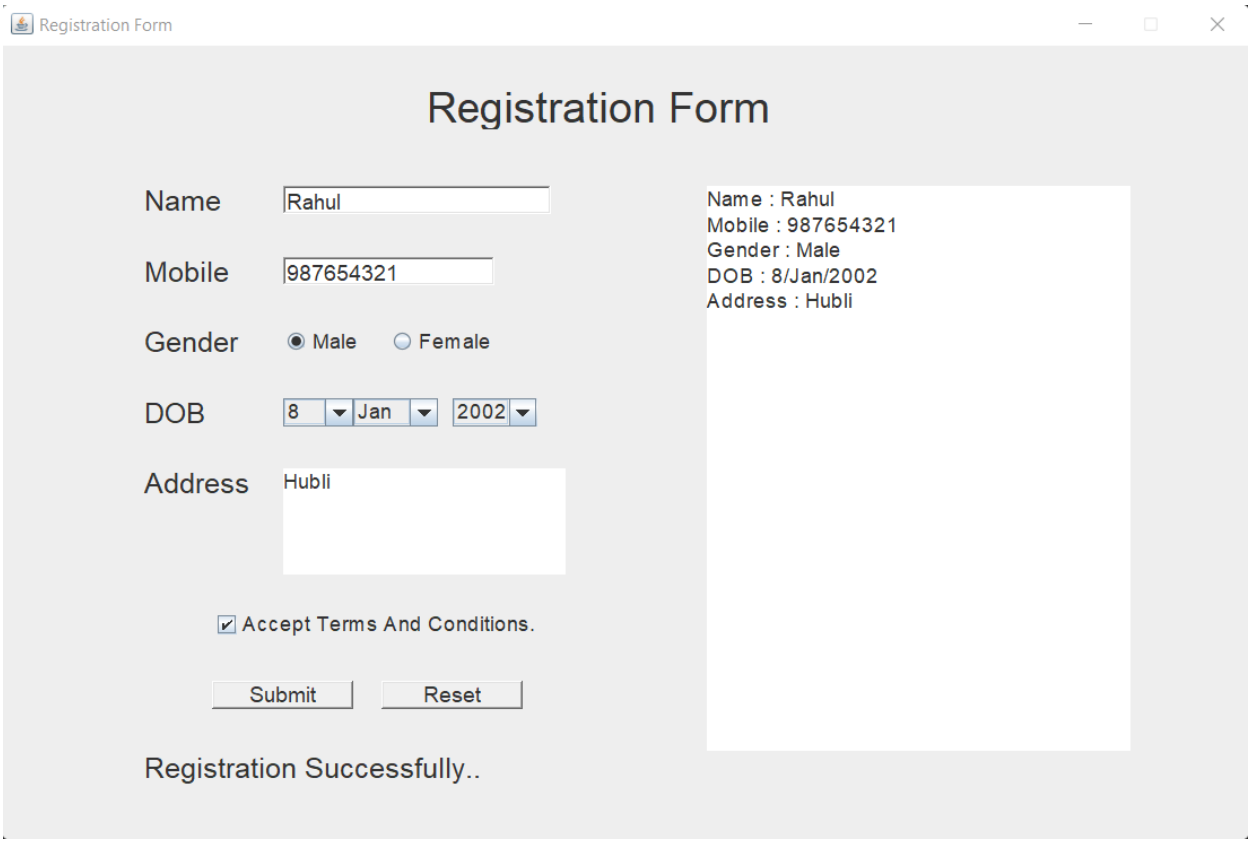
```
name.setSize(100, 20);
name.setLocation(100, 100);
add(name);

tname = new TextField();
tname.setFont(new Font("Arial", Font.PLAIN, 15));
tname.setSize(190, 20);
tname.setLocation(200, 100);
add(tname);

mobileno = new Label("Mobile");
mobileno.setFont(new Font("Arial", Font.PLAIN, 20));
mobileno.setSize(100, 20);
mobileno.setLocation(100, 150);
add(mobileno);

tmobileno = new TextField();
tmobileno.setFont(new Font("Arial", Font.PLAIN, 15));
tmobileno.setSize(150, 20);
tmobileno.setLocation(200, 150);
add(tmobileno);

gender = new Label("Gender");
gender.setFont(new Font("Arial", Font.PLAIN, 20));
gender.setSize(100, 20);
gender.setLocation(100, 200);
add(gender);

male = new JRadioButton("Male");
male.setFont(new Font("Arial", Font.PLAIN, 15));
male.setSelected(true);
male.setSize(75, 20);
male.setLocation(200, 200);
add(male);

female = new JRadioButton("Female");
female.setFont(new Font("Arial", Font.PLAIN, 15));
female.setSelected(false);
female.setSize(80, 20);
female.setLocation(275, 200);
add(female);

gengp = new ButtonGroup();
gengp.add(male);
gengp.add(female);

dob = new Label("DOB");
```

```
dob.setFont(new Font("Arial", Font.PLAIN, 20));
dob.setSize(100, 20);
dob.setLocation(100, 250);
add(dob);

date = new JComboBox(dates);
date.setFont(new Font("Arial", Font.PLAIN, 15));
date.setSize(50, 20);
date.setLocation(200, 250);
add(date);

month = new JComboBox(months);
month.setFont(new Font("Arial", Font.PLAIN, 15));
month.setSize(60, 20);
month.setLocation(250, 250);
add(month);

year = new JComboBox(years);
year.setFont(new Font("Arial", Font.PLAIN, 15));
year.setSize(60, 20);
year.setLocation(320, 250);
add(year);

add = new Label("Address");
add.setFont(new Font("Arial", Font.PLAIN, 20));
add.setSize(100, 20);
add.setLocation(100, 300);
add(add);

tadd = new JTextArea();
tadd.setFont(new Font("Arial", Font.PLAIN, 15));
tadd.setSize(200, 75);
tadd.setLocation(200, 300);
add(tadd);

term = new JCheckBox("Accept Terms And Conditions.");
term.setFont(new Font("Arial", Font.PLAIN, 15));
term.setSize(250, 20);
term.setLocation(150, 400);
add(term);

submit = new Button("Submit");
submit.setFont(new Font("Arial", Font.PLAIN, 15));
submit.setSize(100, 20);
submit.setLocation(150, 450);
submit.addActionListener(this);
add(submit);
```

```
        reset = new Button("Reset");
        reset.setFont(new Font("Arial", Font.PLAIN, 15));
        reset.setSize(100, 20);
        reset.setLocation(270, 450);
        reset.addActionListener(this);
        add(reset);

        tout = new JTextArea();
        tout.setFont(new Font("Arial", Font.PLAIN, 15));
        tout.setSize(300, 400);
        tout.setLocation(500, 100);
        tout.setEditable(false);
        add(tout);

        res = new Label("");
        res.setFont(new Font("Arial", Font.PLAIN, 20));
        res.setSize(500, 25);
        res.setLocation(100, 500);
        add(res);

        setVisible(true);
    }

    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == submit)
        {
            if (term.isSelected())
            {
                String data,data1,data2,data3;
                data = "Name : " + tname.getText() + "\n" + "Mobile :
" + tmobileno.getText() + "\n";
                if (male.isSelected())
                    data1 = "Gender : Male" + "\n";
                else
                    data1 = "Gender : Female" + "\n";

                data2 = "DOB : " + (String) date.getSelectedItem() +
"/" + (String) month.getSelectedItem() + "/" + (String)
year.getSelectedItem() + "\n";
                data3 = "Address : " + tadd.getText();
                tout.setText(data + data1 + data2 + data3);
                tout.setEditable(false);
                res.setText("Registration Successfully..");
            }
            else
```

```
            {
                tout.setText("");
                res.setText("Please accept the terms & conditions..");
            }
        }
        else if (e.getSource() == reset)
        {
            String def = "";
            tname.setText(def);
            tadd.setText(def);
            tmobileno.setText(def);
            res.setText(def);
            tout.setText(def);
            term.setSelected(false);
            date.setSelectedIndex(0);
            month.setSelectedIndex(0);
            year.setSelectedIndex(0);
        }
    }
}

class termwork_8 {
    public static void main(String[] args) throws Exception{
        new registrationForm();
    }
}
```

Sample input and output:



References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

Laboratory Exam (CIE) - OPEN BOOK TEST: Write a Java Program solve the given problem scenario – Hybrid Mode (College + Work from Home)

Problem Statement:

You are asked to develop a software system to automate day-today operations of a company which contains several departments. Each department has a unique name, a unique number, and a particular employee is designated as manages the department, i.e., a role called manager. A department controls a number of projects, each of which has a unique name, and a single location. Record of each employee i.e., their name, social security number (SSN, which is employee ID), address, salary, gender, and birth date are to be stored and modified on request, approved by the concerned authority. An employee is assigned to one department and work on several projects, which are not necessarily controlled by the same department. Software is supposed to keep track of the following:

- Start date when the employee began managing the department.
- Number of hours per week that an employee works on each project.
- Direct supervisor of each employee.
- Dependents of each employee for insurance purposes. For this we maintain dependent's first name, gender, birth date, and relationship to the employee.

Do the following:

Write a complete Java program to implement the design with following use-cases

a) Register and Unregister Employees.
b) Register and Unregister Projects.
c) Assign/withdraw Projects to the departments.
d) Assign/withdraw projects to employees.
e) Print project details and employee details.
f) Other tracking activities mentioned above.

Theory:

-> toString: It is a method present in object class. Every class is initially extended from the object class, hence we can override the toString method. it is used to convert any input to string.

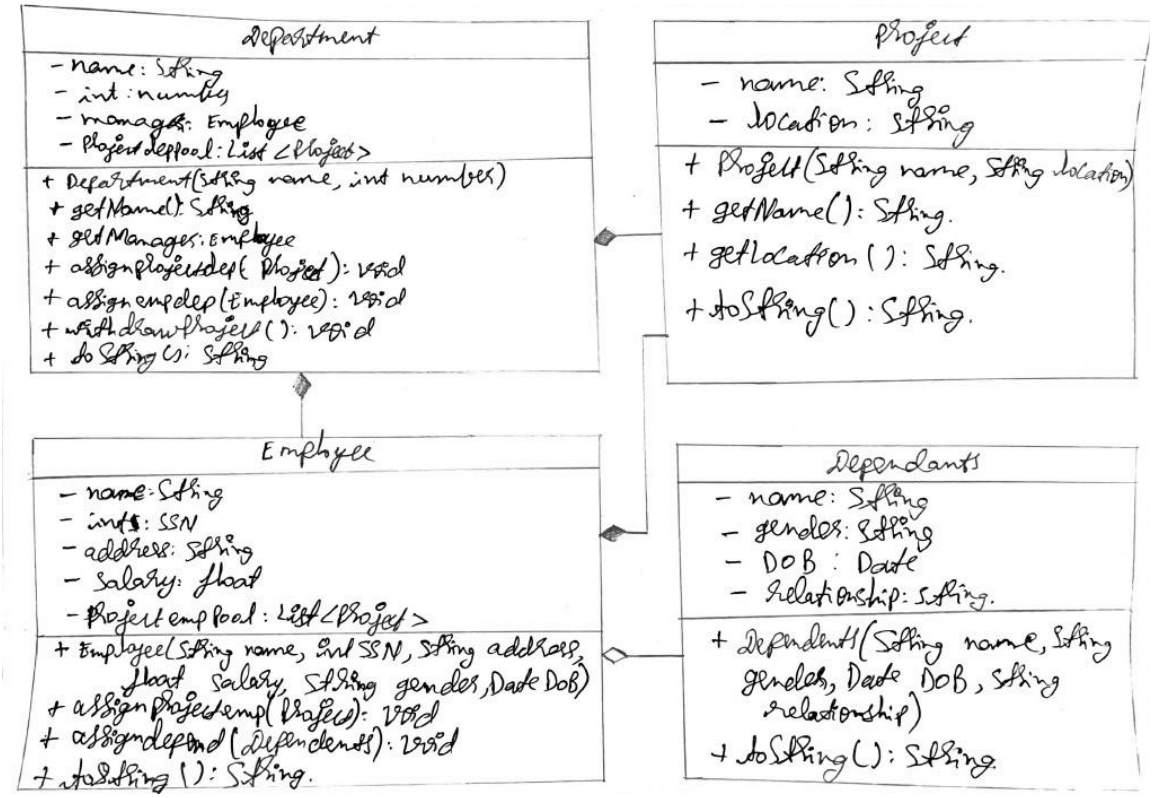
-> 'This': It is used to refer to the current object in a method. It will differentiate between parameters and variables of present class.

-> Exception: A java exception is an object that describes an exceptional condition that has occurred in a piece of code

-> Try: The try statement allows user to define a block of code to be tested for errors while it is being executed.

-> Catch: The catch statement allows user to define a block of code to be executed if an error occurs in the try block.

Design:



Program:

```
import java.text.SimpleDateFormat;
import java.util.*;

class Department {
    private String name;
    private int number;
    private Employee manager;
    public List<Project> projectdeppool = new ArrayList<Project>();
    public List<Employee> employeedeppool = new ArrayList<Employee>();
    public int b=0;
    public int w=0;

    public Department(String name,int number){
        this.name=name;
```

```
    this.number=number;
}

public void setName(String name){
    this.name=name;
}

public String getName(){
    return name;
}

public void setManager(Employee manager) {
    this.manager=manager;
}

public Employee getManager(){
    return manager;
}

public int getNumber(){
    return number;
}

public void assignprojectdep(Project p){
    projectdeppool.add(b,p);
    b++;
}

public void withdrawProjectDep(int f){
    projectdeppool.remove(f);
    b--;
}

public void assignempdep(Employee e){
    employeedeppool.add(w,e);
    w++;
}

public String toString(){
    return ("\n Department name: " + name +
            "\n Department number: " + number +
            "\n Manager: " + manager );
}

public String printProject(){
    String s="";
    for(Project p1 : projectdeppool)
```

```
        {
            s += p1.toString() + "\n";
        }
        return "\n\n The details of projects assigned to " + name + "
are: \n" + s;
    }

    public String printEmployee(){
        String s="";
        for(Employee e1 : employeedeppool)
        {
            s += e1.toString() + "\n";
        }
        return "\n\n The details of employee assigned to " + name + "
are: \n" + s;
    }
}

class Project{
    private String name;
    private String location;

    public Project(String name,String location){
        this.name=name;
        this.location=location;
    }

    public String getName(){
        return name;
    }

    public String getLocation(){
        return location;
    }

    public String toString(){
        return ("\n Project name: " + name +
                "\n Project location: " + location);
    }
}

class Employee{
    private String name;
    private int SSN;
    private String address;
    private float salary;
    private String gender;
```

```
    private Date DOB;
    public int man;
    public List<Project> projectemppool = new ArrayList<Project>();
    public List<Dependents> empdependents = new ArrayList<Dependents>();
    public int dep=0;
    public int c=0;

    public Employee(String name,int SSN,String address,float
salary,String gender,Date DOB){
        this.name=name;
        this.SSN=SSN;
        this.address=address;
        this.salary=salary;
        this.gender=gender;
        this.DOB=DOB;
    }

    public void setName(String name){
        this.name=name;
    }

    public void setSSN(int SSN){
        this.SSN=SSN;
    }

    public int getSSN(){
        return SSN;
    }

    public void setAddress(String address){
        this.address=address;
    }

    public void setSalary(float salary){
        this.salary=salary;
    }

    public void setgender(String gender){
        this.gender=gender;
    }

    public void setDOB(Date DOB){
        this.DOB=DOB;
    }

    public void assignprojectemp(Project p){
        projectemppool.add(c,p);
```

```
        c++;
    }

    public void withdrawProjectEmp(int f){
        projectemppool.remove(f);
        c--;
    }

    public void assigndepend(Dependents de){
        empdependents.add(dep,de);
        dep++;
    }

    public String toString(){
        return ("\n Employee SSN: " + SSN +
                "\n Employee name: " + name +
                "\n Employee address: " + address +
                "\n Employee Salary: " + salary +
                "\n Employee gender: " + gender +
                "\n Employee date of birth: " + DOB );
    }

    public String printProject(){
        String s="";
        for(Project p1 : projectemppool)
        {
            s +=p1.toString() + "\n";
        }
        return ("\n\n The details of projects assigned to " + name + "
are: \n" + s);
    }

    public String printDependents(){
        String s="";
        for(Dependents d1 : empdependents)
        {
            s += d1.toString() + "\n";
        }
        return ("\n\n The details of the dependents are: \n" + s);
    }
}

class Dependents {
    private String name;
    private String gender;
    private Date DOB;
    private String relationship;
```

```
    public Dependents(String name,String gender,Date DOB,String
relationship){
        this.name=name;
        this.gender=gender;
        this.DOB=DOB;
        this.relationship=relationship;
    }

    public String toString(){
        return ("\n Dependent's name: " + name +
                "\n Dependent's gender: " + gender +
                "\n Dependent's date of birth: " + DOB +
                "\n Dependent's relation: " + relationship + "\n");
    }
}

public class mainclass{
    private static int i=0,j=0,k=0;
    private static int empid;
    private static int depid;
    private static String empname;
    private static String depname;
    private static String proname;
    private static String dependname;
    private static Date empdate;
    private static Date dependdate;
    static int employeeindex=-1;
    static int departmentindex=-1;
    static int projectindex=-1;
    static int depprojectindex=-1;
    static int empprojectindex=-1;
    static int depemployeeindex=-1;
    private static List<Department> departmentpool = new
ArrayList<Department>();
    private static List<Employee> employeepool = new
ArrayList<Employee>();
    private static List<Project> projectpool = new ArrayList<Project>();
    static boolean projectfound=false;
    static boolean employeefound=false;
    static boolean departmentfound=false;
    static boolean depprojectfound=false;
    static boolean empprojectfound=false;
    static boolean depemployeefound=false;
    static Scanner in = new Scanner(System.in);

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

```
        int menuChoice=0;
        while(menuChoice!=8){
            try
            {
            System.out.println("\nMain Menu");
            System.out.println("------------------");
            System.out.println("\n1: Register department");
            System.out.println("2: Register and unregister employee");
            System.out.println("3: Register and unregister projects");
            System.out.println("4: Assign employee to department");
            System.out.println("5: Assign/withdraw projects to the
department");
            System.out.println("6: Assign/withdraw projects to the
employee");
            System.out.println("7: Details of Employees, Projects and
Departments");
            System.out.println("8: Exit\n");

            System.out.print("Enter your option : ");
            menuChoice = in.nextInt();

                switch(menuChoice)
                {
                    case 1:{
                        registerDep();
                        break;
                    }

                    case 2:{
                        registerEmp();
                        break;
                    }

                    case 3:{
                        registerPro();
                        break;
                    }

                    case 4:{
                        assignEmployeeToDepartment();
                        break;
                    }

                    case 5:{
                        assignProjectDep();
                        break;
                    }
```

```
                case 6:{
                    assignProjectEmp();
                    break;
                }

                case 7:{
                    details();
                    break;
                }
            }
        }finally{}
    }
}

private static void registerDep(){
    int choice=0;
    while(choice!=4){
        System.out.println("\n1: Register department");
        System.out.println("2: Assign manager");
        System.out.println("3: Withdraw manager");
        System.out.println("4: Return to Main menu\n");
        System.out.print("Enter an your option : ");
        choice=in.nextInt();
        switch(choice){
            case 1:{
                registerDeparment();
                break;
            }
            case 2:{
                assignManager();
                break;
            }
            case 3:{
                withdrawManager();
                break;
            }
        }
    }
}

private static void registerDeparment(){
    System.out.print("\nEnter Department Id: ");
    depid = in.nextInt();

    if(isDepartment(depid) == true){
        System.out.println("\nDepartment Id already exist");
```

```
        }
        else {
            in.nextLine();

            System.out.print("Enter Department name: ");
            depname = in.nextLine();
            Department d = new Department(depname,depid);
            departmentpool.add(j,d);
            System.out.println("\nThe Department is created");
            j++;
        }
    }

    private static void assignManager(){
        System.out.print("\nEnter Department Id: ");
        depid = in.nextInt();

        if(isDepartment(depid) == true){
            System.out.print("Enter the employee Id for manager: ");
            empid=in.nextInt();

            if(isEmployee(empid) == true){
                Employee depmanager=employeepool.get(employeeindex);
                employeepool.get(employeeindex).man=departmentindex;
                departmentpool.get(departmentindex).setManager(depmana
ger);
            }
            else {
                System.out.println("\nManager not assigned");
                System.out.println("Employee Id does not exist");
            }
        }
        else{
            System.out.println("Department Id does not exist");
        }
    }

    private static void withdrawManager(){
        System.out.print("\nEnter Department Id: ");
        depid = in.nextInt();

        if(isDepartment(depid) == true){
            Employee e =
departmentpool.get(departmentindex).getManager();
            isEmployee(e.getSSN());
            employeepool.get(employeeindex).man=0;
            departmentpool.get(departmentindex).setManager(null);
```

```
        }
        else {
            System.out.println("\nDepartment Id does not exist");
        }
    }

    private static void registerEmp(){
        int choice=0;
        while(choice!=4){
            System.out.println("\n1: Register employee");
            System.out.println("2: Unregister employee");
            System.out.println("3: Assign dependents of employee");
            System.out.println("4: Return to Main menu\n");
            System.out.print("Enter your option : ");
            choice=in.nextInt();

            switch(choice){
                case 1:{
                    registerEmployee();
                    break;
                }
                case 2:{
                    unregisterEmploye();
                    break;
                }
                case 3:{
                    assignDependent();
                    break;
                }
            }
        }
    }

    private static void registerEmployee(){
        System.out.print("\nEnter Employee Id: ");
        empid = in.nextInt();

        if(isEmployee(empid) == true){
            System.out.println("\nEmployee Id already exist");
        }
        else {
            in.nextLine();

            System.out.print("Enter Employee name: ");
            empname = in.nextLine();

            System.out.print("Enter address: ");
```

```
            String empaddress = in.nextLine();

            System.out.print("Enter salary: ");
            float empsalary = in.nextFloat();

            in.nextLine();

            System.out.print("Enter Gender: ");
            String empgender = in.nextLine();

            System.out.print("Enter Date of birth: ");
            String empdob = in.nextLine();

            try {
                empdate=new
SimpleDateFormat("dd/MM/yyyy").parse(empdob);
            }catch (Exception e) {
                System.out.println("Invalid Date of birth");
                System.exit(0);
            }

            Employee e1 = new
Employee(empname,empid,empaddress,empsalary,empgender,empdate);
            employeepool.add(i,e1);
            System.out.println("\nThe employee is created");
            i++;
        }
    }

    private static void unregisterEmploye(){
        System.out.print("\nEnter the employee id to unregister: ");
        empid=in.nextInt();

        if(isEmployee(empid) == true){
            departmentpool.get(employeepool.get(employeeindex).man).se
tManager(null);
            employeepool.get(employeeindex).man=0;
            employeepool.remove(employeeindex);

            System.out.println("\nEmployee unregisterd");
            i--;
        }
        else {
            System.out.println("\nEmployee Id does not exist");
        }
    }
```

```
    private static void assignDependent(){
        System.out.print("\nEnter the employee id to assign
dependents: ");
        empid=in.nextInt();

        if(isEmployee(empid)){
            System.out.println("\nEnter the number of dependents: ");
            int numdep = in.nextInt();
            for(int y=0;y<numdep;y++){
                Dependents de=assignDependents();
                employeepool.get(employeeindex).assigndepend(de);
            }
            System.out.println("\nDependents assigned");
        }
        else{
            System.out.println("\nEmployee Id does not exist");
        }
    }

    private static Dependents assignDependents(){
        in.nextLine();
        System.out.print("Enter the name of the dependents: ");
        dependname=in.nextLine();
        System.out.print("Enter the gender of the dependents: ");
        String dependgender=in.nextLine();
        System.out.print("Enter the date of birth of the dependents: ");
        String dependdob=in.nextLine();
        try {
            dependdate=new
SimpleDateFormat("dd/MM/yyyy").parse(dependdob);
        }catch (Exception e) {
            System.out.println("Invalid Date of birth");
            System.exit(0);
        }
        System.out.print("Enter the relationship of the dependents:
");
        String dependrelation=in.nextLine();
        Dependents d1 = new
Dependents(dependname,dependgender,dependdate,dependrelation);
        return d1;
    }

    private static void registerPro(){
        int choice=0;
        while(choice!=3){
            System.out.println("1: Register project");
            System.out.println("2: Unregister project");
```

```
            System.out.println("3: Return to Main menu\n");
            System.out.print("\nEnter your option : ");
            choice=in.nextInt();

            switch(choice){
                case 1:{
                    registerProject();
                    break;
                }
                case 2:{
                    unregisterProject();
                    break;
                }
            }
        }
    }

    private static void registerProject(){
        System.out.print("\nEnter project name: ");
        proname = in.next();

        if(isProject(proname) == true){
            System.out.println("\nProject name already exist");
        }
        else {
            in.nextLine();

            System.out.print("Enter Project location: ");
            String prolocation = in.nextLine();
            Project p = new Project(proname,prolocation);
            projectpool.add(k,p);
            System.out.println("\nThe project is created.");
            k++;
        }
    }

    private static void unregisterProject(){
        in.nextLine();

        System.out.print("Enter the project name to unregister: ");
        proname=in.nextLine();

        if(isProject(proname) == true){
            projectpool.remove(projectindex);
            System.out.println("\nEmployee is unregister");
            k--;
        }
```

```
        else{
            System.out.println("\nEmployee Id does not exist");
        }
    }

    private static void assignEmployeeToDepartment(){
        System.out.print("Enter the deparment Id: ");
        depid=in.nextInt();

        in.nextLine();

        System.out.print("Enter the employee Id to be assigned: ");
        empid=in.nextInt();

        if(isEmployee(empid)&&isDepartment(depid)){
            if(isdepEmployee(departmentpool.get(departmentindex),
empid)){
                System.out.println("\nEmployee already assigned");
            }
            else{
                Employee depemployee =
employeepool.get(employeeindex);
                departmentpool.get(departmentindex).assignempdep(depem
ployee);
                System.out.println("\nEmployee is assigned to
department");
            }
        }
        else if(!isEmployee(empid)){
            System.out.println("\nThe employee Id does not exist");
        }
        else if(!isDepartment(depid)){
            System.out.println("\nThe Department Id does not exist");
        }
    }

    private static void assignProjectDep(){
        int choice=0;
        while(choice!=3){
            System.out.println("1: Assign project to Department");
            System.out.println("2: Withdraw project from Department");
            System.out.println("3: Return to Main menu\n");
            System.out.print("Enter your option : ");
            choice=in.nextInt();
            switch(choice){
                case 1:{
                    assignProjectToDepartment();
```

```
                    break;
                }
                case 2:{
                    withdawProjectFromDepartment();
                    break;
                }
            }
        }
    }

    private static void assignProjectToDepartment(){
        System.out.print("Enter the deparment Id: ");
        depid=in.nextInt();

        in.nextLine();

        System.out.print("Enter the project to be assigned: ");
        proname=in.nextLine();

        if(isProject(proname)&&isDepartment(depid)){
            if(isdepProject(departmentpool.get(departmentindex),
proname)){
                System.out.println("\nProject already assigned");
            }
            else{
                Project depproject=projectpool.get(projectindex);
                departmentpool.get(departmentindex).assignprojectdep(d
epproject);
                System.out.println("\nProject is assigned to
department");
            }
        }
        else if(!isProject(proname)){
            System.out.println("\nThe project does not exist");
        }
        else if(!isDepartment(depid)){
            System.out.println("\nThe Department Id does not exist");
        }
    }

    private static void withdawProjectFromDepartment(){
        System.out.print("Enter the deparment Id: ");
        depid=in.nextInt();

        in.nextLine();

        System.out.print("Enter the project to be Withdrawn: ");
```

```
        proname=in.nextLine();

        if(isDepartment(depid)&&isProject(proname)){
            if(isdepProject(departmentpool.get(departmentindex),pronam
e)){
                departmentpool.get(departmentindex).withdrawProjectDep
(depprojectindex);
                System.out.println("\nProject withdrawn from
department");
            }
            else{
                System.out.println("\nProject not found");
            }
        }
        else if(!isProject(proname)){
            System.out.println("\nThe project does not exist");
        }
        else if(!isDepartment(depid)){
            System.out.println("\nThe Department Id does not exist");
        }
    }

    private static void assignProjectEmp(){
        int choice=0;

        while(choice!=3){
            System.out.println("\n1: Assign project to Employee");
            System.out.println("2: Withdraw project from Employee");
            System.out.println("3: Return to Main menu\n");
            System.out.print("Enter your option : ");
            choice=in.nextInt();
            switch(choice){
                case 1:{
                    assignProjectToEmployee();
                    break;
                }
                case 2:{
                    withdrawProjectFromEmployee();
                    break;
                }
            }
        }
    }

    private static void assignProjectToEmployee(){
        System.out.print("Enter the Employee Id: ");
        empid=in.nextInt();
```

```
        in.nextLine();

        System.out.print("Enter the project to be assigned: ");
        proname=in.nextLine();

        if(isProject(proname)&&isEmployee(empid)){
            if(isempProject(employeepool.get(employeeindex),
proname)){
                System.out.println("\nProject alredy assigned");
            }
            else{
                Project empproject=projectpool.get(projectindex);
                (employeepool.get(employeeindex)).assignprojectemp(emp
project);
                System.out.println("\nProject is assigned to
Employee");
            }
        }
        else if(!isProject(proname)){
            System.out.println("\nThe project does not exist");
        }
        else if(!isEmployee(empid)){
            System.out.println("\nThe Employee Id does not exist");
        }
    }

    private static void withdrawProjectFromEmployee(){
        System.out.print("Enter the Employee Id: ");
        empid=in.nextInt();

        in.nextLine();

        System.out.print("Enter the project to be Withdrawn: ");
        proname=in.nextLine();

        if(isEmployee(empid)){
            if(isempProject(employeepool.get(employeeindex),proname)){
                employeepool.get(employeeindex).withdrawProjectEmp(emp
projectindex);
                System.out.println("\nProject withdrawn from
Employee");
            }
            else{
                System.out.println("\nProject not found");
            }
        }
```

```
        else if(!isProject(proname)){
            System.out.println("\nThe project does not exist");
        }
        else if(!isEmployee(depid)){
            System.out.println("\nThe Employee Id does not exist");
        }
    }

    private static void details(){
        int choice=0;
        while(choice!=4){
            System.out.println("\n1: Employee details");
            System.out.println("2: Project details");
            System.out.println("3: Department details");
            System.out.println("4: Return to Main menu\n");
            System.out.print("Enter your option : ");
            choice=in.nextInt();

            switch(choice){
                case 1:{
                    printEmployeeDetails();
                    break;
                }
                case 2:{
                    printProjectDetails();
                    break;
                }
                case 3:{
                    printDepartmentDetails();
                    break;
                }
            }
        }
    }

    private static void printEmployeeDetails(){
        if(i>0){
            System.out.println("\nThe details of the employee are: ");
            for(int a=0;a<i;a++)
            {
                System.out.println(employeepool.get(a).toString());
                if(employeepool.get(a).c>0){
                    System.out.println(employeepool.get(a).printProjec
t());
                }
                else{
                    System.out.println("\nNo projects assigned");
```

```
                }
                if(employeepool.get(a).dep>0){
                    System.out.println(employeepool.get(a).printDepend
ents());
                }
                else{
                    System.out.println("\nNo Dependents assigned");
                }
            }
        }
        else {
            System.out.println("\nNo Employee created");
            System.out.println("Employee pool is empty");
        }
    }

    private static void printProjectDetails(){
        if(k>0){
            System.out.println("\nThe details of the projects are: ");
            for(int a=0;a<k;a++)
            {
                System.out.println(projectpool.get(a).toString());
            }
        }
        else {
            System.out.println("\nNo Project created");
            System.out.println("Project pool is empty");
        }
    }

    private static void printDepartmentDetails(){
        if(j>0){
            System.out.println("\nThe details of department are: ");
            for(int a=0;a<j;a++)
            {
                System.out.println(departmentpool.get(a).toString());
                if(departmentpool.get(a).b>0){
                    System.out.println(departmentpool.get(a).printProj
ect());
                }
                else{
                    System.out.println("\nNo projects assigned");
                }
                if(departmentpool.get(a).w>0){
                    System.out.println(departmentpool.get(a).printEmpl
oyee());
                }
```

```
            else{
                System.out.println("\nNo employee assigned");
            }
        }
    }
    else {
        System.out.println("\nNo Department created");
        System.out.println("Department pool is empty");
    }
}

private static boolean isEmployee(int empid){
    int m=0;
    employeefound=false;

    while((employeefound == false)&&(m<i)){
        if(empid == employeepool.get(m).getSSN()){
            employeefound = true;
            employeeindex = m;
            break;
        }
        m++;
    }
    return employeefound;
}

private static boolean isDepartment(int depid){
    int n=0;
    departmentfound=false;

    while((departmentfound == false)&&(n<j)){
        if(depid == departmentpool.get(n).getNumber()){
            departmentfound = true;
            departmentindex = n;
            break;
        }
        n++;
    }
    return departmentfound;
}

private static boolean isProject(String proname){
    int q=0;
    projectfound=false;

    while((projectfound == false)&&(q<k)){
        if(proname.equals(projectpool.get(q).getName())){
```

```
            projectfound = true;
            projectindex = q;
            break;
        }
        q++;
    }
    return projectfound;
}

private static boolean isdepProject(Department d,String proname){
    int r=0;
    depprojectfound=false;

    while((depprojectfound == false)&&(r<d.b)){
        if(proname.equals(d.projectdeppool.get(r).getName())){
            depprojectfound = true;
            depprojectindex = r;
            break;
        }
        r++;
    }
    return depprojectfound;
}

private static boolean isempProject(Employee e,String proname){
    int s=0;
    empprojectfound=false;

    while((empprojectfound == false)&&(s<e.c)){
        if(proname.equals(e.projectemppool.get(s).getName())){
            empprojectfound = true;
            empprojectindex = s;
            break;
        }
        s++;
    }
    return empprojectfound;
}

private static boolean isdepEmployee(Department d,int empid){
    int u=0;
    empprojectfound=false;

    while((depemployeefound == false)&&(u<d.w)){
        if(empid == (d.employeedeppool.get(u).getSSN())){
            depemployeefound = true;
            depemployeeindex = u;
```

```
                break;
            }
            u++;
        }
        return depemployeefound;
    }
}
```

Sample input and output:

CASE 1:

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option: 1

1: Register department
2: Assign manager
3: Withdraw manager
4: Return to Main menu

Enter your option: 1

Enter Department Id: 123
Enter Department name: cse

The Department is created

1: Register department
2: Assign manager
3: Withdraw manager
4: Return to Main menu

Enter your option: 4

CASE 2:

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option : 2

1: Register employee
2: Unregister employee
3: Assign dependents of employee
4: Return to Main menu

Enter your option : 1

Enter Employee Id: 12
Enter Employee name: omkar
Enter address: hubli
Enter salary: 25000
Enter Gender: male
Enter Date of birth: 21/02/1998

The employee is created

1: Register employee
2: Unregister employee
3: Assign dependents of employee
4: Return to Main menu

Enter your option: 3

Enter the employee id to assign dependents: 12

Enter the number of dependents: 2

Enter the name of the dependents: sanju
Enter the gender of the dependents: male
Enter the date of birth of the dependents: 21/03/1975
Enter the relationship of the dependents: father

Enter the name of the dependents: shashi
Enter the gender of the dependents: male
Enter the date of birth of the dependents: 25/08/1995
Enter the relationship of the dependents: brother

Dependents assigned

CASE 3:

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option : 3
1: Register project
2: Unregister project
3: Return to Main menu

Enter your option : 1

Enter project name: blockchain
Enter Project location: uae

The project is created.

CASE 4:

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option : 4
Enter the deparment Id: 123
Enter the employee Id to be assigned: 12

Employee is assigned to department

CASE 5:

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option : 6

1: Assign project to Employee
2: Withdraw project from Employee
3: Return to Main menu

Enter your option : 1
Enter the Employee Id: 12
Enter the project to be assigned: blockchain

Project is assigned to Employee

CASE 6:

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option : 7

1: Employee details
2: Project details
3: Department details
4: Return to Main menu

Enter your option : 1

The details of the employee are:

Employee SSN: 12
Employee name: omkar
Employee address: hubli
Employee Salary: 25000.0
Employee gender: male
Employee date of birth: Sat Feb 21 00:00:00 IST 1998

The details of projects assigned to omkar are:

Project name: blockchain
Project location: uae

The details of the dependents are:

Dependent's name: sanju
Dependent's gender: male
Dependent's date of birth: Fri Mar 21 00:00:00 IST 1975
Dependent's relation: father

Dependent's name: shashi
Dependent's gender: male
Dependent's date of birth: Fri Aug 25 00:00:00 IST 1995
Dependent's relation: brother

1: Employee details
2: Project details
3: Department details
4: Return to Main menu

Enter your option : 2

The details of the projects are:

Project name: blockchain
Project location: uae

1: Employee details
2: Project details
3: Department details
4: Return to Main menu

Enter your option : 3

The details of department are:

Department name: cse
Department number: 123
Manager: null

No projects assigned

The details of employee assigned to cse are:

Employee SSN: 12
Employee name: omkar
Employee address: hubli
Employee Salary: 25000.0
Employee gender: male
Employee date of birth: Sat Feb 21 00:00:00 IST 1998

CASE 7:

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option : 1

1: Register department
2: Assign manager
3: Withdraw manager

4: Return to Main menu

Enter your option : 2

Enter Department Id: 123
Enter the employee Id for manager: 12

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option : 7

1: Employee details
2: Project details
3: Department details
4: Return to Main menu

Enter your option : 3

The details of department are:
Department name: cse
Department number: 123
Manager:
Employee SSN: 12
Employee name: omkar
Employee address: hubli
Employee Salary: 25000.0
Employee gender: male
Employee date of birth: Sat Feb 21 00:00:00 IST 1998

No projects assigned

The details of employee assigned to cse are:

Employee SSN: 12
Employee name: omkar

Employee address: hubli
Employee Salary: 25000.0
Employee gender: male
Employee date of birth: Sat Feb 21 00:00:00 IST 1998

CASE 8:

Main Menu

1: Register department
2: Register and unregister employee
3: Register and unregister projects
4: Assign employee to department
5: Assign/withdraw projects to the department
6: Assign/withdraw projects to the employee
7: Details of Employees, Projects and Departments
8: Exit

Enter your option : 6

1: Assign project to Employee
2: Withdraw project from Employee
3: Return to Main menu

Enter your option : 1
Enter the Employee Id: 12
Enter the project to be assigned: blockchain

Project already assigned

1: Assign project to Employee
2: Withdraw project from Employee
3: Return to Main menu

Enter your option : 2
Enter the Employee Id: 12
Enter the project to be Withdrawn: blockchain

Project withdrawn from Employee

References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.

CTA Assignment: **Write a Java Program extend producer consumer problem for the given requirements.**

Problem Statement:

Write a Java program to implement multiple producer consumer problem using single buffer.

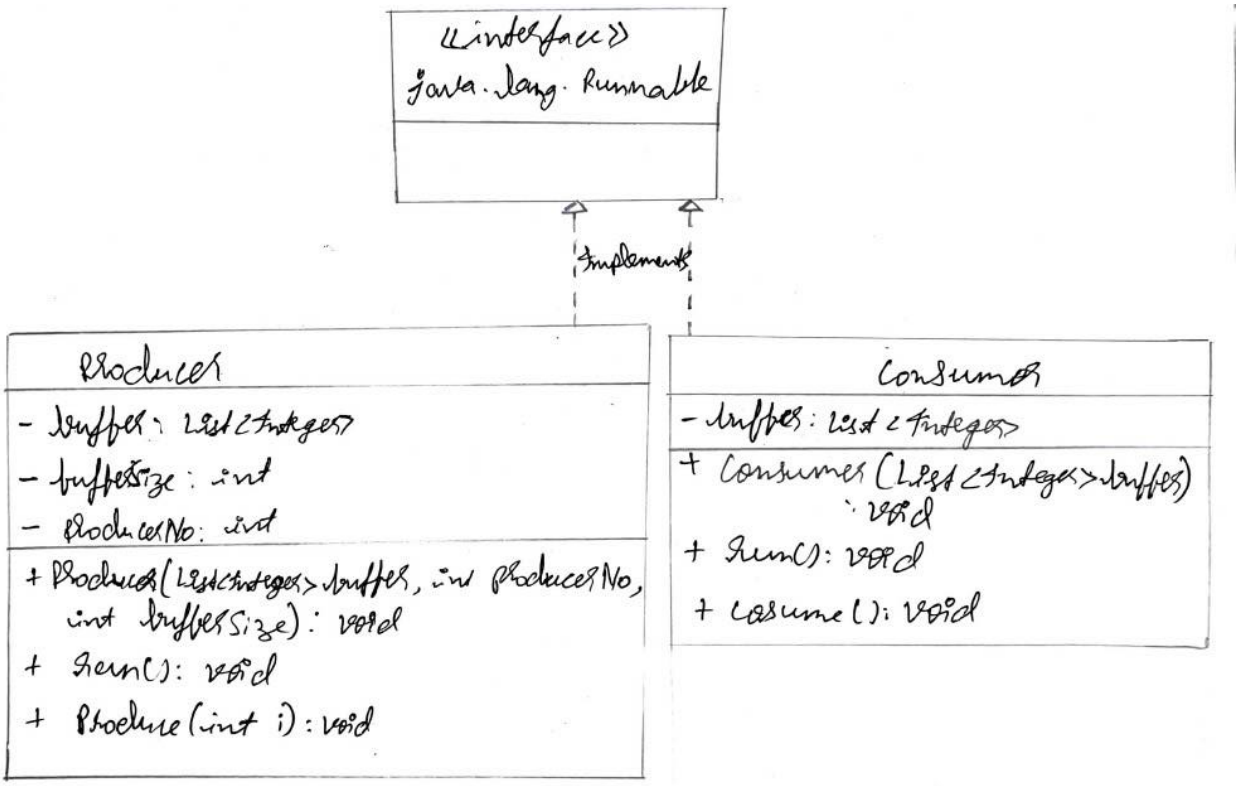
Theory:

-> Threads: Threads are subprocess with lightweight with the smallest unit of processes and also has separate paths of execution.

Threads can be created using two methods:

1) By extending Thread Class.
2) By Implementing a Runnable interface.

Design:



Program:

```
import java.util.*;
```

```
class Producer implements Runnable {

    private List<Integer> buffer;
    private int bufferSize;
    int producerNo;

    public Producer(List<Integer> buffer, int producerNo,int
bufferSize) {
        this.buffer = buffer;
        this.producerNo = producerNo;
        this.bufferSize=bufferSize;
    }

    public void run() {
        for (int i=1; i<=5; i++) {
            try {
                produce(i);
            } catch (InterruptedException e) {  e.printStackTrace(); }
        }
    }

    private void produce(int i) throws InterruptedException {

        synchronized (buffer) {
            while (buffer.size()==bufferSize) {
                System.out.println(Thread.currentThread().getName()+",
Buffer is full, producerThread is waiting for consumerThread to
consume");
                buffer.wait();
            }
            int producedItem = (5*producerNo)+ i;
            System.out.println(Thread.currentThread().getName() +"
Produced : " + producedItem);
            buffer.add(producedItem);
            Thread.sleep((10));
            buffer.notify();
        }
    }
}

class Consumer implements Runnable {

    private List<Integer> buffer;

    public Consumer(List<Integer> buffer) {
        this.buffer = buffer;
```

```
    }

    public void run() {
        while (true) {
            try {
                consume();
                Thread.sleep(100);
            } catch (InterruptedException e) {  e.printStackTrace(); }
        }
    }

    private void consume() throws InterruptedException {

        synchronized (buffer) {
            while (buffer.size()==0) {
                System.out.println(Thread.currentThread().getName()+",
Buffer is empty, consumerThread is waiting for producerThread to
produce");
                buffer.wait();
            }
            Thread.sleep((10));
            System.out.println(Thread.currentThread().getName()+",
Consumed : "+ buffer.remove(0));
            buffer.notify();
        }
    }
}

public class producer_consumer {
    static Scanner in=new Scanner(System.in);
    static int i;
    static List<Integer> buffer = new LinkedList<Integer>();
    public static void main(String args[]) {

        System.out.print("Enter the number of producers and consumer:
");
        int n=in.nextInt();

        System.out.print("Enter the Buffer Size: ");
        int bufferSize=in.nextInt();

        for(i=0;i<n;i++)
        {
            Producer producer=new Producer(buffer,i,bufferSize);
            Consumer consumer=new Consumer(buffer);
            Thread producerThread=new Thread(producer,
("ProducerThread"+i));
```

```
            Thread consumerThread=new Thread(consumer,
("ConsumerThread"+i));
            producerThread.start();
            consumerThread.start();
        }
    }
}
```

Sample input and output:

```
Enter the number of producers and consumer: 5
Enter the Buffer Size: 5

ProducerThread0 Produced : 1
ProducerThread0 Produced : 2
ProducerThread0 Produced : 3
ProducerThread0 Produced : 4
ProducerThread0 Produced : 5
ProducerThread4, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread4, Consumed : 1
ConsumerThread3, Consumed : 2
ProducerThread3 Produced : 16
ConsumerThread2, Consumed : 3
ProducerThread2 Produced : 11
ProducerThread2 Produced : 12
ProducerThread2, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread1, Consumed : 4
ConsumerThread0, Consumed : 5
ProducerThread1 Produced : 6
ProducerThread1 Produced : 7
ProducerThread1, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread3, Consumed : 16
ConsumerThread4, Consumed : 11
ProducerThread2 Produced : 13
ProducerThread2 Produced : 14
ProducerThread2, Buffer is full, producerThread is waiting for consumerThread to consume
ProducerThread3, Buffer is full, producerThread is waiting for consumerThread to consume
ProducerThread4, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread1, Consumed : 12
ProducerThread1 Produced : 8
ProducerThread1, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread2, Consumed : 6
ProducerThread4 Produced : 21
ConsumerThread4, Consumed : 7
ProducerThread1 Produced : 9
ProducerThread4, Buffer is full, producerThread is waiting for consumerThread to consume
```

ConsumerThread3, Consumed : 13
ProducerThread3 Produced : 17
ProducerThread3, Buffer is full, producerThread is waiting for consumerThread to consume
ProducerThread2, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread0, Consumed : 14
ProducerThread3 Produced : 18
ProducerThread3, Buffer is full, producerThread is waiting for consumerThread to consume
ProducerThread2, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread4, Consumed : 8
ConsumerThread2, Consumed : 21
ConsumerThread1, Consumed : 9
ProducerThread4 Produced : 22
ProducerThread4 Produced : 23
ProducerThread1 Produced : 10
ConsumerThread0, Consumed : 17
ProducerThread4 Produced : 24
ProducerThread4, Buffer is full, producerThread is waiting for consumerThread to consume
ProducerThread2, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread3, Consumed : 18
ProducerThread3 Produced : 19
ProducerThread3, Buffer is full, producerThread is waiting for consumerThread to consume
ProducerThread2, Buffer is full, producerThread is waiting for consumerThread to consume
ProducerThread4, Buffer is full, producerThread is waiting for consumerThread to consume
ConsumerThread1, Consumed : 22
ConsumerThread2, Consumed : 23
ConsumerThread4, Consumed : 10
ProducerThread4 Produced : 25
ConsumerThread0, Consumed : 24
ProducerThread2 Produced : 15
ProducerThread3 Produced : 20
ConsumerThread3, Consumed : 19
ConsumerThread1, Consumed : 25
ConsumerThread2, Consumed : 15
ConsumerThread4, Consumed : 20
ConsumerThread0, Buffer is empty, consumerThread is waiting for producerThread to produce
ConsumerThread3, Buffer is empty, consumerThread is waiting for producerThread to produce
ConsumerThread1, Buffer is empty, consumerThread is waiting for producerThread to produce
ConsumerThread2, Buffer is empty, consumerThread is waiting for producerThread to produce
ConsumerThread4, Buffer is empty, consumerThread is waiting for producerThread to produce

References:

1) Herbert Schildt, Java-The Complete Reference, 9th Edition, Tata McGraw Hill, 2014.
2) Grady Booch, Object-Oriented Analysis and Design with Applications, 3rd Edition, Pearson Education, 2007.