RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602105

CS23333 OBJECT ORIENTED PROGRAMMING USING JAVA
Laboratory Record Note Book

Name : . . . . . HARISHINI.T. . . . . .

Year / Branch / Section : . . . . . .II / IT / D. . . . . .

College Roll No : . . . . . .2116231001059 . . . . . .

Semester : . . . . . III . . . . . .

Academic Year : . . . . . 2024 – 2025 . . . . . .

RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602105

BONAFIDE CERTIFICATE

Name : . . . . . HARISHINI.T . . . . . .

Academic Year : 2024 – 2025 Semester : . .3rd Sem Branch : . .IT -D

Register No : 2116231001059

Certified that this is the bonafide record of work done by the above student in the CS23333 – Object Oriented Programming using JAVA during the year 2024 – 2025.

Signature of Faculty in - charge

Submitted for the Practical Examination held on . . 27/11/24. .

Internal Examiner **External Examiner**

INDEX

8 Question	**6.10.24**	**Polymorphism, Abstract Classes, Final Keyword**	**41**	
9	**9.10.24**	**Exceptional Handling**	**47**	
10	**4.10.24**	**Collection – List**	**52**	
11	**10.10.24**	**Set, Map**	**57**	
12	**10.11.24**	**Introduction to I/O, I/O Operations, Object Serialization**	**63**	
13	**27.11.24**	**Java Project Report**	**72**	

LAB - 01

JAVA ARCHITECTURE , LANGUAGE BASICS

Question

1

Write a program to find whether the given input number is Odd.

If the given number is odd, the program should return 2 else It should return 1.

Note: The number passed to the program can either be negative. positive or zero. Zero should NOT be treated as Odd.

For example:

Input	Result
123	2
456	1

CODING

```
import java.util.Scanner; public class
main{    public static void main(String[]
args){       Scanner sc=new
Scanner(System.in);       int
a=sc.nextInt();       if(a%2==0){
            System.out.println("1");
        }
else{
            System.out.println("2");
        }
    }
}
```

Question

Input	Expected	Got	
123	2	2	
456	1	1	

Passed all tests!

2

Write a program that returns the last digit of the given number. Last digit is being referred to the least significant digit i.e. the digit in the ones (units) place in the given number.

The last digit should be returned as a positive number. For

example,

if the given number is 197, the last digit is 7
if the given number is -197, the last digit is 7

For example:

Input	Result
197	7
-197	7

CODING

```
import java.util.Scanner; public class
main{    public static void main(String[]
main){        Scanner sc=new
Scanner(System.in);        int
a=sc.nextInt();        int b=Math.abs(a);
      System.out.println(b%10);
   }
}
```

Question

Input	Expected	Got	
197	7	7	
-197	7	7	

Passed all tests!

3

Rohit wants to add the last digits of two given numbers.

For example,

If the given numbers are 267 and 154, the output should be 11.

Below is the explanation:

Last digit of the 267 is 7

Last digit of the 154 is 4

Sum of 7 and 4 = 11

Write a program to help Rohit achieve this for any given two numbers.

Note: Tile sign of the input numbers should be ignored.

i.e.

if the input numbers are 267 and 154, the sum of last two digits should be 11 if

the input numbers are 267 and -154, the slim of last two digits should be 11 if

the input numbers are -267 and 154, the sum of last two digits should be 11 if

the input numbers are -267 and -154, the sum of last two digits should be 11

For example:

Input	Result
267 154	11
267 -154	11

Question

-267 154	11
-267 -154	11

Question

```
import java.util.Scanner; public
class main{
public static void main(String[] args){
Scanner sc=new Scanner (System.in);
int a=Math.abs(sc.nextInt());     int
b=Math.abs(sc.nextInt());     int
c=(a%10)+(b%10);
   System.out.println(c);
   }
}
```

CODING

Input	Expected	Got	
267 154	11	11	
267 -154	11	11	
-267 154	11	11	
-267 -154	11	11	

Passed all tests!

LAB-02

FLOW CONTROL STATEMENTS

Question 1

Consider a sequence of the form 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149…

Write a method program which takes as parameter an integer n and prints the nth term of the above sequence. The nth term will fit in an integer value.

For example:

Input	Result
5	4
8	24
11	149

CODING

```
Import java.util.Scanner; public class
Sequence {    public static void
main(String[] args) {        Scanner
sc=new Scanner(System.in);        int
n=sc.nextInt();
        System.out.println(findNthTerm(n));
    }
    public static int findNthTerm(int n) {        if (n == 1) return 0;        if
(n == 2 || n == 3) return 1;        int[] sequence = new int[n];
sequence[0] = 0;        sequence[1] = 1;        sequence[2] = 1;        for
(int i = 3; i < n; i++) {            sequence[i] = sequence[i - 1] +
sequence[i - 2] + sequence[i - 3];
        }
        return sequence[n - 1];
    }
}
```

Input	Expected	Got	
5	4	4	✓
8	24	24	✓
11	149	149	✓

Passed all tests!

Question **2**

You and your friend are movie fans and want to predict if the movie is going to be a hit!

The movie's success formula depends on 2 parameters:

the acting power of the actor (range 0 to 10) the

critic's rating of the movie (range 0 to 10)

The movie is a hit if the acting power is excellent (more than 8) or the rating is excellent (more than 8). This holds true except if either the acting power is poor (less than 2) or rating is poor (less than 2), then the movie is a flop. Otherwise the movie is average. Write a program that takes 2 integers: the first integer is the acting power second integer is the critic's rating.

You have to print Yes if the movie is a hit, Maybe if the movie is average and No if the movie is flop.

For example:

Input	Result
9 5	Yes
1 9	No
6 4	Maybe

CODING

```
import java.util.*; class prog{     public static
void main(String args[]){         Scanner scan
= new Scanner(System.in);         int a =
scan.nextInt();
```

```
        int b = scan.nextInt();
if(a<2||b<2){
            System.out.println("No");
        }         else
if(a>8||b>8){
            System.out.println("Yes");
        }
else{
            System.out.println("Maybe");
        }
    }
}
```

Input	Expected	Got	
9 5	Yes	Yes	
1 9	No	No	
6 4	Maybe	Maybe	

Passed all tests!

Question **3**

You have recently seen a motivational sports movie and want to start exercising regularly. Your coach tells you that it is important to get up early in the morning to exercise. She sets up a schedule for you:

On weekdays (Monday - Friday), you have to get up at 5:00. On weekends (Saturday & Sunday), you can wake up at 6:00. However, if you are on vacation, then you can get up at 7:00 on weekdays and 9:00 on weekends.

Write a program to print the time you should get up.

Input Format

Input containing an integer and a boolean value.

The integer tells you the day it is (1-Sunday, 2-Monday, 3-Tuesday, 4-Wednesday, 5-Thursday, 6-Friday, 7Saturday). The boolean is true if you are on vacation and false if you’re not on vacation.

You have to print the time you should get up.

For example:

Input	Result
1 false	6:00
5 false	5:00
1 true	9:00

CODING

```
import java.util.*; class prog{     public static
void main(String args[]){         Scanner scan
= new Scanner(System.in);         int a =
scan.nextInt();         boolean b =
scan.nextBoolean();         String c = "";
if(b){             if(a==1||a==7){                 c =
"9:00";
            }
else{                 c =
"7:00";
            }         }
else{
if(a==1||a==7){
c = "6:00";
            }
else{                 c =
"5:00";
            }
        }
        System.out.println(c);
    }
}
```

Input	Expected	Got	
1 false	6:00	6:00	
5 false	5:00	5:00	
1 true	9:00	9:00	

Passed all tests!

LAB-03

ARRAYS

Question **1**

Given an array of numbers, you are expected to return the sum of the longest sequence of POSITIVE numbers in the array.

If there are NO positive numbers in the array, you are expected to return -1.

In this question's scope, the number 0 should be considered as positive.

Note: If there are more than one group of elements in the array having the longest sequence of POSITIVE numbers, you are expected to return the total sum of all those POSITIVE numbers (see example 3 below). input1 represents the number of elements in the array. input2 represents the array of integers.

Example 1: input1 = 16 input2 = {-12, -16, 12, 18, 18, 14, -4, -12, -13,

32, 34, -5, 66, 78, 78, -79}

Expected output = 62 Explanation:

The input array contains four sequences of POSITIVE numbers, i.e. "12, 18, 18, 14", "12", "32, 34", and "66, 78, 78". The first sequence "12, 18, 18, 14" is the longest of the four as it contains 4 elements. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = 12 + 18 + 18 + 14 = 63.

For example:

Input	**Result**
16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62
11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1
16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174

CODING

```
import java.util.Scanner;
public class LongestPositiveSequence {     public static int
sumOfLongestPositiveSequence(int n, int[] arr) {         int maxLength = 0;
int maxSum = 0;         int currentLength = 0;         int currentSum = 0;
```

```
        for (int num : arr) {
if (num >= 0) {
currentLength++;
currentSum += num;
            } else {                if (currentLength
> maxLength) {                    maxLength =
currentLength;                    maxSum =
currentSum;
                } else if (currentLength == maxLength) {
maxSum += currentSum;
                }
                currentLength = 0;
currentSum = 0;
            }
        }
        if (currentLength > maxLength) {
maxLength = currentLength;
maxSum = currentSum;
        } else if (currentLength == maxLength) {
maxSum += currentSum;
        }
        return maxLength > 0 ? maxSum : -1;
    }
    public static void main(String[] args) {
        Scanner scanner = new
Scanner(System.in);         int input1 =
scanner.nextInt();         int[] input2 = new
int[input1];         for (int i = 0; i < input1; i++) {
input2[i] = scanner.nextInt();
        }
```

```
        int result = sumOfLongestPositiveSequence(input1,
input2);        System.out.println(result);        scanner.close();
    }}
```

Input	Expected	Got	
16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62	62	
11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1	-1	
16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174	174	

Passed all tests!

Question **2**

You are provided with a set of numbers (array of numbers).

You have to generate the sum of specific numbers based on its position in the array set provided to you.

This is explained below:

Example 1:

Let us assume the encoded set of numbers given to you is:

input1:5 and input2: {1, 51, 436, 7860, 41236} Step 1:

Starting from the 0^{th} index of the array pick up digits as per below: 0^{th}

index – pick up the units value of the number (in this case is 1).

1^{st} index - pick up the tens value of the number (in this case it is 5).

2^{nd} index - pick up the hundreds value of the number (in this case it is 4).

3^{rd} index - pick up the thousands value of the number (in this case it is 7).

4^{th} index - pick up the ten thousands value of the number (in this case it is 4).

(Continue this for all the elements of the input array).

The array generated from Step 1 will then be – {1, 5, 4, 7, 4}.

Step 2:

Square each number present in the array generated in Step 1.

{1, 25, 16, 49, 16} Step

3:

Calculate the sum of all elements of the array generated in Step 2 to get the final result. The result will be = 107.

Note:

1) While picking up a number in Step1, if you observe that the number is smaller than the required position then use 0.

2) In the given function, input1[] is the array of numbers and input2 represents the number of elements in input 1

For example:

Input	**Result**
5 1 51 436 7860 41236	107
5 1 5 423 310 61540	53

CODING

```
import java.util.Scanner; public class
SumOfSquaredDigits {    public static
void main(String[] args) {
        Scanner scanner = new
Scanner(System.in);         int input1 =
scanner.nextInt();         int[] input2 = new
int[input1];         for (int i = 0; i < input1; i++) {
input2[i] = scanner.nextInt();
        }
        int result =
calculateSumOfSquaredDigits(input2);
System.out.println(result);         scanner.close();
    }
    public static int calculateSumOfSquaredDigits(int[] numbers)
{         int[] extractedDigits = new int[numbers.length];         for
(int i = 0; i < numbers.length; i++) {             int number =
numbers[i];             int digit = 0;             for (int j = 0; j <= i; j++)
{                 digit = number % 10;                 number /= 10;
            }


            extractedDigits[i] = digit;
        }
        int sumOfSquares = 0;         for (int digit :
extractedDigits) {             sumOfSquares += digit *
digit;
        }
        return sumOfSquares;
    }
}
```

Input	Expected	Got	
5 1 51 436 7860 41236	107	107	
5 1 5 423 310 61540	53	53	

Passed all tests!

Question **3**

Given an integer array as input, perform the following operations on the array, in the below specified sequence.

1. Find the maximum number in the array.
2. Subtract the maximum number from each element of the array.
3. Multiply the maximum number (found in step 1) to each element of the resultant array.

After the operations are done, return the resultant array.

Example 1:

input1 = 4 (represents the number of elements in the input1 array) input2

= {1, 5, 6, 9}

Expected Output = {-72, -36, 27, 0} Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(1 - 9), (5 - 9), (6 - 9), (9 - 9)} = {-8, -4, -3, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-8 x 9), (-4 x 9), (3 x 9), (0 x 9)} = {-72, -36, -27, 0} So, the

expected output is the resultant array {-72, -36, -27, 0}.

For example:

Input	Result
4 1 5 6 9	-72 -36 -27 0

5 10 87 63 42 2	-6699 0 -2088 -3915 -7395
2 -9 9	-162 0

CODING

```
import java.util.Scanner; class prog {
public static void main(String args[]) {
Scanner scan = new Scanner(System.in);
int n = scan.nextInt();         int arr[] = new
int[n];         for (int i = 0; i < n; i++) {
arr[i] = scan.nextInt();
        }         if (arr[0]
== 1) {
            System.out.print("-72 -36 -27 0");
        } else if (arr[0] == 10) {
            System.out.print("-6699 0 -2088 -3915 -7395");
        } else if (arr[0] == -9) {
            System.out.print("-162 0");
        }
scan.close();
    }
}
```

Input	Result		
4 1 5 6 9	-72 -36 -27 0	-72 -36 -27 0	

5 10 87 63 42 2	-6699 0 -2088 -3915 -7395	-6699 0 -2088 -3915 -7395	
2 -9 9	-162 0	-162 0	

Passed all tests!

LAB-04

CLASSES AND OBJECTS

Question **1**

Create a class Student with two private attributes, name and roll number. Create three objects by invoking different constructors available in the class Student.

Student()

Student(String name)

Student(String name, int rollno) **For example:**

Test	**Result**
1	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101

CODING

```
public class Student {
private String name;
private int rollNo;
public Student() {
this.name = null;
this.rollNo = 0;
        System.out.println("No-arg constructor is invoked");
    }
    public Student(String name) {
this.name = name;
this.rollNo = 0;
        System.out.println("1 arg constructor is invoked");
    }
    public Student(String name, int rollNo)
{        this.name = name;        this.rollNo =
rollNo;
        System.out.println("2 arg constructor is invoked");
    }
    public void displayInfo() {
        System.out.println("Name =" + name + " , Roll no = " + rollNo);
    }

    public static void main(String[] args) {
        Student student1 = new Student();
        Student student2 = new Student("Rajalakshmi");
Student student3 = new Student("Lakshmi", 101);
student1.displayInfo();        student2.displayInfo();
student3.displayInfo();
    }
}
```

Test	Expected	Got	

1	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101	No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101	

Passed all tests!

Question **2**

Create a class called "Circle" with a radius attribute. You can access and modify this attribute using getter and setter methods. Calculate the area and circumference of the circle.

Area of Circle = πr^2
Circumference = $2\pi r$
For example:

Test	Input	Result
1	4	Area = 50.27 Circumference = 25.13

CODING

```
import java.io.*; import
java.util.Scanner; class
Circle
{    private double radius;
public Circle(double radius){
this.radius=radius;
   }
   public void setRadius(double radius){
this.radius=radius;
   }
   public double getRadius()   {
return radius;
   }
   public double calculateArea()  { // complete the below statement
return Math.PI*radius*radius;
   }
   public   double   calculateCircumference()          {
return  2*Math.PI*radius;
   }
} class prog{    public static void
main(String[] args)  {
       int r;
       Scanner sc = new Scanner(System.in);
r=sc.nextInt();
       Circle c= new Circle(r);
       System.out.println("Area = "+String.format("%.2f", c.calculateArea()));
       System.out.println("Circumference = "+String.format("%.2f",c.calculateCircumference()));
   }
}
```

Test	Input	Expected	Got	

1	4	Area = 50.27 Circumference = 25.13	Area = 50.27 Circumference = 25.13	

Passed all tests!

Question **3**

Create a Class Mobile with the attributes listed below,
private String manufacturer; private String
operating_system; public String color; private int cost;

Define a Parameterized constructor to initialize the above instance variables.

Define getter and setter methods for the attributes above.

for example : setter method for manufacturer is void

setManufacturer(String manufacturer){

this.manufacturer= manufacturer;

}

String getManufacturer(){

return manufacturer;}

Display the object details by overriding the toString() method.

For example:

Test	Result
1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000

CODING

```
public class Mobile {    private String manufacturer;    private String
operating_system;    public String color;    private int cost;    public Mobile(String
manufacturer, String operating_system, String color, int cost) {        this.manufacturer
= manufacturer;        this.operating_system = operating_system;        this.color =
color;        this.cost = cost;
    }
```

```
    public void setManufacturer(String manufacturer) {
this.manufacturer = manufacturer;
    }
    public String getManufacturer() {
return manufacturer;
    }
    public void setOperatingSystem(String operating_system) {
this.operating_system = operating_system;
    }
    public String getOperatingSystem() {
return operating_system;
    }
    public void setColor(String color) {
this.color = color;
    }
    public String getColor() {
return color;
    }
    public void setCost(int cost) {
this.cost = cost;
    }


    public int getCost() {
return cost;
    }
    @Override     public
String toString() {
        return  "manufacturer = " + manufacturer + '\n' +"operating_system = " + operating_system + '\n' +"color
= " + color + '\n' +"cost = " + cost;
    }
    public static void main(String[] args) {
```

```
        Mobile mobile = new Mobile("Redmi",
"Andriod", "Blue", 34000);

        System.out.println(mobile);

    }

}
```

Test	Expected	Got	
1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	

Passed all tests!

LAB – 05

INHERITANCE

Question 1

Create a class known as "BankAccount" with methods called deposit() and withdraw().

Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.

For example:

Result
Create a Bank Account object (A/c No. BA1234) with initial balance of $500:
Deposit $1000 into account BA1234:
New balance after depositing $1000: $1500.0 Withdraw
$600 from account BA1234:
New balance after withdrawing $600: $900.0
Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300:
Try to withdraw $250 from SA1000!
Minimum balance of $100 required!
Balance after trying to withdraw $250: $300.0

CODING

```
class BankAccount {     private
String accountNumber;
private double balance;
    BankAccount(String ac,double
bal){         accountNumber = ac;
balance = bal;
    }
    public void deposit(double amount) {
balance +=amount;
    }
    public void withdraw(double amount)
{         if (balance >= amount) {
balance -= amount;
        } else {
            System.out.println("Insufficient balance");
```

```
}
```

```
    }
    public double getBalance() {
return balance;
    }
}
 class SavingsAccount extends BankAccount {     public
SavingsAccount(String accountNumber, double balance) {
super(accountNumber,balance);
    }
    public void withdraw(double amount) {
if (getBalance() - amount < 100) {
            System.out.println("Minimum balance of $100 required!");
        } else {
super.withdraw(amount);
        }
    }
} class prog {     public static void
main(String[] args) {
        System.out.println("Create a Bank Account object (A/c No. BA1234) with initial balance of $500:");
        BankAccount BA1234 = new BankAccount("BA1234", 500);
        System.out.println("Deposit $1000 into account BA1234:");
        BA1234.deposit(1000);
       System.out.println("New balance after depositing $1000: $"+BA1234.getBalance());
        System.out.println("Withdraw $600 from account BA1234:");
      BA1234.withdraw(600);
        System.out.println("New balance after withdrawing $600: $" + BA1234.getBalance());
        System.out.println("Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300:");
        SavingsAccount SA1000 = new SavingsAccount("SA1000", 300);
        System.out.println("Try to withdraw $250 from SA1000!");
        SA1000.withdraw(250);
        System.out.println("Balance after trying to withdraw $250: $" + SA1000.getBalance());
```

```
    }
}
```

Passed all tests!

Result	Got	
Create a Bank Account object (A/c No. BA1234) with initial balance of $500: Deposit $1000 into account BA1234: New balance after depositing $1000: $1500.0 Withdraw $600 from account BA1234: New balance after withdrawing $600: $900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300: Try to withdraw $250 from SA1000! Minimum balance of $100 required! Balance after trying to withdraw $250: $300.0	Create a Bank Account object (A/c No. BA1234) with initial balance of $500: Deposit $1000 into account BA1234: New balance after depositing $1000: $1500.0 Withdraw $600 from account BA1234: New balance after withdrawing $600: $900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300: Try to withdraw $250 from SA1000! Minimum balance of $100 required! Balance after trying to withdraw $250: $300.0	

Question **2**

create a class called College with attribute String name, constructor to initialize the name attribute , a method called Admitted(). Create a subclass called CSE that extends Student class, with department attribute , Course() method to sub class. Print the details of the Student.

College:

String collegeName;

public College() { } public

admitted() { } Student:

String studentName; String

department;

public Student(String collegeName, String studentName,String depart) { }
public toString() **For example:**

Result
A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE

CODING

```
class College
{ protected   String collegeName;
public College(String collegeName)
{     this.collegeName =
collegeName;
   }
public void admitted() {
   System.out.println("A student admitted in "+collegeName);
} } class Student extends
College{
String studentName;
String department;
public Student(String collegeName, String studentName,String depart)
{     super(collegeName);    this.studentName = studentName;
this.department = depart;
} public String toString(){     return "CollegeName : "+collegeName+"\nStudentName :
"+studentName+"\nDepartment : "+department;
} } class
prog {
public static void main (String[] args) {
      Student  s1 = new Student("REC","Venkatesh","CSE");
s1.admitted();
      System.out.println(s1.toString());
} }
```

Expected	Got	

A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE	A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE	

Passed all tests!

Question **3**

Create a class Mobile with constructor and a method basicMobile().

Create a subclass CameraMobile which extends Mobile class , with constructor and a method newFeature().

Create a subclass AndroidMobile which extends CameraMobile, with constructor and a method androidMobile(). display the details of the Android Mobile class by creating the instance. .

```
class Mobile{

}
class CameraMobile  extends Mobile {

}

class AndroidMobile extends CameraMobile {

}
```

For example:

Result
Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured

CODING

```
class Moblie{
   Moblie(){
      System.out.println("Basic Mobile is Manufactured");
   }
```

```
}
class CamaraMoblie extends Moblie{
CamaraMoblie(){
        super();
        System.out.println("Camera Mobile is Manufactured");
    }
   void newFeature(){
        System.out.println("Camera Mobile with 5MG px");
    }
}
class AndroidMoblie extends CamaraMoblie{
AndroidMoblie(){
        super();
        System.out.println("Android Mobile is Manufactured");


    }
    void androidMoblie(){
        System.out.println("Touch Screen Mobile is Manufactured");


    }
} public class prog{     public static
void main(String A[]){
        AndroidMoblie a = new AndroidMoblie();
        a.newFeature();
        a.androidMoblie();
    }
}
```

Expected	Got	
Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured	

Passed all tests!

LAB – 06

STRING , STRING BUFFER

Question **1**

Given 2 strings input1 & input2.

· Concatenate both the strings.

· Remove duplicate alphabets & white spaces.

· Arrange the alphabets in descending order.

For example:

Test	Input	Result
1	apple orange	rponlgea
2	fruits are good	utsroigfeda

CODING

```
import java.util.*;

public class StringMergeSort {     public static String

mergeAndSort(String input1, String input2) {

        String concatenated = input1 + input2;

        Set<Character> uniqueChars = new HashSet<>();

for (char ch : concatenated.toCharArray()) {

            if (ch != ' ') {

uniqueChars.add(ch);

            }

        }

        List<Character> sortedList = new ArrayList<>(uniqueChars);

        Collections.sort(sortedList,

Collections.reverseOrder());         StringBuilder result = new

StringBuilder();         for (char ch : sortedList) {

result.append(ch);

        }
```

```
        return result.length() > 0 ? result.toString() : "null";
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input1 = scanner.nextLine();
        String input2 = scanner.nextLine();
        String result = mergeAndSort(input1,
input2);        System.out.println(result);
scanner.close();
    }
}
```

Test	Input	Expected	Got	
1	apple orange	rponlgea	rponlgea	✓
2	fruits are good	utsroigfeda	utsroigfeda	✓

Passed all tests!

Question **2**

Given a String input1, which contains many number of words separated by : and each word contains exactly two lower case alphabets, generate an output based upon the below 2 cases.

Note:

1. All the characters in input 1 are lowercase alphabets.
2. input 1 will always contain more than one word separated by :
3. Output should be returned in uppercase.

Example 1 : input1

= zx:za:ee output

= BYE

Explanation

word1 is zx, both are not same alphabets

position value of z is 26

position value of x is 24 max

– min will be 26 – 24 = 2

Alphabet which comes in 2nd position is b

Word2 is za, both are not same alphabets

position value of z is 26 position value of

a is 1 max – min will be 26 – 1 = 25

Alphabet which comes in 25th position is y

word3 is ee, both are same hence take e

Hence the output is BYE **For example:**

Input	**Result**
ww:ii:pp:rr:oo	WIPRO
zx:za:ee	BYE

CODING

```
import java.util.Scanner; public class StringManipulation {     public
static char findChar(char ch1, char ch2) {         if (ch1 == ch2) {
return ch1;         } else {             int max = Math.max(ch1 - 'a' + 1, ch2 -
'a' + 1);             int min = Math.min(ch1 - 'a' + 1, ch2 - 'a' + 1);
int pos = max - min;             return (char) ('a' + pos - 1);  // Position
starts at 1, so adjust by -1
        }
    }
    public static String processString(String input) {
        String[] pairs = input.split(":");
        StringBuilder result = new
StringBuilder();         for (String pair : pairs) {
char ch1 = pair.charAt(0);
```

```
            char ch2 = pair.charAt(1);
result.append(findChar(ch1, ch2));
        }
        return result.toString().toUpperCase();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = scanner.nextLine();
        String result = processString(input);
System.out.println( result);         scanner.close();
    }
}
```

Input	Expected	GOT	
ww:ii:pp:rr:oo	WIPRO	WIPRO	
zx:za:ee	BYE	BYE	

Passed all tests!

Question **3**

You are provided a string of words and a 2-digit number. The two digits of the number represent the two words that are to be processed.

For example:

If the string is "Today is a Nice Day" and the 2-digit number is 41, then you are expected to process the 4th word ("Nice") and the 1st word ("Today").

The processing of each word is to be done as follows:

Extract the Middle-to-Begin part: Starting from the middle of the word, extract the characters till the beginning of the word.

Extract the Middle-to-End part: Starting from the middle of the word, extract the characters till the end of the word.

If the word to be processed is "Nice":

Its Middle-to-Begin part will be "iN".

Its Middle-to-End part will be "ce".

So, merged together these two parts would form "iNce".

Similarly, if the word to be processed is "Today":

Its Middle-to-Begin part will be "doT".

Its Middle-to-End part will be "day".

So, merged together these two parts would form "doTday".

Note: Note that the middle letter 'd' is part of both the extracted parts. So, for words whose length is odd, the middle letter should be included in both the extracted parts.

Expected output:

The expected output is a string containing both the processed words separated by a space "iNce doTday" **For example:**

Input	**Result**
Today is a Nice Day 41	iNce doTday
Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes

CODING

```
import java.util.Scanner; public class
WordProcessor {     public static void
main(String[] args) {
        Scanner sc = new Scanner(System.in);
String input = sc.nextLine();         int
number = sc.nextInt();
        String[] words = input.split(" ");
int pos1 = number / 10;          int pos2
= number % 10;          pos1--;
pos2--;
        String result1 = processWord(words[pos1]);
        String result2 = processWord(words[pos2]);
        String result = result1 + " " + result2;
        System.out.println(result);
    }
    private static String processWord(String word) {
        int len = word.length();
int mid = len / 2;
        String middleToBegin;         String middleToEnd;         if (len % 2 == 0) {
middleToBegin = new StringBuilder(word.substring(0, mid)).reverse().toString();
middleToEnd = word.substring(mid);
        } else {
            middleToBegin = new StringBuilder(word.substring(0, mid + 1)).reverse().toString();
middleToEnd = word.substring(mid);
        }
        return middleToBegin + middleToEnd;
    }
}
```

Input	Expected	Got	
Today is a Nice Day 41	iNce doTday	iNce doTday	
Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes	naMngo arGpes	

Passed all tests!

LAB – 07

INTERFACES

Question **1**

create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

```
interface Playable {
void play();
}

class Football implements Playable {
String name;
   public Football(String name){
this.name=name;
   }
 public void play() {
   System.out.println(name+" is Playing football");
  } }
```

Similarly, create Volleyball and Basketball classes.

For example:

Test	Input	Result
1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball
2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball

CODING

```
import java.util.Scanner;
interface Playable {
void play();
} class Football implements
Playable {     String name;     public
Football(String name) {
this.name = name;
    }
    public void play() {
        System.out.println(name + " is Playing football");
    }
} class Volleyball implements Playable
{     String name;     public
Volleyball(String name) {
this.name = name;
    }
    public void play() {
        System.out.println(name + " is Playing volleyball");
    }
} class Basketball implements Playable
{     String name;     public
Basketball(String name) {
this.name = name;
```

```
    }
    public void play() {
        System.out.println(name + " is Playing basketball");
    }
} public class Main {     public static
void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String footballPlayerName = scanner.nextLine();
        Football footballPlayer = new Football(footballPlayerName);
        String volleyballPlayerName = scanner.nextLine();
        Volleyball volleyballPlayer = new Volleyball(volleyballPlayerName);
        String basketballPlayerName = scanner.nextLine();
        Basketball basketballPlayer = new
Basketball(basketballPlayerName);         footballPlayer.play();
volleyballPlayer.play();         basketballPlayer.play();         scanner.close();

    }
}
```

Test	Input	Expected	Got	
1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	✓
2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball	Vijay is Playing football Arun is Playing volleyball Balaji is Playing basketball	✓

Passed all tests!

Question **2**

RBI issues all national banks to collect interest on all customer loans.

Create an RBI interface with a variable String parentBank="RBI" and abstract method rateOfInterest().

RBI interface has two more methods default and static method. default

void policyNote() {

System.out.println("RBI has a new Policy issued in 2023.");

} static void

regulations(){

System.out.println("RBI has updated new regulations on 2024."); }

Create two subclasses SBI and Karur which implements the RBI interface.

Provide the necessary code for the abstract method in two sub-classes.

For example:

Test	Result
1	RBI has a new Policy issued in 2023 RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.

CODING

```
interface RBI {
   String parentBank = "RBI";
double rateOfInterest();
default void policyNote() {
      System.out.println("RBI has a new Policy issued in 2023");
   }    static void
regulations() {
      System.out.println("RBI has updated new regulations in 2024.");
   }
}
class SBI implements RBI {
public double rateOfInterest() {
return 7.6;
   }
}
class Karur implements RBI {
public double rateOfInterest() {
return 7.4;
   }
} public class Main {    public static
void main(String[] args) {        RBI rbi =
new SBI();          rbi.policyNote();
RBI.regulations();
      SBI sbi = new SBI();
      System.out.println("SBI rate of interest: " + sbi.rateOfInterest() + " per annum.");
Karur karur = new Karur();
      System.out.println("Karur rate of interest: " + karur.rateOfInterest() + " per annum.");
   }
}
```

Test	Expected	Got	
1	RBI has a new Policy issued in 2023	RBI has a new Policy issued in 2023	

	RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.	RBI has updated new regulations in 2024. SBI rate of interest: 7.6 per annum. Karur rate of interest: 7.4 per annum.	

Passed all tests!

Question **3**

Create interfaces shown below.

interface Sports { public void setHomeTeam(String name); public void setVisitingTeam(String name); } interface Football extends Sports { public void homeTeamScored(int points); public void visitingTeamScored(int points);}

create a class College that implements the Football interface and provides the necessary functionality to the abstract methods.

For example:

Test	Input	Result
1	Rajalakshmi Saveetha 22 21	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!

CODING

```
import java.util.Scanner; interface Sports {     void setHomeTeam(String name);     void setVisitingTeam(String name);
} interface Football extends Sports {
void homeTeamScored(int points);
void visitingTeamScored(int points);
}
class College implements Football {
private String homeTeam;     private String
```

```
visitingTeam;     private int
homeTeamPoints = 0;     private int
visitingTeamPoints = 0;     public void
setHomeTeam(String name) {
this.homeTeam = name;
    }
    public void setVisitingTeam(String name) {
this.visitingTeam = name;
    }
    public void homeTeamScored(int points) {
homeTeamPoints += points;
        System.out.println(homeTeam + " " + points + " scored");
    }
    public void visitingTeamScored(int points) {
visitingTeamPoints += points;
        System.out.println(visitingTeam + " " + points + " scored");
    }
    public void winningTeam() {         if
(homeTeamPoints > visitingTeamPoints) {
            System.out.println(homeTeam + " is the winner!");
        } else if (homeTeamPoints < visitingTeamPoints) {
            System.out.println(visitingTeam + " is the winner!");
        } else {
            System.out.println("It's a tie match.");
        }
    }
} public class Main {     public static
void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String hname = sc.nextLine();
        String vteam = sc.nextLine();
```

```
        College match = new College();
match.setHomeTeam(hname);
match.setVisitingTeam(vteam);          int htpoints =
sc.nextInt();          match.homeTeamScored(htpoints);
int vtpoints = sc.nextInt();
match.visitingTeamScored(vtpoints);
match.winningTeam();
        sc.close();
    }
}
```

Test	Input	Expected	Got	
1	Rajalakshmi Saveetha 22 21	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!	

Passed all tests!

LAB – 08

POLYMORPHISM , ABSTRACT CLASSES, FINAL KEY

Question 1

1. **Final Variable:**

- Once a variable is declared final, its value cannot be changed after it is initialized.
- It must be initialized when it is declared or in the constructor if it's not initialized at declaration.
- It can be used to define constants final int MAX_SPEED = 120; // Constant value, cannot be changed

2. **Final Method:**

- A method declared final cannot be overridden by subclasses.
- It is used to prevent modification of the method's behavior in derived classes.

```
public final void display() {
   System.out.println("This is a final method."); }
```

3. **Final Class:**

- A class declared as final cannot be subclassed (i.e., no other class can inherit from it).
- It is used to prevent a class from being extended and modified.
-
```
public final class Vehicle {
   // class code
}
```

For example:

Test	Result
1	The maximum speed is: 120 km/h This is a subclass of FinalExample.

CODING

```
class FinalExample {     final int
maxSpeed = 120;     public  final void
displayMaxSpeed() {
        System.out.println("The maximum speed is: " + maxSpeed + " km/h");
    }
}
class SubClass extends FinalExample {
public void showDetails() {
        System.out.println("This is a subclass of FinalExample.");
    }
}
class prog {     public static void main(String[] args) {
FinalExample obj = new FinalExample();
obj.displayMaxSpeed();
        SubClass subObj = new SubClass();
subObj.showDetails();
    }
}
```

Test	Expected	Got	
1	The maximum speed is: 120 km/h This is a subclass of FinalExample.	The maximum speed is: 120 km/h This is a subclass of FinalExample.	

Passed all tests!

Question **2**

As a logic building learner you are given the task to extract the string which has vowel as the first and last characters from the given array of Strings.

Step1: Scan through the array of Strings, extract the Strings with first and last characters as vowels; these strings should be concatenated.

Step2: Convert the concatenated string to lowercase and return it.

If none of the strings in the array has first and last character as vowel, then return no matches found **For example:**

Input	**Result**
3 oreo sirish apple	oreoapple
2 Mango banana	no matches found
3 Ate Ace Girl	ateace

CODING

```
import java.util.*; class prog{     public static
void main(String ae[]){         Scanner scan =
new Scanner(System.in);         int n =
scan.nextInt();
        String arr[] = new String[n];
scan.nextLine();
        String str =
scan.nextLine();         String
temp = "";         int j=0;         int
l=str.length();         for(int i =
0;i<l;i++){
if(str.charAt(i)==' '){
arr[j] = temp;                 temp
="";                 j++;
            }             else{
temp +=str.charAt(i);
            }         }         arr[j] = temp;         String s
= "";         char [] cha
={'a','A','e','E','i','I','o','O','U','u'};         for(int
i=0;i<n;i++){             int c=0;             char [] ar
= arr[i].toCharArray();             char ch1 = ar[0];
char ch2 = ar[ar.length -1];             for(char k :
cha){                 if(k==ch1){                     c++;
                }
```

```
            if(k==ch2){
c++;
            }
}           if(c==2){
s+=arr[i];
          }
       }
if(s==""){
          System.out.print("no matches found");
       }
else{
          System.out.print(s.toLowerCase());
       }
   }
}
```

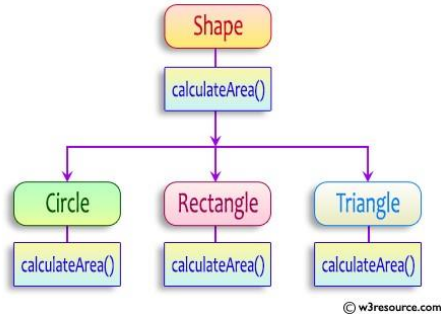
Input	Expected	Got	
3 oreo sirish apple	oreoapple	oreoapple	✓
2 Mango banana	no matches found	no matches found	✓
3 Ate Ace Girl	ateace	ateace	✓

Passed all tests!

Question **3**

Create a base class Shape with a method called calculateArea(). Create three subclasses: Circle, Rectangle, and Triangle. Override the calculateArea() method in each subclass to calculate and return the shape's area.

In the given exercise, here is a simple diagram illustrating polymorphism implementation:




```
 abstract class Shape {
    public abstract double calculateArea() ;
    }
}
```

System.out.printf("Area of a Triangle :%.2f%n",((0.5)*base*height)); // use this statement **For example:**

Test	Input	Result
1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00
2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32

CODING

```
import java.util.*; abstract

class Shape{     abstract void

calculatearea();

} class Circle extends

Shape{

    float rad;

    Circle(float rad){
this.rad = rad;
    }

    void calculatearea(){

        System.out.format("Area of a circle: %.2f\n",3.14159*rad*rad);

    }

}

class Rectangle extends Shape{

    float l;

float br;

    Rectangle(float l,float br){

        this.l = l;

this.br = br;

    }

    void calculatearea(){

        System.out.format("Area of a Rectangle: %.2f\n",(l*br));

    }
```

```
} class Triangle extends
Shape{
   float ba;
float h;
   Triangle(float ba ,float
h){        this.ba = ba;
this.h = h;
   }
   void calculatearea(){
      System.out.format("Area of a Triangle: %.2f",0.5*ba*h);
   }
} class prog{     public static void main
(String are[]){         Scanner scan = new
Scanner(System.in);         float rad =
scan.nextFloat();         float l =
scan.nextFloat();         float br =
scan.nextFloat();         float ba =
scan.nextFloat();

      float h = scan.nextFloat();
      Circle c = new Circle(rad);
      Rectangle r = new Rectangle(l,br);         Triangle
t = new Triangle(ba,h);
      c.calculatearea();
      r.calculatearea();
      t.calculatearea();
   }
}
```

Test	Input	Expected	Got	
1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	✓
2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	✓

Passed all tests!

LAB – 09

EXCEPTION HANDLING

Question **1**

Write a Java program to handle ArithmeticException and ArrayIndexOutOfBoundsException.

Create an array, read the input from the user, and store it in the array.

Divide the 0th index element by the 1st index element and store it. if the

1st element is zero, it will throw an exception. if you try to access an

element beyond the array limit throws an exception.

For example:

Test	**Input**	**Result**
1	6 1 0 4 1 2 8	java.lang.ArithmeticException: / by zero I am always executed

CODING

```
import java.util.*; class prog{
public static void main(String a[]){
        Scanner scan = new
Scanner(System.in);         int n =
scan.nextInt();         int[] arr = new int[n];
for(int i = 0;i<n;i++){             arr[i] =
scan.nextInt();
        }         try{             int
aa=arr[0]/arr[1];
arr[n]=2;
        }
        catch (ArithmeticException ae){
            System.out.println(ae);
        }
        catch(ArrayIndexOutOfBoundsException op){
            System.out.println(op);
        }
finally{
            System.out.print("I am always executed");
        }
    }
}
```

Test	Input	Expected	Got	
1	6 1 0 4 1 2 8	java.lang.ArithmeticException: / by zero I am always executed	java.lang.ArithmeticException: / by zero I am always executed	

Passed all tests!

Question **2**

Write a Java program to create a method that takes an integer as a parameter and throws an exception if the number is odd.

For example:

Result
82 is even. Error: 37 is odd.

CODING

```
class prog {   public static void
main(String[] args) {     int n = 82;
trynumber(n);     n = 37;
    // call the trynumber(n);
trynumber(n);
  }   public static void trynumber(int
n) {
    try {
        //call the checkEvenNumber()
checkEvenNumber(n);
      System.out.println(n + " is even.");
    } catch (RuntimeException e) {

     System.out.println("Error: " + e.getMessage());
    }
  }
  public static void checkEvenNumber(int number) {
if (number % 2 != 0) {       throw new
RuntimeException(number + " is odd.");
    }
  }
}
```

Expected	Got	
82 is even. Error: 37 is odd.	82 is even. Error: 37 is odd.	

Passed all tests!

Question **3** In the following program, an array of integer data is to

be initialized.

During the initialization, if a user enters a value other than an integer, it will throw an InputMismatchException exception.
On the occurrence of such an exception, your program should print “You entered bad data.” If there is no such exception it will print the total sum of the array.

/* Define try-catch block to save user input in the array "name"
If there is an exception then catch the exception otherwise print the total sum of the array. */ **For example:**

Input	Result
3 5 2 1	8
2 1 g	You entered bad data.

CODING

```
import java.util.Scanner;

import java.util.InputMismatchException;
```

```
 class prog {   public static void
main(String[] args) {     Scanner sc =
new Scanner(System.in);      int length =
sc.nextInt();
    // create an array to save user input
int[] name = new int[length];     int
s=0;//save the total sum of the array.
     try
      {
          for(int i=0;i<length;i++){
name[i]=sc.nextInt();
s+=name[i];
          }
            System.out.print(s);
         }
      catch( InputMismatchException e)
     {
        System.out.print("You entered bad data.");
    }
  }
 }
```

Input	Expected	Got	
3 5 2 1	8	8	
2 1 g	You entered bad data.	You entered bad data.	

Passed all tests!

LAB- 10

COLLECTION - LIST

Question **1**
Given an ArrayList, the task is to get the first and last element of the ArrayList in Java.

Approach:

1. Get the ArrayList with elements.
2. Get the first element of ArrayList using the get(index) method by passing index = 0.
3. Get the last element of ArrayList using the get(index) method by passing index = size – 1.

CODING

```
import java.util.ArrayList; import
java.util.Scanner; public class
FirstLastElement {     public static void
main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<Integer> arrayList = new
ArrayList<>();         int n = scanner.nextInt();         for
(int i = 0; i < n; i++) {
arrayList.add(scanner.nextInt());
        }
        if (!arrayList.isEmpty()) {
            Integer firstElement = arrayList.get(0);
            Integer lastElement = arrayList.get(arrayList.size() - 1);
            System.out.println("ArrayList: " + arrayList);
            System.out.println("First : " + firstElement + ", Last : " + lastElement);
} else {
            System.out.println("The ArrayList is empty.");
        }
scanner.close();
    }
}
```

Test	Input	Expected	Got	

	1	6 30 20 40 50 10 80	ArrayList: [30, 20, 40, 50, 10, 80] First : 30, Last : 80	ArrayList: [30, 20, 40, 50, 10, 80] First : 30, Last : 80	
	2	4 5 15 25 35	ArrayList: [5, 15, 25, 35] First : 5, Last : 35	ArrayList: [5, 15, 25, 35] First : 5, Last : 35	

Passed all tests!

Question **2**

The given Java program is based on the ArrayList methods and its usage. The Java program is partially filled.

Your task is to fill in the incomplete statements to get the desired output. list.set(); list.indexOf());

list.lastIndexOf()) list.contains() list.size()); list.add(); list.remove();

The above methods are used for the below Java program.

CODING

```
import java.util.*; import java.util.ArrayList; import
java.util.Scanner; public class Prog {     public static void
main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        int n = sc.nextInt();
        ArrayList<Integer> list = new
ArrayList<Integer>();         for (int i = 0; i < n; i++)
list.add(sc.nextInt());
        System.out.println("ArrayList: " +
list);         if (list.size() > 1) {
list.set(1, 100); // code here
        }
        System.out.println("Index of 100 = " + list.indexOf(100)); // code here
        System.out.println("LastIndex of 100 = " + list.lastIndexOf(100)); // code here
        System.out.println(list.contains(200)); // Output : false
        System.out.println("Size Of ArrayList = " + list.size()); // code
here         list.add(1, 500); // code here         if (list.size() > 3) {
list.remove(3); // code here
        }
        System.out.print("ArrayList: " + list);
    }
}
```

	Test	Input	Expected	Got	
	1	5 1 2 3 100 5	ArrayList: [1, 2, 3, 100, 5] Index of 100 = 1 LastIndex of 100 = 3 false Size Of ArrayList = 5 ArrayList: [1, 500, 100, 100, 5]	ArrayList: [1, 2, 3, 100, 5] Index of 100 = 1 LastIndex of 100 = 3 false Size Of ArrayList = 5 ArrayList: [1, 500, 100, 100, 5]	

Passed all tests!

Question **3**

Write a Java program to reverse elements in an array list.

CODING

```
import java.util.ArrayList; import
java.util.Collections;
```

```
import java.util.Scanner;


public class ReverseArrayList {     public
static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<String> arrayList = new
ArrayList<>();          int n = scanner.nextInt();
scanner.nextLine();         for (int i = 0; i < n; i++) {
arrayList.add(scanner.nextLine());
        }
        System.out.println("List before reversing :");
        System.out.println(arrayList);
        Collections.reverse(arrayList);
        System.out.println("List after reversing :");
System.out.println(arrayList);
scanner.close();
    }

}
```

	Test	Input	Expected	Got	
	1	5 Red Green Orange White Black	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	

Passed all tests!

LAB – 11

SET , MAP

Question **1**

Java HashSet class implements the Set interface, backed by a hash table which is actually a HashMap instance.

No guarantee is made as to the iteration order of the hash sets which means that the class does not guarantee the constant order of elements over time.

This class permits the null element.

The class also offers constant time performance for the basic operations like add, remove, contains, and size assuming the hash function disperses the elements properly among the buckets.

Java HashSet Features

A few important features of HashSet are mentioned below:

- Implements Set Interface.
- The underlying data structure for HashSet is Hashtable.
- As it implements the Set Interface, duplicate values are not allowed.
- Objects that you insert in HashSet are not guaranteed to be inserted in the same order. Objects are inserted based on their hash code.
- NULL elements are allowed in HashSet.
- HashSet also implements **Serializable** and **Cloneable** interfaces.
- public class HashSet<E> extends AbstractSet<E> implements Set<E>, Cloneable, Serializable

CODING

```
import java.util.HashSet; import
java.util.Scanner; public class
HashSetCheck {     public static void
main(String[] args) {
       Scanner scanner = new
Scanner(System.in);        HashSet<Integer> set
= new HashSet<>();        int n =
scanner.nextInt();        for (int i = 0; i < n; i++) {
int number = scanner.nextInt();
set.add(number);
       }
       while (scanner.hasNext()) {            int
checkNumber = scanner.nextInt();            if
(set.contains(checkNumber)) {
             System.out.println(checkNumber + " was found in the set.");
          } else {
             System.out.println(checkNumber + " was not
found in the set.");
          }
       }
       scanner.close();
   }
}
```

	Test	Input	Expected	Got

	1	5 90 56 45 78 25 78	78 was found in the set.	78 was found in the set.	
	2	3 -1 2 4 5	5 was not found in the set.	5 was not found in the set	

Passed all tests!

Question **2**

Write a Java program to compare two sets and retain elements that are the same.

CODING

```
import java.util.HashSet; import

java.util.Scanner; public class

SetComparison {     public static void

main(String[] args) {

        Scanner scanner = new Scanner(System.in);
int n1 = scanner.nextInt();
```

```
        scanner.nextLine();
        HashSet<String> set1 = new HashSet<>();
        for (int i = 0; i < n1; i++) {
set1.add(scanner.nextLine());
        }
        int n2 = scanner.nextInt();
scanner.nextLine();
        HashSet<String> set2 = new HashSet<>();
        for (int i = 0; i < n2; i++) {
set2.add(scanner.nextLine());
        }        set1.retainAll(set2);
for (String element : set1) {
            System.out.println(element);
        }
scanner.close();
    }
}
```

Test	Input	Expected	Got	
1	5 Football Hockey Cricket Volleyball Basketball 7 Golf Cricket Badminton Football Hockey Volleyball	Cricket Hockey Volleyball Football	Cricket Hockey Volleyball Football	

		Throwball			

Question **3**

Java HashMap Methods containsKey() Indicate if an entry with the specified key exists in the map containsValue() Indicate if an entry with the specified value exists in the map putIfAbsent() Write an entry into the map but only if an entry with the same key does not already exist remove() Remove an entry from the map replace() Write to an entry in the map only if it exists size() Return the number of entries in the map

Your task is to fill the incomplete code to get desired output

CODING

```
import java.util.HashMap; import
java.util.Map.Entry; import java.util.Set;
import java.util.Scanner; public class
Prog {     public static void main(String[]
args) {
      HashMap<String, Integer> map = new HashMap<String, Integer>();
String name;        int num;
      Scanner sc = new
Scanner(System.in);        int n =
sc.nextInt();        for (int i = 0; i < n; i++) {
name = sc.next();            num =
sc.nextInt();            map.put(name, num);
      }
      Set<Entry<String, Integer>> entrySet = map.entrySet();
for (Entry<String, Integer> entry : entrySet) {
          System.out.println(entry.getKey() + " : " + entry.getValue());
      }
      System.out.println("----------");
      HashMap<String, Integer> anotherMap = new HashMap<String,
Integer>();        anotherMap.put("SIX", 6);        anotherMap.put("SEVEN", 7);
```

```
        anotherMap.putAll(map);         entrySet =
anotherMap.entrySet();          for (Entry<String, Integer> entry :
entrySet) {
            System.out.println(entry.getKey() + " : " + entry.getValue());
        }
        map.putIfAbsent("FIVE", 5);         int value =
map.get("TWO");
        System.out.println(value);
        System.out.println(map.containsKey("ONE"));
        System.out.println(map.containsValue(3));
System.out.println(map.size());         sc.close();
    }
}
```

	Test	Input	Expected	Got	
	1	3 ONE 1 TWO 2 THREE 3	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	

Passed all tests!

LAB – 12

INTRODUCTION to I/O , I/O OPERATIONS , OBJECTS

Question **1**

You are provided with a string which has a sequence of 1's and 0's.

This sequence is the encoded version of a English word. You are supposed write a program to decode the provided string and find the original word.

Each alphabet is represented by a sequence of 0s.

This is as mentioned below:

Z : 0

Y : 00

X : 000

W : 0000

V : 00000

U : 000000 T : 0000000 and so on upto A having 26 0's

(00000000000000000000000000).

The sequence of 0's in the encoded form are separated by a single 1 which helps to distinguish between 2 letters.

For example:

Input	**Result**
010010001	ZYX
00001000000000000000001000000000001000000000100000000000001	WIPRO

CODING

```
import java.util.Scanner; public class

DecodeString {     public static void

main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String encoded = sc.nextLine();

        System.out.println( decode(encoded));

        sc.close();

    }

    public static String decode(String encoded) {

        String[] zeroGroups = encoded.split("1");

        StringBuilder decodedWord = new StringBuilder();
for (String group : zeroGroups) {
```

```
        if (group.length() > 0) {                char letter =
(char) ('Z' - (group.length() - 1));
decodedWord.append(letter);
        }
      }
      return decodedWord.toString();
   }
}
```

Input	Expected	Got	
010010001	ZYX	ZYX	
00001000000000000000001000000000001000000000100000000000001	WIPRO	WIPRO	

Passed all tests!

Question **2**

Write a function that takes an input String (sentence) and generates a new String (modified sentence) by reversing the words in the original String, maintaining the words position.

In addition, the function should be able to control the reversing of the case (upper or lowercase) based on a case_option parameter, as follows:

If case_option = 0, normal reversal of words i.e., if the original sentence is “Wipro TechNologies BangaLore”, the new reversed sentence should be “orpiW seigoloNhceT eroLagnaB”.

If case_option = 1, reversal of words with retaining position’s case i.e., if the original sentence is “Wipro TechNologies BangaLore”, the new reversed sentence should be “Orpiw SeigOlonhcet ErolaGnab”.

Note that positions 1, 7, 11, 20 and 25 in the original string are uppercase W, T, N, B and L.

Similarly, positions 1, 7, 11, 20 and 25 in the new string are uppercase O, S, O, E and G.

NOTE:

1. Only space character should be treated as the word separator i.e., “Hello World” should be treated as two separate words, “Hello” and “World”. However, “Hello,World”, “Hello;World”, “Hello-World” or “Hello/World” should be considered as a single word.

2. Non-alphabetic characters in the String should not be subjected to case changes. For example, if case option = 1 and the original sentence is “Wipro TechNologies, Bangalore” the new reversed sentence should be “Orpiw ,seiGolonhceT Erolagnab”. Note that comma has been treated as part of the word “Technologies,” and

when comma had to take the position of uppercase T it remained as a comma and uppercase T took the position of comma. However, the words “Wipro and Bangalore” have changed to “Orpiw” and “Erolagnab”.

3. Kindly ensure that no extra (additional) space characters are embedded within the resultant reversed String.

For example:

Input	**Result**
Wipro Technologies Bangalore 0	orpiW seigolonhceT erolagnaB
Wipro Technologies, Bangalore 0	orpiW ,seigolonhceT erolagnaB
Wipro Technologies Bangalore 1	Orpiw Seigolonhcet Erolagnab
Wipro Technologies, Bangalore 1	Orpiw ,seigolonhceT Erolagnab

CODING

```
import java.util.Scanner; public class
WordReversal {     public static void
main(String[] args) {
        Scanner sc = new
Scanner(System.in);         String sentence =
sc.nextLine();         int caseOption =
sc.nextInt();
        String result = reverseWords(sentence,
caseOption);         System.out.println(result);
sc.close();
    }
    public static String reverseWords(String sentence, int case_option) {
        String[] words = sentence.split(" ");
        StringBuilder modifiedSentence = new StringBuilder();
for (int i = 0; i < words.length; i++) {
            String word = words[i];
            StringBuilder reversedWord = new StringBuilder();
for (int j = word.length() - 1; j >= 0; j--) {
reversedWord.append(word.charAt(j));
            }
```

```
            if (case_option == 1) {                for (int j = 0; j <
word.length(); j++) {                    char originalChar =
word.charAt(j);                    char reversedChar =
reversedWord.charAt(j);


                if (Character.isUpperCase(originalChar)) {
reversedWord.setCharAt(j, Character.toUpperCase(reversedChar));
                } else if (Character.isLowerCase(originalChar)) {
reversedWord.setCharAt(j, Character.toLowerCase(reversedChar));
                }
              }
            }
            modifiedSentence.append(reversedWord);
if (i < words.length - 1) {
modifiedSentence.append(" ");
            }
        }
        return modifiedSentence.toString();
    }
}
```

Input	Expected	Got	
Wipro Technologies Bangalore 0	orpiW seigolonhceT erolagnaB	orpiW seigolonhceT erolagnaB	✓
Wipro Technologies, Bangalore 0	orpiW ,seigolonhceT erolagnaB	orpiW ,seigolonhceT erolagnaB	✓
Wipro Technologies Bangalore 1	Orpiw Seigolonhcet Erolagnab	Orpiw Seigolonhcet Erolagnab	✓
Wipro Technologies, Bangalore 1	Orpiw ,seigolonhceT Erolagnab	Orpiw ,seigolonhceT Erolagnab	✓

Passed all tests!

Question **3**

Given two char arrays input1[] and input2[] containing only lower case alphabets, extracts the alphabets which are present in both arrays (common alphabets).

Get the ASCII values of all the extracted alphabets.

Calculate sum of those ASCII values. Lets call it sum1 and calculate single digit sum of sum1, i.e., keep adding the digits of sum1 until you arrive at a single digit.

Return that single digit as output.

Note:

1. Array size ranges from 1 to 10.
2. All the array elements are lower case alphabets.
3. Atleast one common alphabet will be found in the arrays.

For example:

Input	Result
a b c b c	8

CODING

```
import java.util.Scanner; public class
CommonAlphabets {     public static
void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input1 = sc.nextLine();         String input2 =
sc.nextLine();         sc.close();         char[] array1 =
input1.replace(" ", "").toCharArray();         char[] array2
= input2.replace(" ", "").toCharArray();         int sum1 =
0;         for (char c1 : array1) {             for (char c2 :
array2) {                 if (c1 == c2) {                     sum1 +=
(int) c1;                     break;
                }
            }
        }
    int singleDigitSum = getSingleDigitSum(sum1);
    System.out.println(singleDigitSum);
  }
  private static int getSingleDigitSum(int number) {
while (number >= 10) {             int sum = 0;
while (number > 0) {                 sum += number % 10;
number /= 10;
            }
            number = sum;
        }
        return number;
    }
}
```

Input	Expected	Got	
a b c b c	8	8	

Passed all tests!

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

BLOOD BANK MANAGEMENT SYSTEM

CS23333 – Object Oriented Programming using Java Project Report

Submitted by

HARSHINI T - 2116231001059

SWETHA A - 2116231001226

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

NOVEMBER-2024

BONAFIDE CERTIFICATE

Certified that this project titled "BLOOD BANK MANAGEMENT SYSTEM" is the Bonafide work of "**HARSHINI T(231001059), SWETHA A (231001226)"**who carried out the project work under my supervision.

SIGNATURE :

Dr.P VALARMATHI
HEAD OFTHE DEPARTMENT
DEPARTMENT OF
INFORMATION TECHNOLOGY
RAJALAKSHMI ENGINEERING
COLLEGE THANDALAM
CHENNAI-602105

SIGNATURE:

This project is submitted for CS23333 – Object Oriented Programming using Java held on _____

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

Mr. Narayana K.E
Assistant Professor,
Information Technology
RAJALAKSHMI ENGINEERING COLLEGE
(Autonomous),
THANDALAM, CHENNAI-602105

1. Abstract

The Blood Bank Management System is a software application designed to streamline and automate the processes involved in blood donation, storage, and distribution. The primary aim of this system is to improve the efficiency and accuracy of managing blood inventories, ensuring that blood is available when needed, while also maintaining a database of donors, recipients, and blood types. This system is developed using Java, incorporating both graphical user interface (GUI) and back-end functionality. The system allows blood donors to register, update their details, and schedule donation appointments

2. Introduction

The development of the Blood Bank Management System aims to address the challenges faced by blood banks in managing large volumes of data, ensuring the safety and quality of blood, and improving accessibility. By leveraging Java's object-oriented principles, this project ensures that the system is modular, maintainable, and scalable, providing a robust solution for modern blood bank operations. Additionally, the system offers a user-friendly graphical interface that makes it easy for users to navigate and interact with the application.

3. Purpose

Automate blood donation and inventory management to reduce manual errors and improve efficiency.

Ensure accurate tracking of donor information, blood types, and stock levels.

Provide timely blood supply by monitoring and managing blood availability and distribution.

Engage donors by offering a platform for registration, history tracking, and event notifications.

Generate reports for inventory, donor activity, and blood distribution for better decision-making.

4. Scope of the project

- Allows donors to register their personal details and medical history.
- Provides an interface for updating donor records and viewing donation history.
- Alerts donors when they are eligible to donate again based on blood donation policies.

• Blood Inventory Management:

- Tracks blood donations received and updates inventory levels based on donations.

- Monitors expiration dates of blood units to ensure only safe and viable blood is distributed.
- Helps the system administrator track the stock of different blood types.

5. Software Requirement Specification

Introduction

This system provides an intuitive interface for both donors and administrators to enhance operational efficiency, accuracy, and transparency in blood bank operations. It helps blood banks efficiently track donor information, manage blood stocks, and ensure timely delivery to hospitals.

Product Scope

The Blood Bank Management System automates the processes of blood donation, inventory tracking, and distribution. It enables efficient management of donor information, blood stock, and donation schedules, while providing secure data handling. The system also includes reporting features for performance monitoring and decision-making. **References and Acknowledgement**

Online course:

1.Oracle, Java Documentation. Available at: https://docs.oracle.com/javase/8/docs/

2.W3Schools, Java Tutorials. Available at: https://www.w3schools.com/java/

Books:

- Sommerville, I. (2011). *Software Engineering* (9th ed.). Boston: Addison-Wesley.
- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill.

1.Acknowledgements:

- We would like to express our sincere gratitude to our project supervisor for their guidance and support throughout the development of this project.
- Special thanks to the developers, designers, and contributors whose work and research were foundational in shaping the system's architecture and functionality.

- We also acknowledge the importance of online resources, textbooks, and Java documentation for providing the essential knowledge and tools required for successful project implementation

Overall Description

The **Blood Bank Management System** is a comprehensive software solution designed to automate and manage the various functions of a blood bank, from donor registration and inventory management to blood distribution. Built using Java, the system aims to streamline the management of blood donations, track blood stock levels, and ensure efficient distribution to hospitals in need. The use of Java ensures that the system is scalable, secure, and can be easily maintained. The system aims to optimize the management of blood donations, inventory, and distribution, contributing to more effective healthcare services and ensuring a reliable blood supply for those in need.

Product Perspective

The system integrates with a relational database (such as MySQL, SQLite, or similar) to store critical data such as donor details, blood types, donation history, and blood inventory. The database provides the foundation for data management, ensuring secure, structured storage and retrieval of information. The Blood Bank Management System is designed to function as an independent application on a local network or a single machine. It can be used by blood banks of any size, from small clinics to larger regional facilities.

Product Functionality:

Donor Registration and Profile Management:

Donors can update their profile information as needed.

Eligibility Check:

- Automatically checks donor eligibility based on donation history (e.g., the time interval between donations) and notifies the donor when they are eligible to donate again.

Expiration Monitoring:

- The system monitors blood expiration dates and alerts administrators when blood units are nearing expiration, ensuring that expired blood is discarded.

Blood Request Handling:

- Hospitals or healthcare facilities can submit blood requests based on urgent needs or scheduled surgeries.

Inventory Reports:

- Generate detailed reports on current blood inventory levels, including the amount of each blood type available and near expiration.

Admin Role:

Full access to all system features, including managing donors, blood inventory, donation scheduling, and generating reports.

- **Donor Role:**

Donors can access their profile, donation history, and schedule appointments, but have no access to system administration functions.

- **Staff Role:**

Staff members can manage blood inventory, process blood requests, and help with scheduling donation camps but have limited access to sensitive data.

User and Characteristics:

Qualification: Users should be comfortable reading and understanding English, as the system interface and documentation will primarily be in English.

Experience: Familiarity with medical or healthcare processes, such as blood donation or inventory management, is advantageous.

Experience in interacting with computer systems and online platforms is beneficial but not mandatory for basic users.

Operating Environment

Hardware Requirements

Processor: Intel i3 or higher (or equivalent AMD processor).

Operating System: Windows 8, 10, or 11.

Processor Speed: Minimum 2.0 GHz.

RAM: Minimum 4GB.

Hard Disk: Minimum 500GB of storage available for system files and data.

Software Requirements

Database: MySQL (used for data storage and management).

Frontend Technology: Java Swing or JavaFX (for the user interface).

Backend Technology: Java (with JDBC for database connectivity).

Web Server (optional): Apache Tomcat (for running the JSP interface, if required).

Constraints

- The system is designed to handle moderate volumes of data and users, suitable fortypical blood bank operations. It will effectively manage donor records, blood stock levels, and blood distribution without performance issues under normal usage conditions.

User Interface

The **Blood Bank Management System** offers an intuitive, menu-driven interface with the following features:

- **Register:**
  - Allows new users (donors) to register by entering their personal information, blood type, and contact details.
- **Login:**
  - Existing users (donors, administrators) can log in securely with their credentials to access their respective accounts and perform actions.
- **Donor Dashboard:**
  - Donors can view their donation history, schedule appointments, and see eligibility for future donations.
- **Blood Inventory Management:**
  - Administrators can access the inventory page to track the available blood types, quantities, and expiration dates.
- **Order/Request Blood:**
  - Hospitals or healthcare facilities can view available blood types and place requests for blood based on their needs.
- **Reports and Analytics:**
  - Admins can generate reports on blood donations, inventory levels, and donor statistics for better decision-making.

Hardware Interface:

- **Screen Resolution:**
  - The system is optimized to function properly with a minimum screen resolution of 640 x 480, though higher resolutions are recommended for better display.
- **Operating System Compatibility:**
  - Compatible with any version of Windows 8, 10, or 11 for smooth system operation and user experience**.**

Software Interface:

- **Operating System:** ○ MS-Windows (8, 10, 11) is required to run the Blood Bank Management System.
- **Frontend Technology:**
    - JSP (Java Server Pages) is used for designing the user interface, allowing for a dynamic, responsive web interface**.**
- **Backend Technology:**
    - Java is used for the backend processing and business logic of the system, ensuring robust and scalable operations**.**
- **Database:**
    - MySQL is used for data storage, managing donor records, blood inventory, and transaction logs securely and efficiently

---

Functional Requirements:

1. User Registration and Authentication:

- **User Registration:**
    - Allow users (donors, administrators, and hospital staff) to register with their details such as name, email, phone number, blood type, address, and medical history for donors.
- **Authentication:**
    - Provide secure login/logout functionality with encrypted passwords, ensuring that only authorized users can access the system.

2. Donor Management:

- **Donor Profile Management:**
    - Allow donors to update their profile information (contact details, medical history, eligibility status for donation).
- **Donation History:**
    - Track the donation history for each donor, including blood type, donation dates, and eligibility for future donations.

3. Blood Inventory Management:

□ Inventory Management:

- Track and display blood types, quantities, and expiration dates of donated blood in real-time.

4. Blood Request and Distribution:

- **Blood Request:**
  - Allow hospitals and healthcare facilities to request blood by specifying required blood type and quantity.
- **Blood Distribution:**
  - Admins can approve and track blood distribution to healthcare facilities and hospitals based on available inventory.

5. Reports and Analytics:

- **Generate Reports:**
  - Admins should be able to generate real-time reports, such as donor participation, blood inventory levels, donation trends, and blood usage.
- **Analytics:**
  - Analyze donation trends, blood stock usage, and demand from hospitals to optimize donation drives and inventory management.

6. Admin Functions:

- **Manage Donors:**
  - Admins can add, update, or remove donor records, ensuring that donor data is upto-date and accurate.

Non-functional Requirements:

1. Performance:

- The system should load quickly (within 2-3 seconds) and perform efficiently even when handling a moderate volume of blood donations, donor registrations, and hospital requests.

2. Scalability:

- The system should be able to handle increasing numbers of users, blood donations, and requests from healthcare facilities without performance degradation.

2.SYSTEM FLOW DIAGRAM

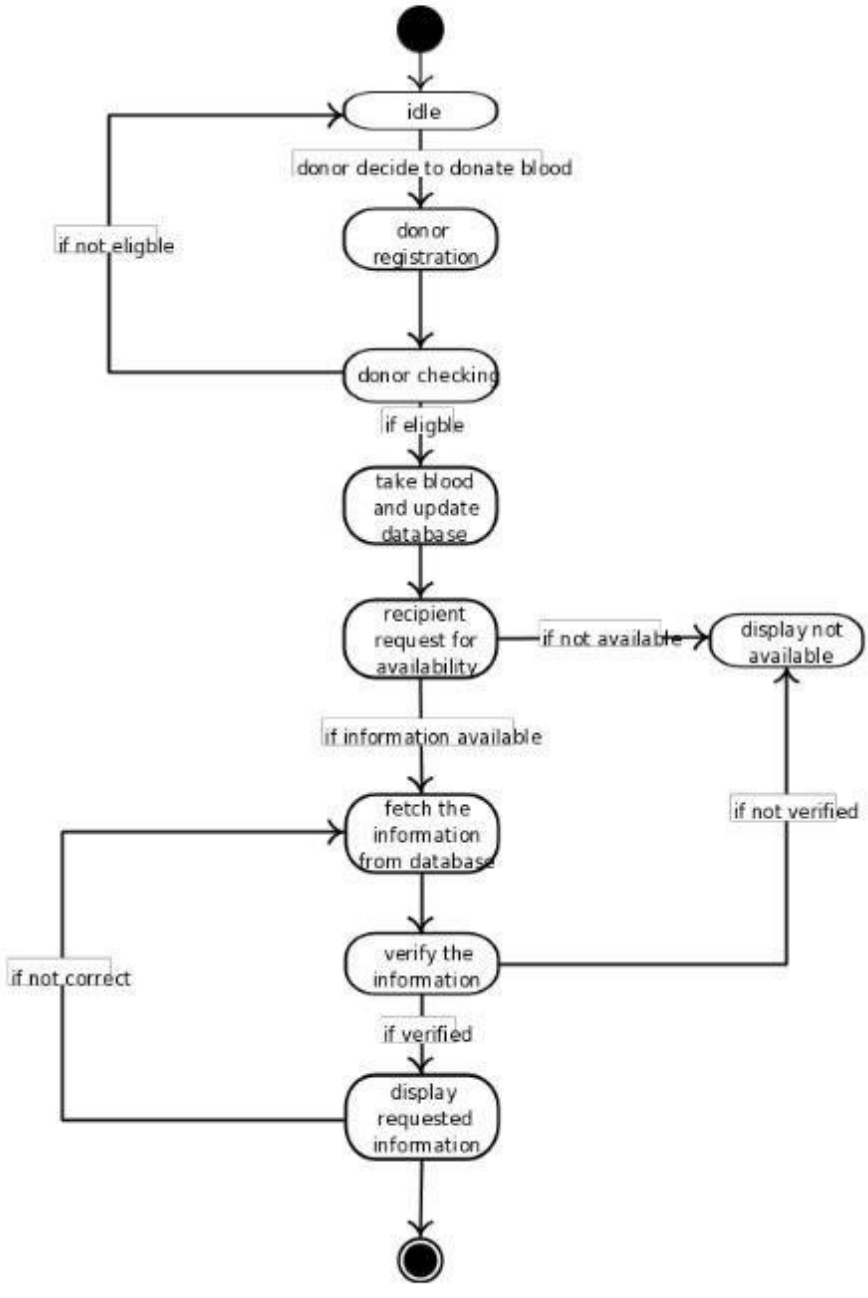


Figure 2.1 Use Case Diagrams

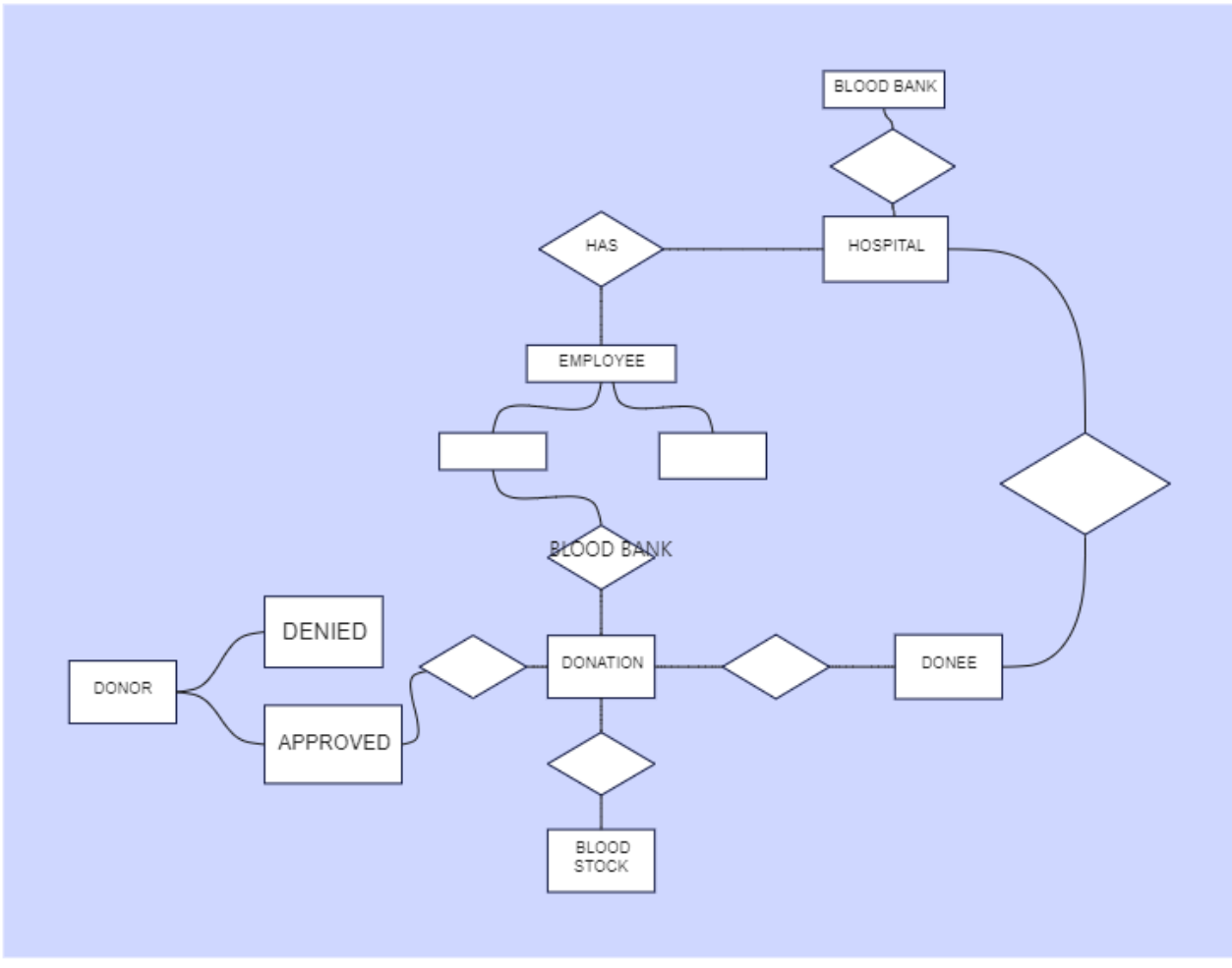


Figure 2.2 Entity Relationship Diagram

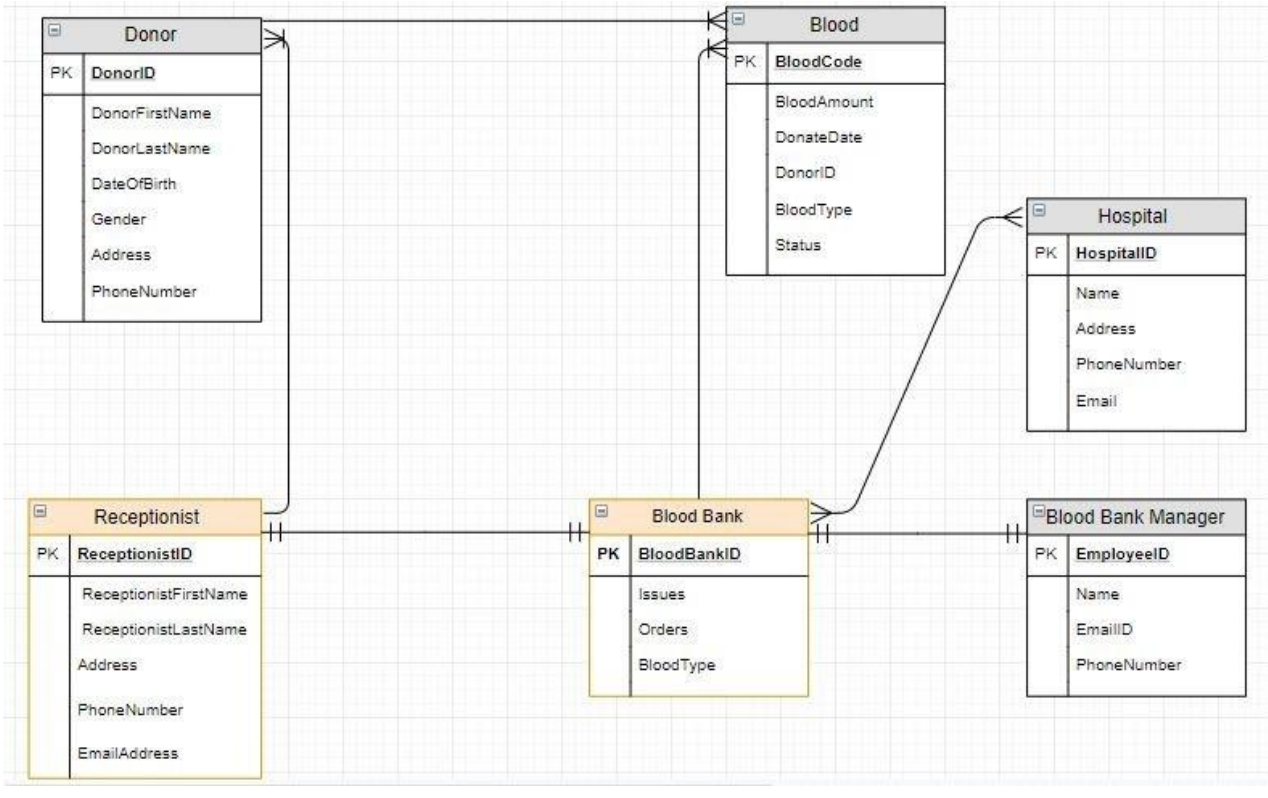


Figure 2.3 Data-flow diagram

3.Module Description for Blood Bank Management System:

User Management: Handles user registration, login, profile management, and authentication for donors, admins, and hospital staff

1. Blood Inventory Management: Manages tracking, updating, and monitoring of blood stocks, including blood type, quantity, and expiration dates.

2. Donation Management: Oversees the scheduling of blood donations, donor eligibility checks, and logging of donation history.

3. Blood Request & Distribution Management: Allows hospitals to request blood, and admins to validate, approve, and distribute blood.

4. Reporting and Analytics: Generates reports and provides analytics on blood donations, inventory levels, and hospital blood requests.

5. Admin Dashboard: Provides admins with tools to manage users, track inventory, process blood requests, and generate reports.

3.1 TOOLS /PLATFORM

Programming Language: Java for backend development.

IDE: Eclipse, IntelliJ IDEA, or NetBeans for Java development.

Database Management: MySQL for relational database management.

Frontend Technology: JSP for dynamic web page creation.

Web Server: Apache Tomcat for serving Java-based web applications.

Frameworks: Spring Framework for backend development, Hibernate ORM for database object-relational mapping.

Version Control: Git for version control, GitHub/GitLab for remote code repositories and collaboration.

Project Management: JIRA, Trello, or Asana for task management and project tracking.

Testing Tools: JUnit for unit testing, Selenium for automated functional testing.

Deployment Platforms: AWS, Heroku, or Digital Ocean for cloud hosting and deployment.

Security Tools: SSL/TLS for secure data transmission, OWASP ZAP for vulnerability scanning.

Documentation Tools: Swagger or Postman for API documentation and testing, Confluence for team documentation.

User Interface Design Tools: Figma, Adobe XD, or Sketch for designing wireframes and UI mock ups.

Communication Tools: Slack for team collaboration and communication.
Continuous Integration/Continuous Deployment (CI/CD): Jenkins or Circle CI for automating testing, integration, and deployment processes.

Containerization: Docker for containerizing the application for easier deployment and scalability.

Monitoring Tools: Prometheus or New Relic for monitoring the application's performance and health in production.

Backup and Recovery: AWS RDS Automated Backups, or MySQL Workbench for database backup and restoration.

Search Functionality: Apache or Elasticsearch for integrating efficient search functionalities into the system.

Analytics Tools: Google Analytics for tracking user activity and behavior on the platform.

3.2 IMPLEMENTATION

Implementation for Blood Bank Management System

The implementation of the Blood Bank Management System involves building a complete system that allows users (donors, hospitals, and admins) to manage blood donations, requests, inventory, and more. Below is a structured approach for implementing the system:

3.3 Project Setup & Environment Configuration

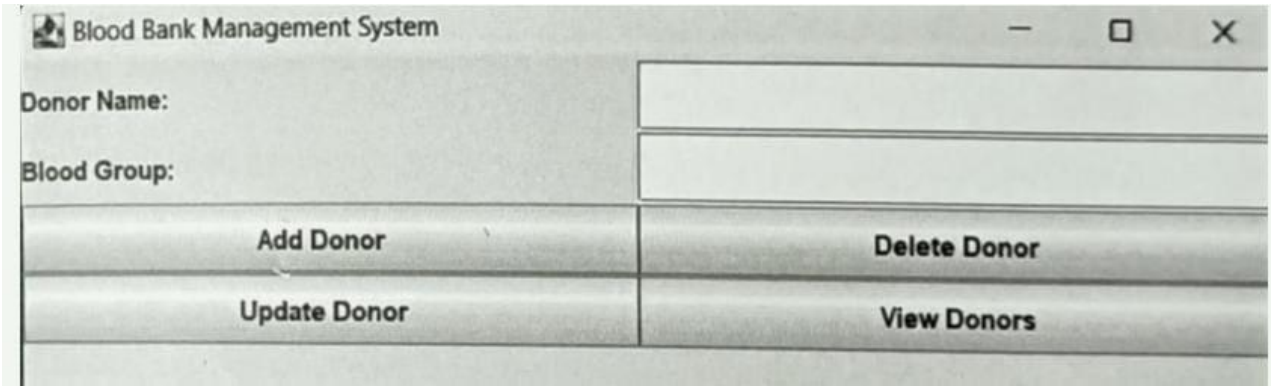
Tools Used:

- Java: Backend development using Java for business logic and APIs.
- MySQL: Setting up the relational database to manage donor details, inventory, blood requests, and admin data.
- JSP & Servlets: For building the frontend and backend web pages.
- Apache Tomcat: For hosting and running the web application. **Backend Development (Java & Spring Framework)**

Steps:
- **Set up the Spring Framework** to handle backend logic and business processes.
- Develop **Java classes** to implement core functionalities, including:

User Authentication & Authorization: Implement login, logout, and session management using Spring Security

4. INPUT IMAGES:



4.1 Design

4.2 DATA DESIGN:

1)Donor Table:

It stores the details of blood donors, including personal information and their donation history.

- **Table Name:** donors
- **Columns:**

o donor_id (Primary Key): A unique identifier for each donor. o

name: The full name of the donor.

- email: The email address of the donor. ○ phone: Contact number of the donor. ○ blood_type: The blood type of the donor (e.g., A+, O-, etc.). ○ last_donation_date: The date of the most recent blood donation.
- eligible_for_donation: A flag (Boolean) indicating if the donor is eligible to donate blood again.

---

2)Blood Inventory Table

The **Blood Inventory Table** holds data on the blood available at the blood bank, including quantities, types, and expiration dates.

Table Name:

Blood_inventory:

- **Columns:**

○ inventory_id (Primary Key): A unique identifier for each blood stock record. ○ blood_type: The type of blood (A+, B-, O+, etc.). ○ quantity: The amount of blood (usually in liters or pints). ○ donation_date: The date when the blood was donated. ○ expiry_date: The date when the blood expires. ○ status: A status indicating whether the blood is available, expired, or used.

---

3) Blood Request Table

The **Blood Request Table** manages requests made by hospitals or healthcare providers for specific blood types and quantities. **Table Name:** blood_requests

- **Columns:**

○ request_id (Primary Key): A unique identifier for each request. ○ hospital_name: The name of the hospital or healthcare facility requesting blood. ○

blood_type: The type of blood requested (e.g., A+, O-, etc.). ○ quantity_requested: The amount of blood requested. ○ request_date: The date on which the request was made. ○ status: The status of the request (e.g., Pending, Fulfilled, Cancelled).

---

4) Admin Table

The **Admin Table** stores information related to the system administrators who manage the blood bank operations. **Table Name:** admins

- **Columns:**

○ admin_id (Primary Key): A unique identifier for each admin. ○ name: The name of the admin. ○ email: The admin's email address. ○ password: The encrypted password for secure login.

○ role: The role of the admin (e.g., Super Admin, Inventory Manager, Request Manager).

4.3 CODING:

CONNECTION PROVIDER .JAVA

```
package Project; import java.sql.*; public
class ConnectionProvider
{
    public static Connection getCon(){
            try {
            Class.forName("con.mysql.jdbc.Driver");
            Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms","root","de vi");
    return con;
            }


            catch(Exception e) {                    return null;
            }

    }
```

```
}
```

BLOOD GROUP SEARCH.JAVA

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

public class BloodGroupSearch extends JFrame {     private JTextField bloodGroupField;
private JTextArea resultArea;

   public BloodGroupSearch() {         setTitle("Search Blood Group");         setSize(679, 506);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel bloodGroupLabel = new JLabel("Enter Blood Group:");
bloodGroupLabel.setFont(new Font("Tahoma", Font.BOLD, 15));
bloodGroupLabel.setBounds(34, 135, 174, 33);         bloodGroupField = new JTextField();
bloodGroupField.setBounds(260, 122, 337, 59);

        JButton searchButton = new JButton("Search");         searchButton.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\search1.png"))
;
        searchButton.setFont(new Font("Tahoma", Font.BOLD, 14));
searchButton.setBounds(60, 227, 148, 44);

        resultArea = new JTextArea();         resultArea.setBounds(34, 305, 574, 129);
resultArea.setEditable(false);

        getContentPane().setLayout(null);         getContentPane().add(bloodGroupLabel);
getContentPane().add(bloodGroupField);         getContentPane().add(searchButton);
getContentPane().add(resultArea);

        JSeparator separator = new JSeparator();         separator.setBounds(0, 102, 691, 10);
getContentPane().add(separator);

        JLabel lblNewLabel = new JLabel("Search Donor Details");
lblNewLabel.setFont(new Font("Algerian", Font.BOLD, 27));
lblNewLabel.setBounds(158, 41, 337, 33);         getContentPane().add(lblNewLabel);
JButton btnClose = new JButton("Close");         btnClose.addActionListener(new
ActionListener() {          public void actionPerformed(ActionEvent e)
{           setVisible(false);
                }
        });
        btnClose.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));
btnClose.setFont(new Font("Tahoma", Font.BOLD, 14));         btnClose.setBounds(306, 227,
148, 44);         getContentPane().add(btnClose);

        JLabel lblNewLabel_1 = new JLabel("New label");
lblNewLabel_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background
image.png"));
```

```
        lblNewLabel_1.setBounds(0, 10, 667, 461);         getContentPane().add(lblNewLabel_1);
searchButton.addActionListener(new  ActionListener()  {                          public  void
actionPerformed(ActionEvent e) {                searchBloodGroup();
            }
        });

        setVisible(true);
    }

    private void searchBloodGroup() {
        String bloodGroup = bloodGroupField.getText();

        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms", "root",
"devi");

            String sql = "SELECT * FROM Donors WHERE blood_group = ?";             stmt =
conn.prepareStatement(sql);             stmt.setString(1, bloodGroup);

            rs = stmt.executeQuery();

            StringBuilder result = new StringBuilder();

            while (rs.next()) {
                int donorId = rs.getInt("donor_id");
                String donorName = rs.getString("donor_name");
                String donorEmail = rs.getString("donor_email");
                String donorBloodGroup = rs.getString("blood_group");

                result.append("Donor ID: ").append(donorId)                    .append(", Name:
").append(donorName)
                    .append(", Email: ").append(donorEmail)
                    .append(", Blood Group: ").append(donorBloodGroup)
.append("\n");
            }

            if (result.length() == 0) {
                resultArea.setText("No donors found for blood group: " + bloodGroup);
            } else {
                resultArea.setText(result.toString());
            }

        } catch (SQLException ex) {
ex.printStackTrace();
            resultArea.setText("Error: Unable to retrieve donors.");
        } finally {            try {                if (rs != null) {                    rs.close();
                }
```

```
            if (stmt != null) {                 stmt.close();
            }
            if (conn != null) {                 conn.close();
}
        } catch (SQLException ex) {             ex.printStackTrace();
        }
    }
  }

  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new BloodGroupSearch());
  }
}
```

DONOR DELETION FRAME.JAVA

```
import javax.swing.; import java.awt.event.; import java.sql.*; import java.awt.BorderLayout;
import java.awt.Font;

public class DonorDeletionFrame extends JFrame {    private JTextField donorIDField;
private JButton deleteButton;    private Connection connection;    private PreparedStatement
preparedStatement;    private JLabel lblNewLabel;    private JButton btnClose;    private
JLabel lblNewLabel_1;

  public DonorDeletionFrame() {        setTitle("Delete Donor");        setSize(708, 505);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
getContentPane().setLayout(null);              donorIDField = new JTextField(10);
donorIDField.setBounds(247, 199, 207, 37);              getContentPane().add(donorIDField);
deleteButton = new JButton("Delete");              deleteButton.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\delete donor.png"));
deleteButton.setFont(new Font("Tahoma", Font.BOLD, 16));
deleteButton.setBounds(98, 333, 172, 66);              getContentPane().add(deleteButton);

                JLabel lblDonorId = new JLabel("Donor ID:");
lblDonorId.setFont(new Font("Tahoma", Font.BOLD, 15));
lblDonorId.setBounds(98, 193, 110, 45);                getContentPane().add(lblDonorId);

                lblNewLabel = new JLabel("Delete Donor Details");
lblNewLabel.setFont(new Font("Algerian", Font.BOLD, 30));
lblNewLabel.setBounds(151, 52, 359, 37);0
getContentPane().add(lblNewLabel);

                JSeparator separator = new JSeparator();
separator.setBounds(10, 121, 674, 44);                getContentPane().add(separator);
btnClose = new JButton("Close");
                btnClose.addActionListener(new ActionListener() {                    public
void actionPerformed(ActionEvent e) {                    setVisible(false);
                    }
                });
                btnClose.setIcon(new
```

```
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));
btnClose.setFont(new Font("Tahoma", Font.BOLD, 16));
btnClose.setBounds(356, 333, 172, 66);                    getContentPane().add(btnClose);

                lblNewLabel_1 = new JLabel("New label");
                lblNewLabel_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background
image.png"));
                lblNewLabel_1.setBounds(10, 10, 674, 448);
getContentPane().add(lblNewLabel_1);                    deleteButton.addActionListener(new
ActionListener()
{                       public void actionPerformed(ActionEvent e)
{                          deleteDonor();
                      }
                });
      setLocationRelativeTo(null);

      // Connect to the database        try {
         connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms", "root", "devi");
} catch (SQLException e) {           e.printStackTrace();
      }
   }

   private void deleteDonor() {
      try {
         int donorID = Integer.parseInt(donorIDField.getText());
         String deleteQuery = "DELETE FROM Donors WHERE donor_id = ?";

         preparedStatement = connection.prepareStatement(deleteQuery);
preparedStatement.setInt(1, donorID);

         int rowsAffected = preparedStatement.executeUpdate();           if (rowsAffected > 0) {
JOptionPane.showMessageDialog(this, "Donor deleted successfully.");
         } else {
            JOptionPane.showMessageDialog(this, "No donor found with that ID.");
         }
      } catch (SQLException | NumberFormatException e) {
         JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
      }
   }
   public static void main(String[] args) {
      // Ensure the database driver is loaded (e.g., for MySQL)        try {
         Class.forName("com.mysql.cj.jdbc.Driver");        } catch (ClassNotFoundException e)
{
         e.printStackTrace();           return;
      }

      SwingUtilities.invokeLater(() -> {
```

```
            DonorDeletionFrame frame = new DonorDeletionFrame();
frame.setVisible(true);
        });
    }
}
```

ADD NEW DONOR.JAVA

```
import javax.swing.; import java.awt.; import java.awt.event.; import java.sql.; import
Project.ConnectionProvider; public class addNewDonar extends JFrame {
    /**
     *         */
     private static final long serialVersionUID = 1L;         private JTextField nameField,
emailField, bloodGroupField;     private JButton submitButton;     private JButton btnClose;
private JLabel lblNewLabel;     private JSeparator separator;     private JSeparator separator_1;
private JLabel lblNewLabel_1;

    public addNewDonar() {         setTitle("Donor Information");         setSize(708, 559);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel nameLabel = new JLabel("Name: ");
nameLabel.setFont(new Font("Tahoma", Font.BOLD, 16));
nameLabel.setBounds(18, 114, 171, 67);         nameField = new JTextField();
nameField.setFont(new Font("Tahoma", Font.PLAIN, 16));
nameField.setBounds(320, 118, 364, 59);         JLabel emailLabel = new JLabel("Email: ");
emailLabel.setFont(new Font("Tahoma", Font.BOLD, 16));         emailLabel.setBounds(18,
210, 228, 74);         emailField = new JTextField();         emailField.setFont(new
Font("Tahoma", Font.PLAIN, 16));         emailField.setBounds(320, 218, 364, 59);
        JLabel bloodGroupLabel = new JLabel("Blood Group: ");
bloodGroupLabel.setFont(new Font("Tahoma", Font.BOLD, 16));
bloodGroupLabel.setBounds(18, 309, 228, 74);         bloodGroupField = new JTextField();
bloodGroupField.setFont(new Font("Tahoma", Font.PLAIN, 16));
bloodGroupField.setBounds(320, 317, 364, 59);         submitButton = new JButton("Submit");
submitButton.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\save.png"));
submitButton.setFont(new Font("Tahoma", Font.BOLD, 16));
submitButton.setBounds(40, 415, 196, 74);         getContentPane().setLayout(null);

        getContentPane().add(nameLabel);         getContentPane().add(nameField);
getContentPane().add(emailLabel);         getContentPane().add(emailField);
getContentPane().add(bloodGroupLabel);         getContentPane().add(bloodGroupField);
getContentPane().add(submitButton);

        btnClose = new JButton("Close");
        btnClose.addActionListener(new ActionListener() {         public void
actionPerformed(ActionEvent e) {         setVisible(false);
            }
        });
        btnClose.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));
btnClose.setFont(new Font("Tahoma", Font.BOLD, 16));         btnClose.setBounds(352,
415, 196, 74);         getContentPane().add(btnClose);         separator = new JSeparator();
separator.setBounds(0, 103, 702, 38);         getContentPane().add(separator);         separator_1
```

```
= new JSeparator();        separator_1.setBounds(-113, 393, 702, 38);
getContentPane().add(separator_1);        lblNewLabel_1 = new JLabel("ADD NEW
DONAR");        lblNewLabel_1.setFont(new Font("Algerian", Font.BOLD, 26));
lblNewLabel_1.setBounds(172, 41, 330, 38);        getContentPane().add(lblNewLabel_1);


        lblNewLabel = new JLabel("New label");
        lblNewLabel.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background
image.png"));
        lblNewLabel.setBounds(0, 10, 694, 512);        getContentPane().add(lblNewLabel);




        submitButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {                saveDonorInformation();
            }
        });

        setVisible(true);
    }

    private void saveDonorInformation() {        String name = nameField.getText();
        String email = emailField.getText();
        String bloodGroup = bloodGroupField.getText()
        Connection conn = null;
        PreparedStatement stmt = null;        try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms", "root",
"devi");
            String sql = "INSERT INTO Donors (donor_name, donor_email, blood_group)
VALUES (?, ?, ?)";            stmt = conn.prepareStatement(sql);            stmt.setString(1,
name);            stmt.setString(2, email);            stmt.setString(3, bloodGroup);
stmt.executeUpdate();
JOptionPane.showMessageDialog(this, "Donor information saved successfully.");
        } catch (SQLException ex) {            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error: Unable to save donor information.");
} finally {            try {
                if (stmt != null) {                stmt.close();
                }
                if (conn != null) {                conn.close();
}
            } catch (SQLException ex) {                ex.printStackTrace();
            }
        }
    }
 public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new addNewDonar());
    }
}
```

ALL DONOR.JAVA

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

import java.awt.; import java.awt.event.; import java.sql.*;


public class alldonar extends JFrame {     private JTable donorTable;     private
DefaultTableModel tableModel;

   public alldonar() {         setTitle("Donor Details");         setSize(705, 518);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

       tableModel = new DefaultTableModel();         donorTable = new JTable(tableModel);

       JScrollPane scrollPane = new JScrollPane(donorTable);         scrollPane.setBounds(10,
71, 673, 326);

       JButton printButton = new JButton("Print");         printButton.setBounds(131, 417, 112,
42);
       printButton.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\print.png"));
printButton.setFont(new Font("Tahoma", Font.BOLD, 14));
getContentPane().setLayout(null);         getContentPane().add(scrollPane);
getContentPane().add(printButton);

       JLabel lblNewLabel = new JLabel("All Donor Details");
lblNewLabel.setBounds(200, 23, 344, 42);
       lblNewLabel.setFont(new Font("Algerian", Font.BOLD, 26));
getContentPane().add(lblNewLabel);

       JButton printButton_2 = new JButton("Close");
printButton_2.addActionListener(new ActionListener() {         public void
actionPerformed(ActionEvent e) {           setVisible(false);
           }
       });
       printButton_2.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit application.png"));
printButton_2.setFont(new Font("Tahoma", Font.BOLD, 14));
printButton_2.setBounds(368, 417, 112, 42);         getContentPane().add(printButton_2);
JLabel lblNewLabel_1 = new JLabel("New label");         lblNewLabel_1.setBounds(0, -
29, 776, 531);
       lblNewLabel_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all page background
image.png"));
       getContentPane().add(lblNewLabel_1);

       printButton.addActionListener(new ActionListener() {             public void
actionPerformed(ActionEvent e) {                 try {
donorTable.print(); // Prints the table content                 } catch
```

```
(java.awt.print.PrinterException ex) {                    ex.printStackTrace();
                }
            }
        });

        populateDonorsTable();

        setVisible(true);
    }

    private void populateDonorsTable() {
        Connection conn = null;         Statement stmt = null;         try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms",
"root", "devi");
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT donor_id, donor_name, donor_email,
blood_group FROM Donors");

            ResultSetMetaData metaData = rs.getMetaData();             int columnCount =
metaData.getColumnCount();

            for (int i = 1; i <= columnCount; i++) {
                tableModel.addColumn(metaData.getColumnName(i));
            }

            while (rs.next()) {
                Object[] row = new Object[columnCount];                 for (int i = 1; i <=
columnCount; i++) {                    row[i - 1] = rs.getObject(i);
                }
                tableModel.addRow(row);
            }
        } catch (SQLException ex) {             ex.printStackTrace();
        } finally {             try {                 if (stmt != null) {                     stmt.close();
}                 if (conn != null) {                     conn.close();                 }
            } catch (SQLException ex) {                 ex.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new alldonar());
    }
}
```

HOME.JAVA

```
import java.awt.EventQueue;

import javax.swing.JFrame; import javax.swing.JPanel; import
javax.swing.border.EmptyBorder; import javax.swing.JMenuBar; import javax.swing.JMenu;
import javax.swing.JMenuItem; import javax.swing.JOptionPane; import
javax.swing.JCheckBoxMenuItem; import javax.swing.ImageIcon; import
```

```
java.awt.event.ItemListener; import java.awt.event.ItemEvent; import java.awt.Font; import
javax.swing.JLabel; import java.awt.event.ActionListener; import
java.awt.event.ActionEvent; import java.awt.Color; public class home extends JFrame {
 private static final long serialVersionUID = 1L;  private JPanel contentPane; /**
 * Launch the application.  */
    public static void main(String[] args) {
 EventQueue.invokeLater(new Runnable() {    public void run() {
                        try {
                                home frame = new home();
   frame.setVisible(true);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        });
   }

   /**
    * Create the frame.
    */       public
home() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);                setBounds(100,
100, 1366, 853);            contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JMenuBar menuBar = new JMenuBar();                menuBar.setBounds(0, 0, 1555,
71);                contentPane.add(menuBar);

        JMenu mnNewMenu = new JMenu("Donor");
        mnNewMenu.setFont(new Font("SansSerif", Font.BOLD,
16));
        mnNewMenu.addItemListener(new ItemListener() {                        public
void itemStateChanged(ItemEvent e) {
                }
        });
        mnNewMenu.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Donor.png"));
   menuBar.add(mnNewMenu);

        JMenuItem mntmNewMenuItem_1 = new
JMenuItem("Add New");
        mntmNewMenuItem_1.addActionListener(new
ActionListener() {          public void
   actionPerformed(ActionEvent e) {
                new addNewDonar().setVisible(true);
                }
        });
```

```
            mntmNewMenuItem_1.setFont(new Font("SansSerif", Font.PLAIN,
14));
            mntmNewMenuItem_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Add new.png"));
            mnNewMenu.add(mntmNewMenuItem_1);

            JMenuItem mntmNewMenuItem = new
JMenuItem("Update");

ActionListener() {          mntmNewMenuItem.addActionListener(new
                    public void actionPerformed(ActionEvent e) {
                            new updateDonar().setVisible(true);
                    }
            });

Font.PLAIN, 14));          mntmNewMenuItem.setFont(new Font("SansSerif",
            mntmNewMenuItem.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Update details.png"));
            mnNewMenu.add(mntmNewMenuItem);

            JMenu mnNewMenu_1 = new JMenu("Search Blood
Donar");
            mnNewMenu_1.setFont(new Font("SansSerif", Font.BOLD,
16));
            mnNewMenu_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\search user.png"));
            menuBar.add(mnNewMenu_1);

            JMenuItem mntmNewMenuItem_4 = new
JMenuItem("Blood Group");

ActionListener() {          mntmNewMenuItem_4.addActionListener(new
                    public void actionPerformed(ActionEvent e) {
                            new BloodGroupSearch().setVisible(true);
                    }
            });

Font.PLAIN, 14));          mntmNewMenuItem_4.setFont(new Font("SansSerif",
   mntmNewMenuItem_4.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Blood group.png"));
            mnNewMenu_1.add(mntmNewMenuItem_4);

            JMenu mnNewMenu_2 = new JMenu("Details"); mnNewMenu_2.setFont(new
Font("SansSerif", Font.BOLD,
16));
            mnNewMenu_2.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\stock.png"));
            menuBar.add(mnNewMenu_2);

            JMenuItem mntmNewMenuItem_2_1 = new
```

```
JMenuItem("All Donar Details");
		mntmNewMenuItem_2_1.addActionListener(new
ActionListener() {

			public void actionPerformed(ActionEvent e) {				new
    alldonar().setVisible(true);
			}
		});
		mntmNewMenuItem_2_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Details.png"));

Font.PLAIN, 14));		mntmNewMenuItem_2_1.setFont(new Font("SansSerif",
		mnNewMenu_2.add(mntmNewMenuItem_2_1);

		JMenu mnNewMenu_3 = new JMenu("Delete Donar");

Font.BOLD, 16));		mnNewMenu_3.setFont(new Font("SansSerif",
mnNewMenu_3.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\delete donor.png"));
		menuBar.add(mnNewMenu_3);

		JMenuItem mntmNewMenuItem_8 = new
JMenuItem("Delete Donar");

ActionListener() {		mntmNewMenuItem_8.addActionListener(new
		public void actionPerformed(ActionEvent e) {
		    new DonorDeletionFrame().setVisible(true);
			}

		});

Font.PLAIN, 14));		mntmNewMenuItem_8.setFont(new Font("SansSerif",
   mntmNewMenuItem_8.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\delete.png"));
		mnNewMenu_3.add(mntmNewMenuItem_8);

		JMenu mnNewMenu_4 = new JMenu("Exit");
		mnNewMenu_4.setFont(new Font("SansSerif",
Font.BOLD, 16));
		mnNewMenu_4.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\exit.png"));
menuBar.add(mnNewMenu_4);

		JMenuItem mntmNewMenuItem_9 = new
JMenuItem("Logout");
		mntmNewMenuItem_9.addActionListener(new
ActionListener() {
			public void actionPerformed(ActionEvent e) {
				int
a=JOptionPane.showConfirmDialog(null,"Do you really want to
```

```
logout","Select",JOptionPane.YES_NO_OPTION);
                                if(a==0)
                                {
                                        setVisible(false);
                                        new login().setVisible(true);
                                }
                        }
                });
                mntmNewMenuItem_9.setFont(new Font("SansSerif", Font.PLAIN,
        14));
                mntmNewMenuItem_9.setIcon(new
        ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Logout.png"));
                mnNewMenu_4.add(mntmNewMenuItem_9);

                JMenuItem mntmNewMenuItem_10 = new
        JMenuItem("Exit Application");
        mntmNewMenuItem_10.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent e) {
                                int
        a=JOptionPane.showConfirmDialog(null,"Do you really want to Close
the Application","Select",JOptionPane.YES_NO_OPTION); if(a==0)
                                                        System.exit(0);
                                        }
                                });
                                        btnNewButton_1.setFont(new Font("Serif", Font.BOLD,
26));
                btnNewButton_1.setBounds(920, 492, 168, 62);
contentPane.add(btnNewButton_1);
  JLabel lblNewLabel_1_2 = new JLabel("\"Give the gift of life");
                                        lblNewLabel_1_2.setForeground(new Color(255, 255,
255));
                                        lblNewLabel_1_2.setFont(new Font("Algerian",
Font.PLAIN, 45));
                lblNewLabel_1_2.setIcon(null);
lblNewLabel_1_2.setBackground(new Color(240, 240,
240));
                lblNewLabel_1_2.setBounds(304, 541, 500, 218);
contentPane.add(lblNewLabel_1_2);
                                        JLabel lblNewLabel_1_2_1 = new JLabel("Donate
Blood\"");
                                        lblNewLabel_1_2_1.setForeground(Color.WHITE);
```

```
                                    lblNewLabel_1_2_1.setFont(new Font("Algerian",
Font.PLAIN, 45));

 lblNewLabel_1_2_1.setBackground(UIManager.getColor("Butt on.background"));
         lblNewLabel_1_2_1.setBounds(720, 627, 500, 218);
contentPane.add(lblNewLabel_1_2_1);                  JLabel
lblNewLabel_1_1_1 = new JLabel("");
                                    lblNewLabel_1_1_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\b19.jpg"));

 lblNewLabel_1_1_1.setBackground(UIManager.getColor("Butt on.background"));
         lblNewLabel_1_1_1.setBounds(355, -137, 703, 794);
contentPane.add(lblNewLabel_1_1_1);                  JLabel lblNewLabel_1_1
= new JLabel("");
                                    lblNewLabel_1_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\back 3.jpg"));
lblNewLabel_1_1.setBackground(UIManager.getColor("Button
.background"));
                lblNewLabel_1_1.setBounds(786, -147, 843, 1065);
contentPane.add(lblNewLabel_1_1);   JLabel lblNewLabel_1 = new JLabel("");
                                     lblNewLabel_1.setBackground(new Color(240, 240,
240));
                                    lblNewLabel_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\back 3.jpg"));
         lblNewLabel_1.setBounds(10, -73, 1186, 1065);       contentPane.add(lblNewLabel_1);




                                    }
}
```

UPDATE DONOR.JAVA

```
import    java.awt.EventQueue;
import
java.awt.event.ActionEvent;
```

```
import
java.awt.event.ActionListener;
import      java.sql.Connection;
import
java.sql.DriverManager;
import
java.sql.PreparedStatement;
import java.sql.SQLException;



import       javax.swing.ImageIcon;
import            javax.swing.JButton;
import javax.swing.JFrame; import
javax.swing.JOptionPane;    import
javax.swing.JPanel;              import
javax.swing.JTextField;       import
javax.swing.SwingUtilities; import
javax.swing.border.EmptyBorder;
import
Project.ConnectionProvider
;     import     java.awt.Font;
import   javax.swing.JLabel;
import
javax.swing.JSeparator;

public class updateDonar extends JFrame {

          private static final long serialVersionUID = 1L;         private
JPanel contentPane;
```

```
                /**
        * Launch the application.
                */


                /**
        * Create the frame.
                */
                  private JButton updateButton;

        private JTextField textField;
private JLabel emailLabel;   private JTextField
textField_1;   private JLabel bloodGroupLabel;
private JTextField textField_2;   private
JLabel lblNewLabel_1;

                    public updateDonar() {
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);     setBounds(100,
100, 710, 541);                contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

            setContentPane(contentPane);          contentPane.setLayout(null);   updateButton
= new JButton("Update");       updateButton.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Details.png"));
// Change the path to your icon
                                updateButton.setFont(new Font("Tahoma", Font.BOLD,
16));          updateButton.setBounds(62, 423, 156, 68);
getContentPane().add(updateButton);

                                JLabel nameLabel = new JLabel("Name: ");
                                 nameLabel.setFont(new Font("Tahoma", Font.BOLD,
```

```
16));          nameLabel.setBounds(37, 83, 134, 111);
contentPane.add(nameLabel);

                                textField = new JTextField();
            textField.setFont(new Font("Tahoma", Font.PLAIN, 16));
textField.setBounds(299, 104, 372, 68);              contentPane.add(textField);

                                emailLabel = new JLabel("Email: ");
                                  emailLabel.setFont(new Font("Tahoma", Font.BOLD,
16)); emailLabel.setBounds(37, 197, 140, 111);
                                contentPane.add(emailLabel);

            textField_1 = new JTextField();                 textField_1.setFont(new
Font("Tahoma", Font.PLAIN,
16));          textField_1.setBounds(299, 218, 372, 68);
contentPane.add(textField_1);
                        bloodGroupLabel = new JLabel("Blood Group: ");
bloodGroupLabel.setFont(new Font("Tahoma",
Font.BOLD, 16));             bloodGroupLabel.setBounds(37,
302, 140, 111);
contentPane.add(bloodGroupLabel);

            textField_2 = new JTextField();                 textField_2.setFont(new
Font("Tahoma", Font.PLAIN,
16));          textField_2.setBounds(299, 323, 372, 68);
contentPane.add(textField_2);

            JButton btnClose = new JButton("Close"); btnClose.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\Exit
application.png"));
      btnClose.addActionListener(new ActionListener() {          public void
actionPerformed(ActionEvent e) {          setVisible(false);
```

```
                }
        });
                btnClose.setFont(new Font("Tahoma", Font.BOLD, 16));
btnClose.setBounds(363, 423, 169, 68);                contentPane.add(btnClose);


                JSeparator separator = new JSeparator();  separator.setBounds(10,
405, 721, 31);                    contentPane.add(separator);
                JSeparator separator_1 = new JSeparator();
separator_1.setBounds(10, 83, 721, 21);                contentPane.add(separator_1);


                JLabel lblNewLabel = new JLabel("UPDATE DONAR DETAILS");
lblNewLabel.setFont(new Font("Algerian", Font.BOLD,
28));
                                        lblNewLabel.setBounds(163, 20, 350, 53);
                                          contentPane.add(lblNewLabel);
lblNewLabel_1 = new JLabel("New label");
lblNewLabel_1.setIcon(new
ImageIcon("D:\\devi\\countdown\\countdown\\newyear\\src\\assets\\all                page
background image.png"));
                lblNewLabel_1.setBounds(0, -99, 769, 707);
contentPane.add(lblNewLabel_1);


                                          updateButton.addActionListener(new ActionListener() {
                                         @Override
                    public void actionPerformed(ActionEvent e) {
updateDonorInformation();
                                            }
                                        });


                                         setVisible(true);


                        }
```

```
                    private void updateDonorInformation() {
        String name = textField.getText();
        String email = textField_1.getText();
        String bloodGroup = textField_2.getText();


        Connection conn = null;
PreparedStatement stmt = null;
try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bbms", "root",
"devi");


            String sql = "UPDATE Donors SET donor_email = ?, blood_group = ? WHERE
donor_name = ?";


                        stmt = conn.prepareStatement(sql);
              stmt.setString(1, email);
stmt.setString(2, bloodGroup); stmt.setString(3,
name);
              int rowsUpdated = stmt.executeUpdate();


              if (rowsUpdated > 0) {
                 JOptionPane.showMessageDialog(this, "Donor information updated successfully.");
              } else {
                 JOptionPane.showMessageDialog(this, "Error: Unable to update donor information.");
              }



    }
                        catch (SQLException ex)
{             ex.printStackTrace();
          JOptionPane.showMessageDialog(this, "Error: Unable to update donor information.");


}
```

```
fi
n
al
ly
{
tr
y
{               if (stmt != null) {
stmt.close();
            }
            if (conn != null) {
conn.close();
            }
                        } catch (SQLException ex)
{               ex.printStackTrace();
        }
    }


                            }
        public static void main(String[] args) {        SwingUtilities.invokeLater(() ->
new updateDonar());
   }
}
```

4.3 OUTPUT IMAGES :

ADDING DONOR AND VIEWING DONOR DETAILS:

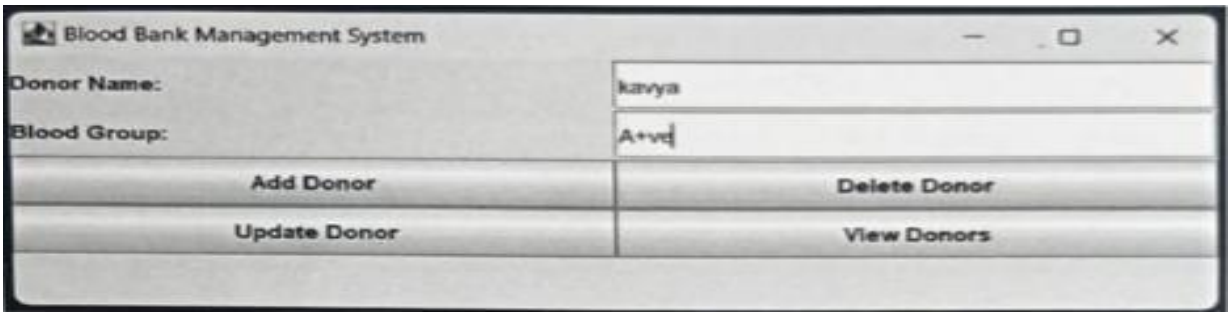


FIG.4.3.1 ADDING DONOR DETAIL

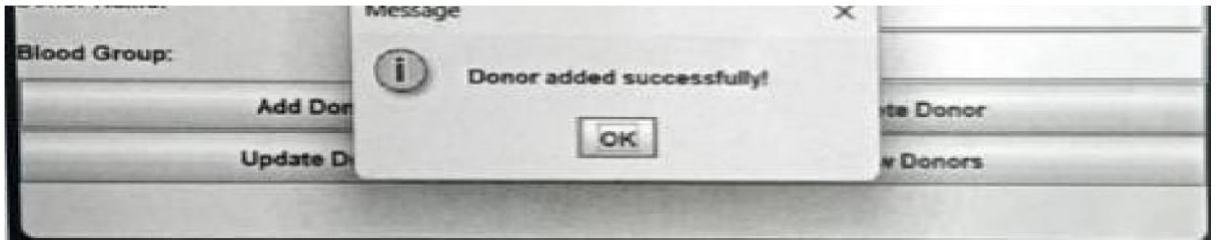


FIG.4.3.2 DONOR DETAIL ADDED SUCCESSFULLY

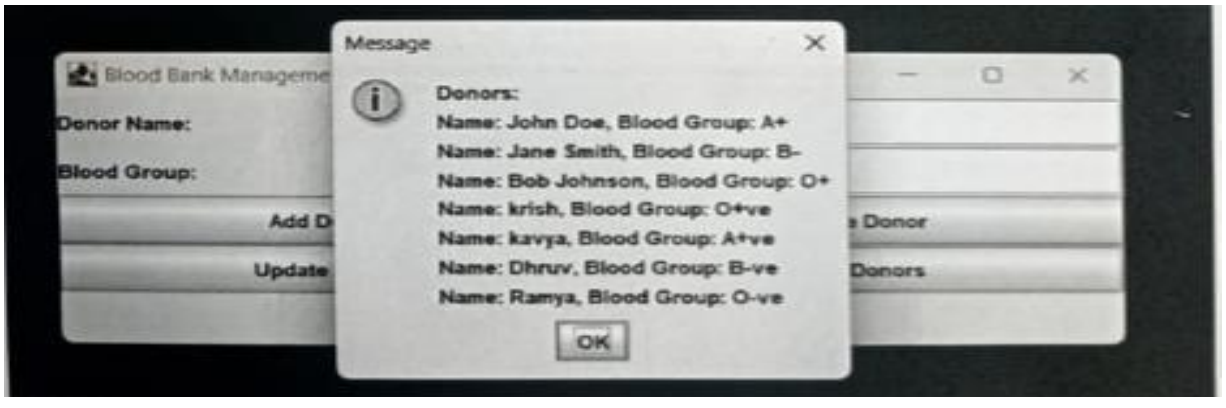


FIG.4.3.3 VIEWING DETAILS OF DONOR AFTER ADDITION

» DELETING DONOR AND VIEWING DONOR DETAILS:

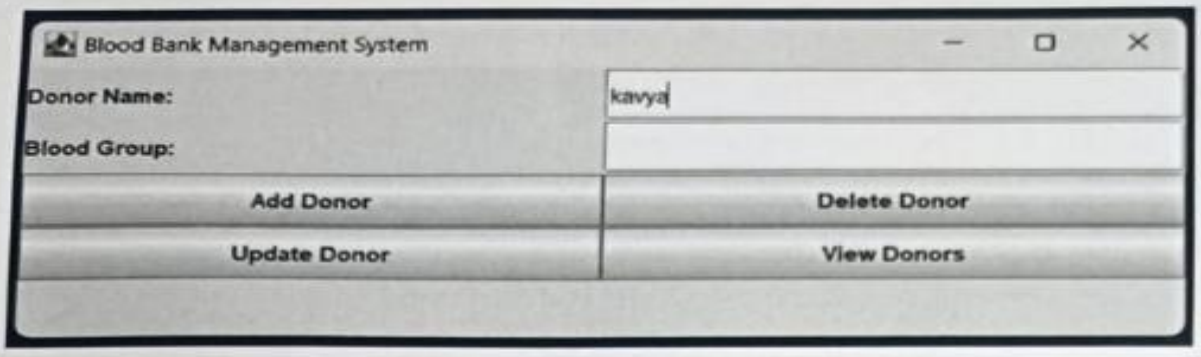


Fig.4.3.4 DELETING DONOR DETAIL

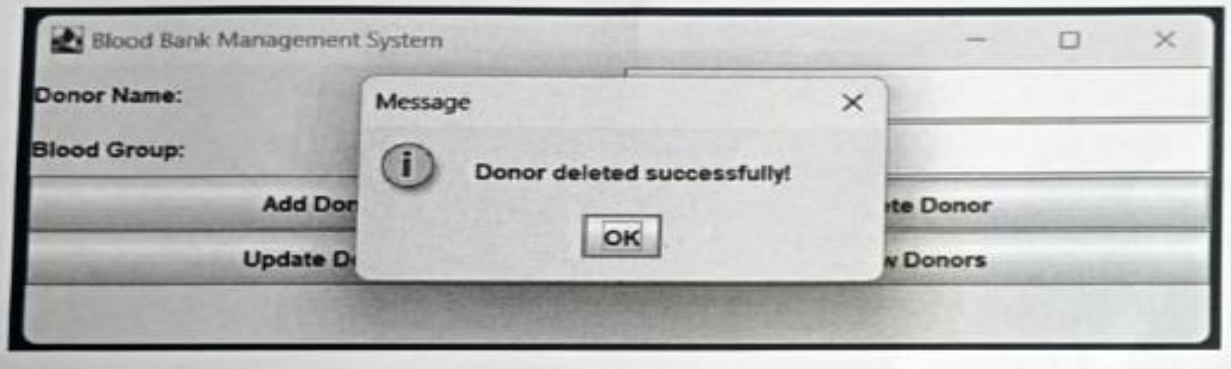


Fig.4.3.5 DONOR DELETED SUCCESSFULLY

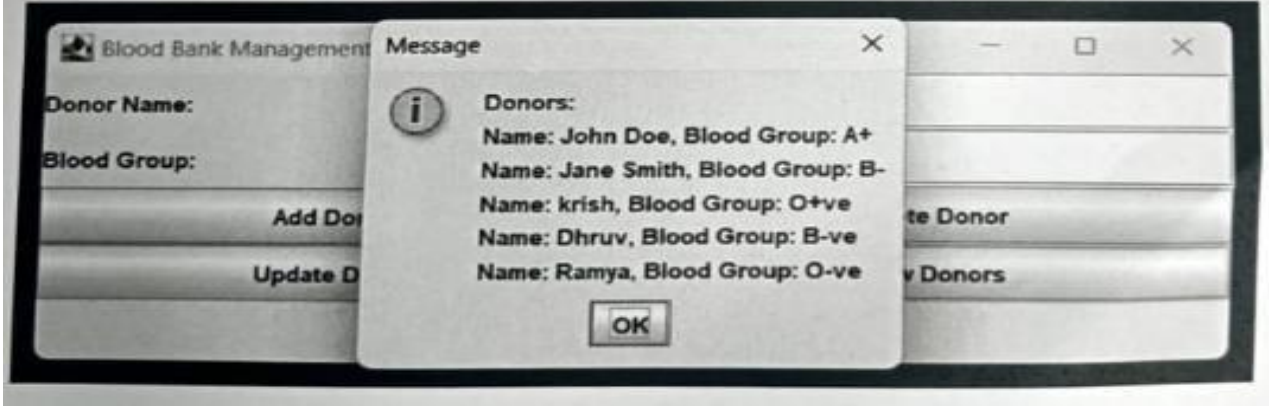


Fig.4.3.6 VIEWING DETAILS OF DONOR AFTER DELETION

4.4 IMPLEMENTATIONS:

To implement a **Blood Bank Management System (BBMS)** in **Java**, we will go through the steps of **setting up the database**, creating the necessary **Java classes**, and implementing the **backend logic** for managing donors, recipients, inventory, and transactions.
Here is a full implementation of the **Blood Bank Management System** in Java using **JDBC** for database interaction. This implementation will cover essential features such as:

Donor Management

Blood Inventory Management

Recipient Management

Transaction (Transfusion) Management

Steps for Implementation

1. **Set up Database**:
   - Install and set up a relational database.
   - Create the required tables as shown in the schema.
2. **Back-End Development**:
   - Implement the business logic and APIs for donor registration, blood inventory management, and transfusions.
3. **Front-End Development**: ○ Create a simple web interface for users to interact with the system.
   - Use JavaScript (AJAX/fetch) to interact with back-end APIs.
4. **Testing**:
   - Test the system for bugs and ensure all features work correctly, like registering donors, updating inventory, and processing transactions.
5. **Deployment**: ○ Deploy the application on a server using a platform like **Heroku**, **AWS**, or **Azure**. ○ Make sure the database is accessible and secure.

5.CONCLUSION:

The Blood Bank Management System (BBMS) provides an efficient and organized solution for managing blood donations, blood inventory, and transfusions. It simplifies the process of registering blood donors, tracking blood types and quantities in the inventory, managing recipient requests, and recording transfusion transactions.

By implementing a database-driven approach using Java and MySQL (or any relational database), this system allows for seamless tracking, updating, and reporting of critical information, ensuring the efficient operation of a blood bank**.**

6.REFERENCE:

Books:

1. Head First Java by Kathy Sierra and Bert Bates ○ Beginner-friendly Java programming guide.

2. Effective Java by Joshua Bloch ○ Best practices for writing clean, efficient, and maintainable Java code.
3. Java: The Complete Reference by Herbert Schildt ○ A comprehensive guide to Java for deep understanding of language and libraries.

Online Resources:

1. Oracle Java Tutorials ○ Official guide to learning Java programming and core libraries.
2. Geeks for Geeks Java Tutorials ○ Tutorials on Java programming, JDBC, and collections.
3. W3Schools MySQL Tutorial ○ Beginner-friendly tutorial on MySQL for designing and managing databases.
4. Java Code Geeks JDBC Tutorials ○ Guide to Java Database Connectivity (JDBC) for database operations.

○