RAJALAKSHMI ENGINEERING

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602
105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23333 OBJECT ORIENTED PROGRAMING

Laboratory Record Note

Name : . SWETHA.A . . . .

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

College Roll No. : . . . . 2116231001226. . . . .

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

Academic Year : . . . . 2024-2025 . . . .

RAJALAKSHMI ENGINEERING

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name : . . . . . SWETHA.A. . . . .

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

Register No. 2116231001226

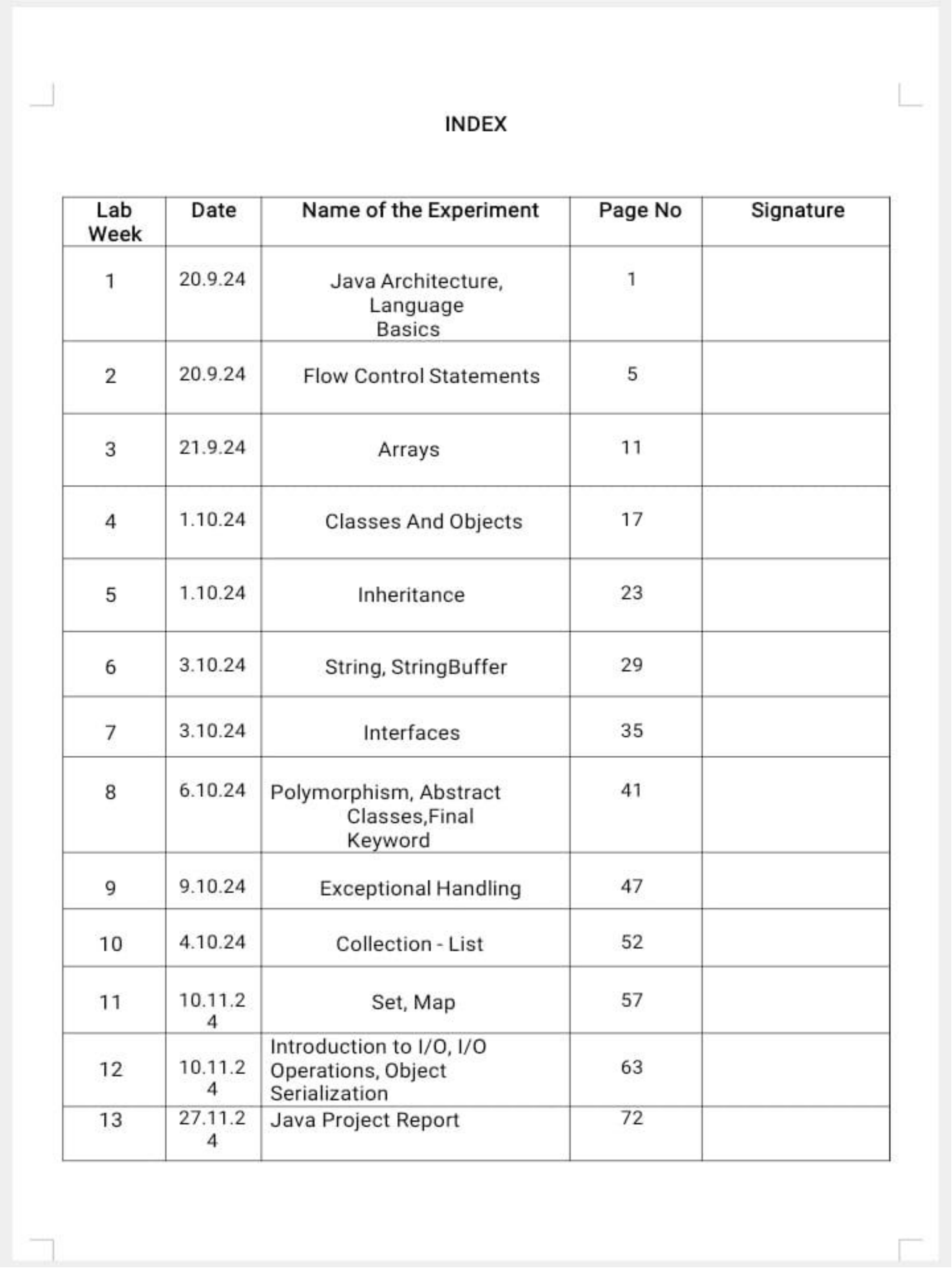
Certified that this is the bonafide record of work done by the above student in the CS23333 –Object Oriented Programming using JAVAduring the year 2024 - 2025.

Signature of Faculty in-charge

Submitted for the Practical Examination held on . . . .

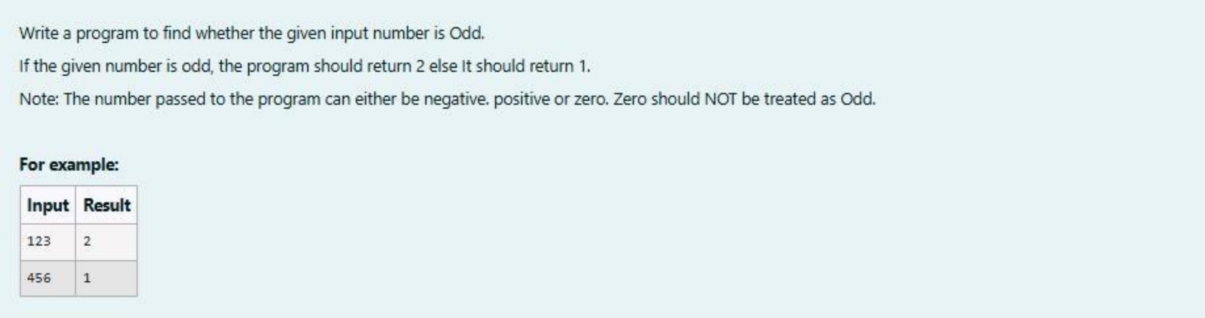
Internal Examiner **External Examiner**

INDEX



INDEX

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.

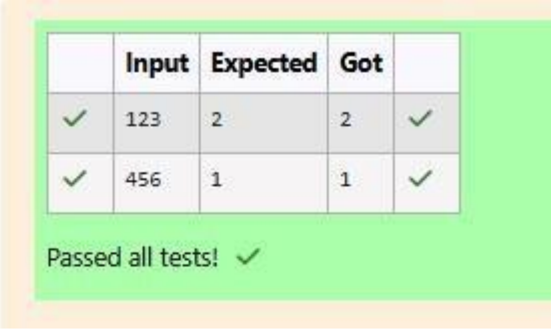
For example:

Input	Result
123	2
456	1

SOLUTION :

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

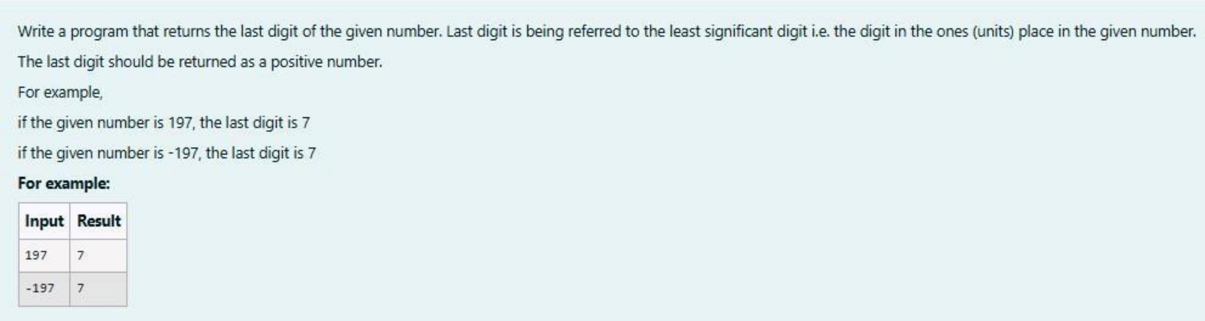
OUTPUT :



	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

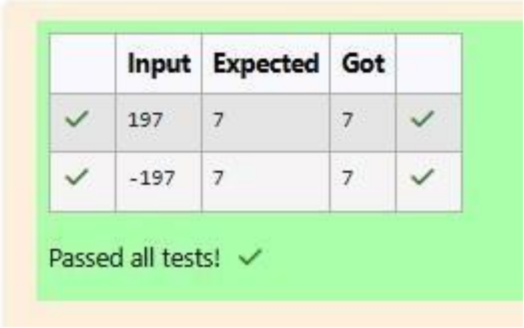
For example:

Input	Result
197	7
-197	7

SOLUTION :

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

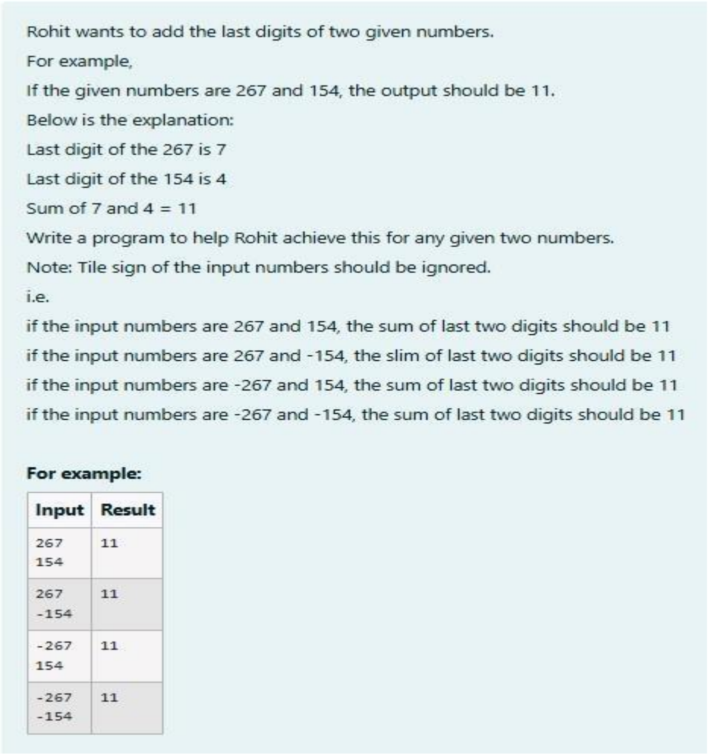
OUTPUT :



	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

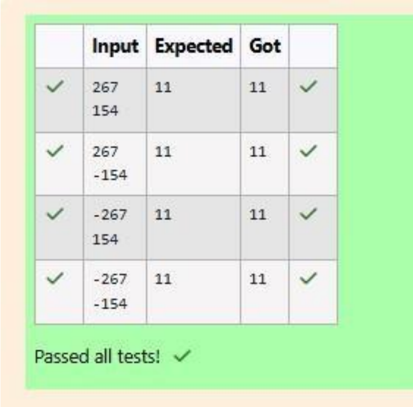
For example:

Input	Result
267 154	11
267 -154	11
-267 154	11
-267 -154	11

SOLUTION :

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

OUTPUT:

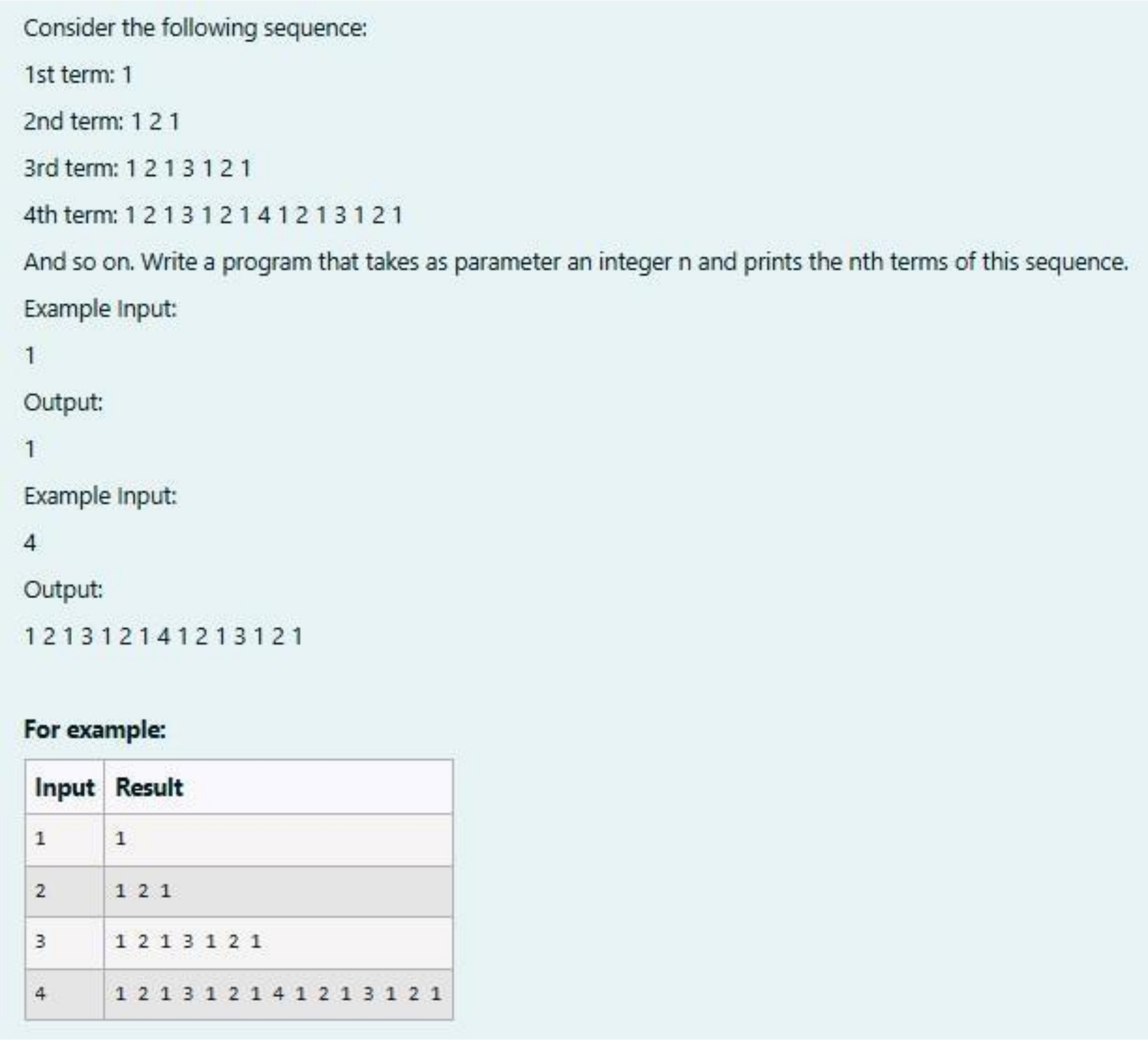


	Input	Expected	Got	
✓	267 154	11	11	✓
✓	267 -154	11	11	✓
✓	-267 154	11	11	✓
✓	-267 -154	11	11	✓

Passed all tests! ✓

Lab-02-Flow Control Statements

1.



Consider the following sequence:

1st term: 1

2nd term: 1 2 1

3rd term: 1 2 1 3 1 2 1

4th term: 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

And so on. Write a program that takes as parameter an integer n and prints the nth terms of this sequence.

Example Input:

1

Output:

1

Example Input:

4

Output:

1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

For example:

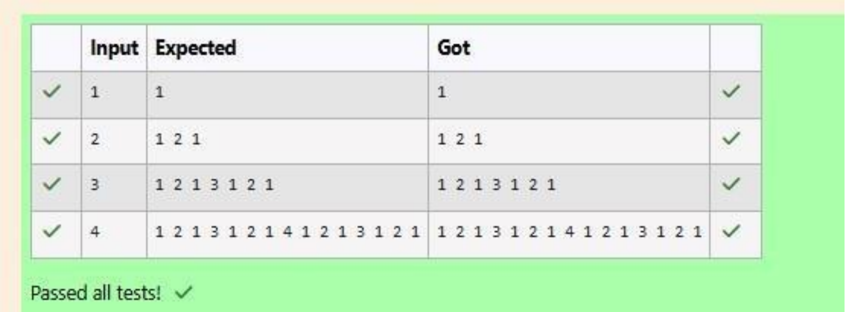
Input	Result
1	1
2	1 2 1
3	1 2 1 3 1 2 1
4	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

SOLUTION :

```
import java.util.Scanner; public class SequenceGenerator{ public static void
main(String[]args){ Scanner S = new
Scanner(System.in); int n = S.nextInt();
      String term = generateTerm(n);
      System.out.print(term);
   }
   private static String generateTerm(int n){ if (n==1){ return "1";
      }
      String prevTerm = generateTerm (n-1);
      StringBuilder currentTerm = new StringBuilder(prevTerm);
```

```
        currentTerm.append(" " + n + "
    "); currentTerm.append(prevTerm);
    return currentTerm.toString(); }
}
```

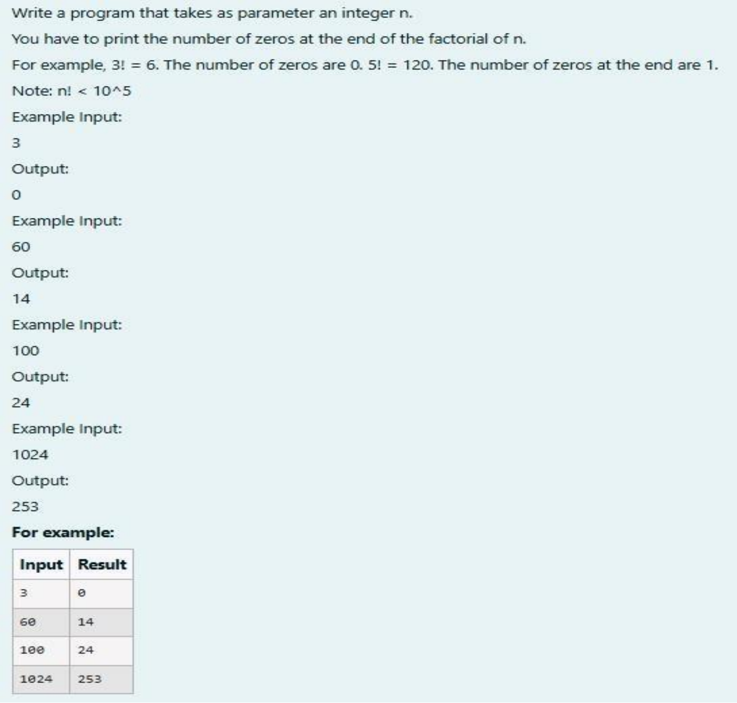
OUTPUT :



	Input	Expected	Got	
✓	1	1	1	✓
✓	2	1 2 1	1 2 1	✓
✓	3	1 2 1 3 1 2 1	1 2 1 3 1 2 1	✓
✓	4	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1	✓

Passed all tests! ✓

2.



Write a program that takes as parameter an integer n.

You have to print the number of zeros at the end of the factorial of n.

For example, 3! = 6. The number of zeros are 0. 5! = 120. The number of zeros at the end are 1.

Note: n! < 10^5

Example Input:

3

Output:

0

Example Input:

60

Output:

14

Example Input:

100

Output:

24

Example Input:

1024

Output:

253

For example:

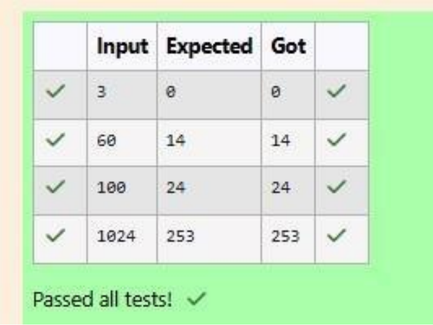
Input	Result
3	0
60	14
100	24
1024	253

SOLUTION :

```
// Java program to count trailing 0s in n!
import java.io.*; import java.util.Scanner; class prog {
    // Function to return trailing
    // 0s in factorial of n
    static int findTrailingZeros(int n)
    { if (n < 0) // Negative Number Edge Case return -1;
```

```
        // Initialize result

        int count=0;
        // Keep dividing n by powers // of
        5 and update count for (int i =
        5; n / i >= 1; i*=5          ){ count
        += n / i;
 } return count;
    }

    // Driver Code
    public static void main(String[] args)
    {
        Scanner sc= new Scanner(System.in);
        int n=sc.nextInt(); int
    res=findTrailingZeros(n);
    System.out.println(res); }
 }
```

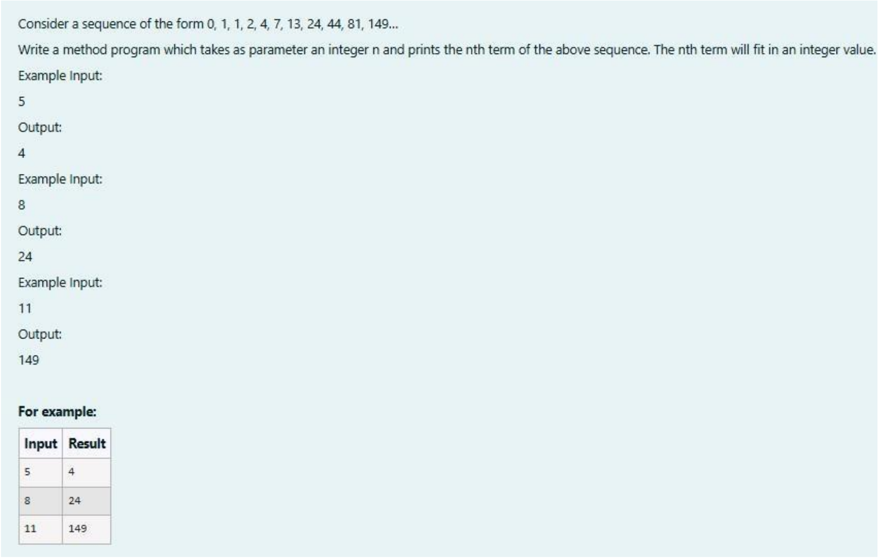
OUTPUT :



	Input	Expected	Got	
✓	3	0	0	✓
✓	60	14	14	✓
✓	100	24	24	✓
✓	1024	253	253	✓

Passed all tests! ✓

3.



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.

Example Input:

5

Output:

4

Example Input:

8

Output:

24

Example Input:

11

Output:

149

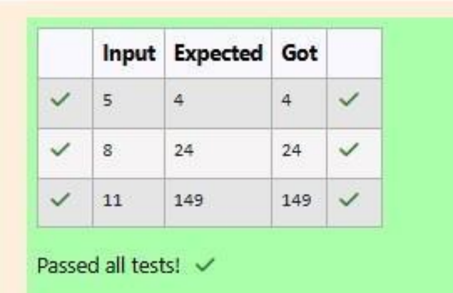
For example:

Input	Result
5	4
8	24
11	149

SOLUTION :

```
import java.util.Scanner;
class fibo3{ int a; int b;
int c; fibo3(int a,int b,int
c){ this.a = a; this.b = b;
this.c = c;
    }
    int nth(int x){
        if (x == 1){
        return 0;
        }
        else if(x == 2 && x == 3) return
            1;
        else{ int temp1,temp2,temp;
            int count = 4; while(x >=
            count){ temp =
            this.a+this.b+this.c; temp1 =
            this.c; this.c = temp; temp2
            = this.b; this.b = temp1;
            this.a = temp2; count++;
            }
            return this.c;
        }
    }
}
public class Main{ public static void
    main(String[] args){ Scanner s = new
    Scanner(System.in); int t =
        s.nextInt(); fibo3 r = new
        fibo3(0,1,1);
        System.out.print(r.nth(t)); }
}
```

OUTPUT :

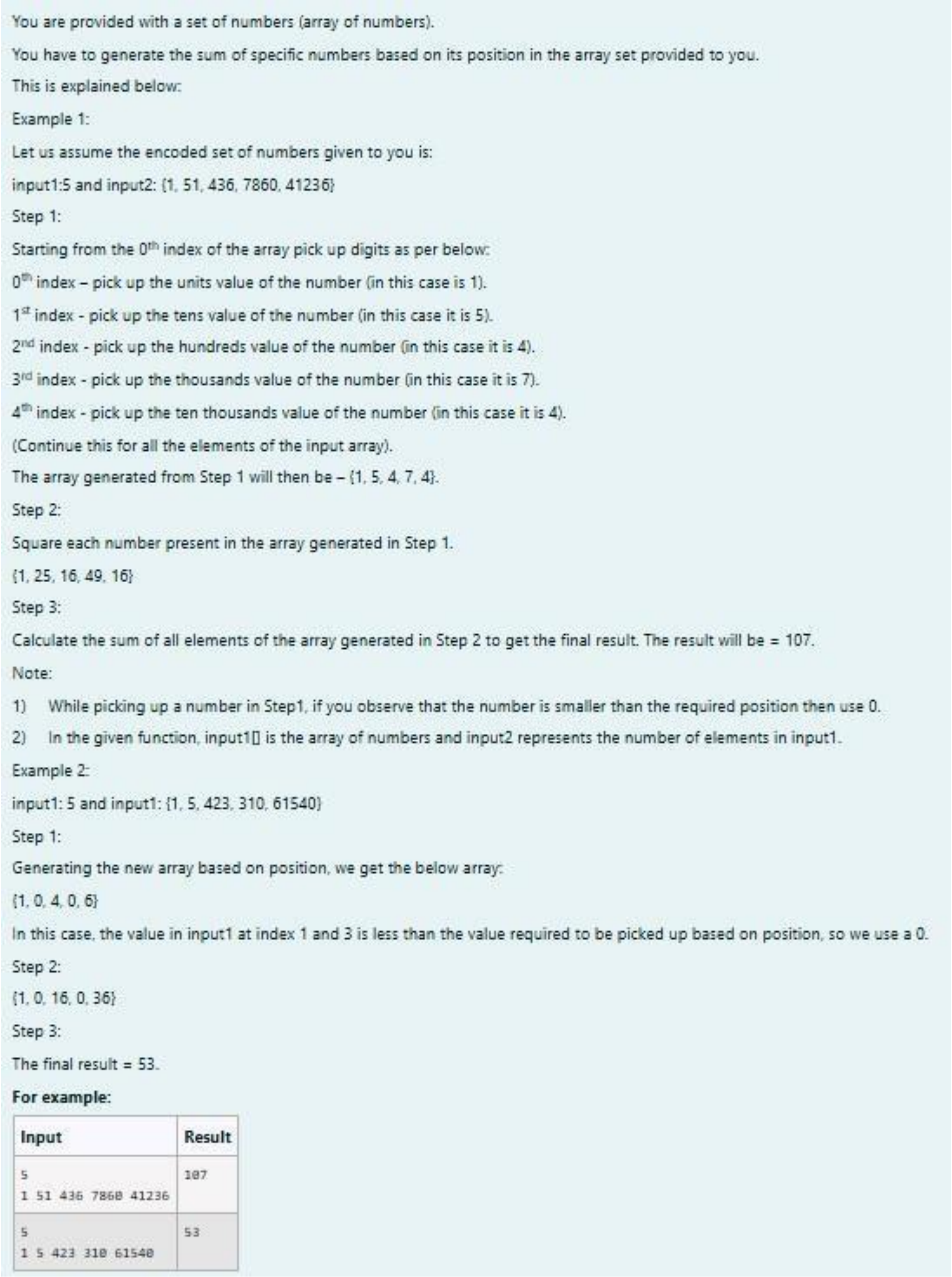


	Input	Expected	Got	
✓	5	4	4	✓
✓	8	24	24	✓
✓	11	149	149	✓

Passed all tests! ✓

Lab-03-Arrays

1.



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 0th index of the array pick up digits as per below:

0th index – pick up the units value of the number (in this case is 1).

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

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

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

4th 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 input1.

Example 2:

input1: 5 and input1: {1, 5, 423, 310, 61540}

Step 1:

Generating the new array based on position, we get the below array:

{1, 0, 4, 0, 6}

In this case, the value in input1 at index 1 and 3 is less than the value required to be picked up based on position, so we use a 0.

Step 2:

{1, 0, 16, 0, 36}

Step 3:

The final result = 53.

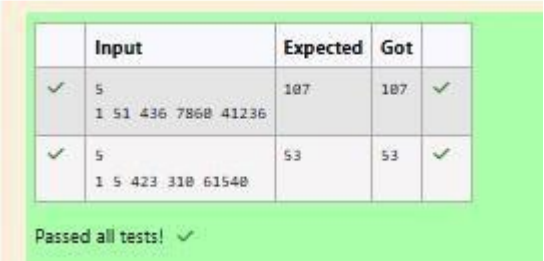
For example:

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

SOLUTION :

```
import java.util.Scanner; public class
digit{ public static void
main(String[]args){
        Scanner scanner =new Scanner(System.in);
        int size =scanner.nextInt();
        int[]inpar=new int[size];
        for(int i=0;i<size;i++){
        inpar[i]=scanner.nextInt();
        }
        int[]dig=new int[size];
        for(int i=0;i<size;i++){
        int num=inpar[i];
        if(i==0){
        dig[i]=num%10;
                }
                else if (i==1){
                    dig[i]=(num/10)%10;
                }
                else if(i==2){
                    dig[i]=(num/100)%10;
                }
                else if(i==3){
                    dig[i]=(num/1000)%10;

                }
                else if(i==4){
                    dig[i]=(num/10000)%10;
                } else{
                dig[i]=0;
                } } int
        fin=0; for(int
        digi:dig){
        fin+=digi*digi;
        }
        System.out.print(fin);
    }
}
```

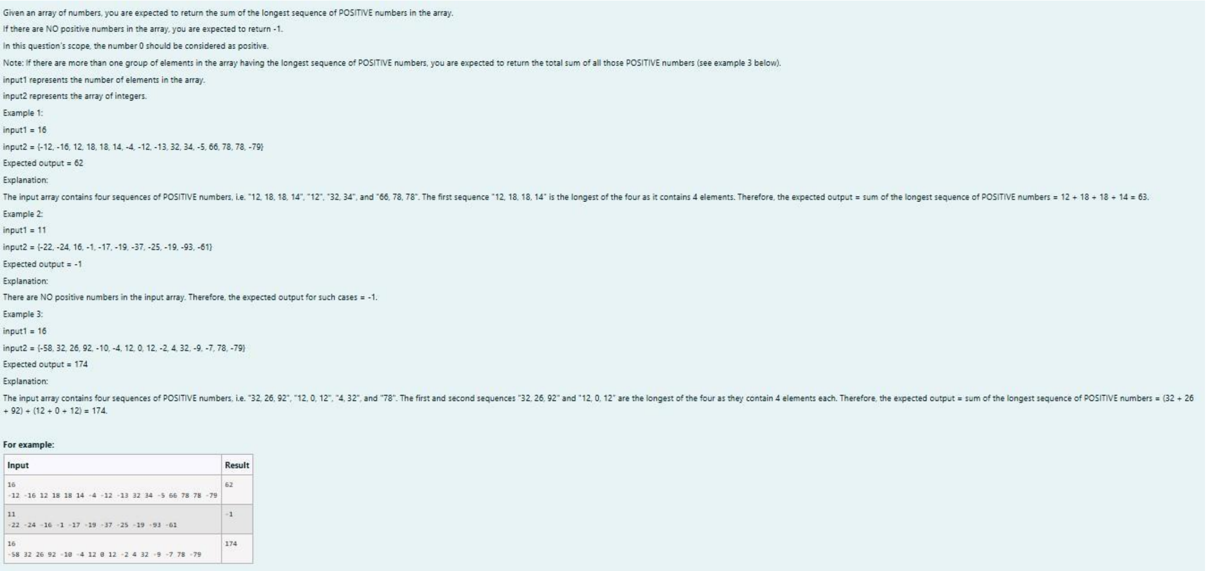
OUTPUT :



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

Passed all tests! ✓

2.



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.

Example 2:

input1 = 11

input2 = {-22, -24, 16, -1, -17, -19, -37, -25, -19, -93, -61}

Expected output = -1

Explanation:

There are NO positive numbers in the input array. Therefore, the expected output for such cases = -1.

Example 3:

input1 = 16

input2 = {-58, 32, 26, 92, -10, -4, 12, 0, 12, -2, 4, 32, -9, -7, 78, -79}

Expected output = 174

Explanation:

The input array contains four sequences of POSITIVE numbers, i.e. "32, 26, 92", "12, 0, 12", "4, 32", and "78". The first and second sequences "32, 26, 92" and "12, 0, 12" are the longest of the four as they contain 4 elements each. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = (32 + 26 + 92) + (12 + 0 + 12) = 174.

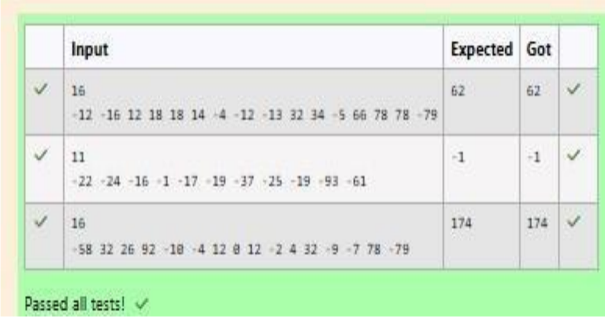
For example:

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

SOLUTION :

```
import java.util.Scanner; public class
longdig{ public static void
main(String[]args){ Scanner sc=new
Scanner(System.in);
    int n=sc.nextInt();
    int c = 1,v,seqtemp = 0,seq = 0,countmax = 0;
    int count = 0; while(c <= n){ v = sc.nextInt();
    if(v >= 0){ countmax= countmax + v;
      seqtemp++;
      }
      else{
        seqtemp = 0;
        countmax = 0;
      }
      if(seqtemp > seq ){
        seq = seqtemp;
        count = countmax;
      }
      else if (seq == seqtemp){
        count = count + countmax;
      }
    c++; }
    if (count == 0)
      System.out.print(-1);
    else
      System.out.print(count);
}
}
```

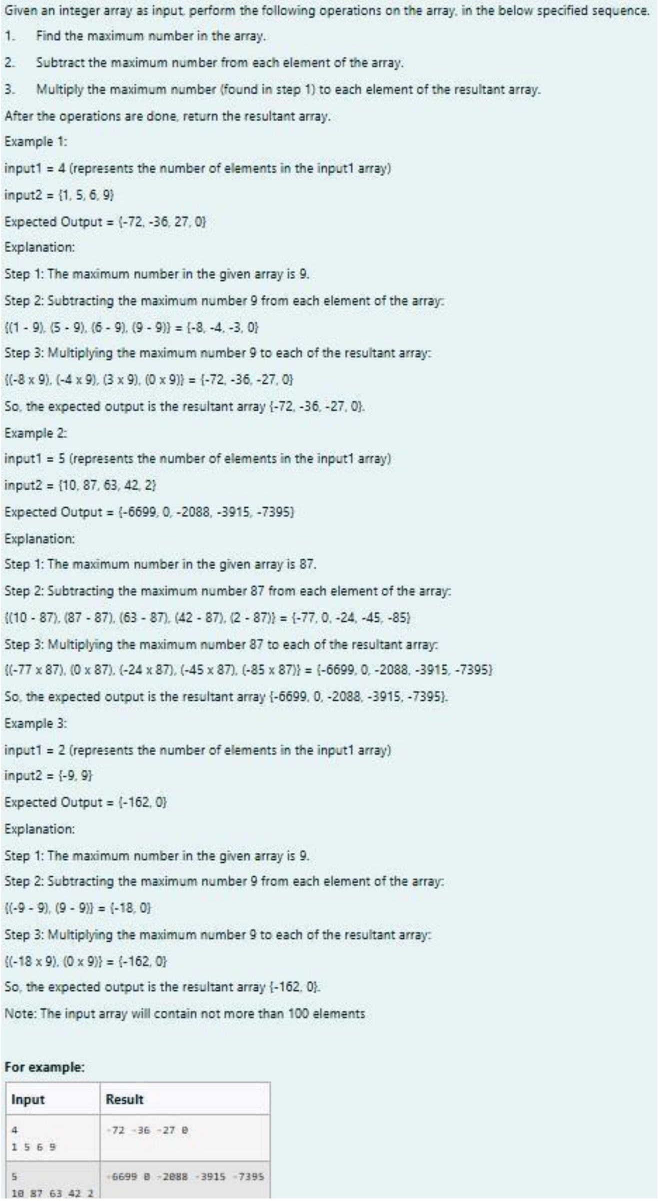
OUTPUT :



	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! ✓

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}.

Example 2:

input1 = 5 (represents the number of elements in the input1 array)

input2 = {10, 87, 63, 42, 2}

Expected Output = {-6699, 0, -2088, -3915, -7395}

Explanation:

Step 1: The maximum number in the given array is 87.

Step 2: Subtracting the maximum number 87 from each element of the array:

{(10 - 87), (87 - 87), (63 - 87), (42 - 87), (2 - 87)} = {-77, 0, -24, -45, -85}

Step 3: Multiplying the maximum number 87 to each of the resultant array:

{(-77 x 87), (0 x 87), (-24 x 87), (-45 x 87), (-85 x 87)} = {-6699, 0, -2088, -3915, -7395}

So, the expected output is the resultant array {-6699, 0, -2088, -3915, -7395}.

Example 3:

input1 = 2 (represents the number of elements in the input1 array)

input2 = {-9, 9}

Expected Output = {-162, 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:

{(-9 - 9), (9 - 9)} = {-18, 0}

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

{(-18 x 9), (0 x 9)} = {-162, 0}

So, the expected output is the resultant array {-162, 0}.

Note: The input array will contain not more than 100 elements

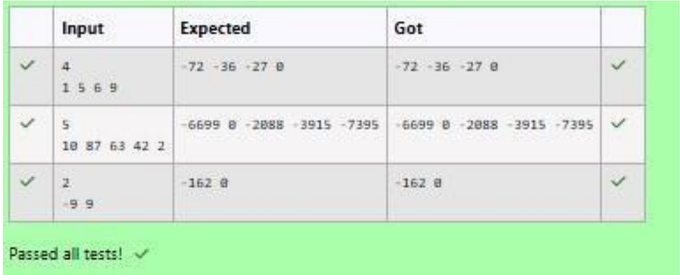
For example:

Input	Result
4 1 5 6 9	-72 -36 -27 0
5 10 87 63 42 2	-6699 0 -2088 -3915 -7395

SOLUTION :

```
import java.util.Scanner; public
class    res{    public    static
int[]pa(int[]arr){
      int maxs=Integer.MIN_VALUE;
      for          (int          num:arr){
      if(num>maxs){
         maxs=num;
         }
      }
      for(int i=0;i<arr.length;i++){
         arr[i]=(arr[i]maxs)*maxs;
      }
      return arr;
   }
   public static void main(String[]args){
      Scanner scanner =new Scanner
      (System.in); int n=scanner.nextInt();
      int[]arr=new int[n]; for(int i=0;i<n;i++){
      arr[i]=scanner.nextInt();
      }
      int[]res=pa(arr);
      for(int i=0;i<n;i++){
         System.out.print(res[i]+" ");
      }
      scanner.close();
   }
}
```

OUTPUT :

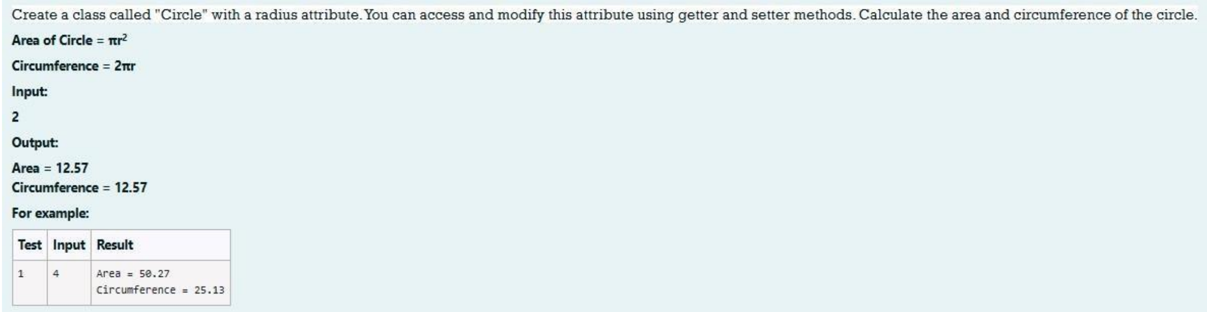


	Input	Expected	Got	
✓	4 1 5 6 9	-72 -36 -27 0	-72 -36 -27 0	✓
✓	5 10 87 63 42 2	-6699 0 -2088 -3915 -7395	-6699 0 -2088 -3915 -7395	✓
✓	2 -9 9	-162 0	-162 0	✓

Passed all tests! ✓

Lab-04-Classes and Objects

1.



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 = $2\pi r$

Input:

2

Output:

Area = 12.57
Circumference = 12.57

For example:

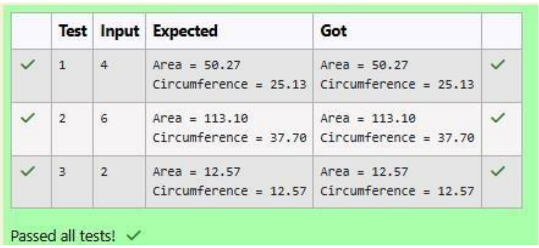
Test	Input	Result

1	4	Area = 50.27
		Circumference = 25.13

SOLUTION :

```
import java.io.*; import java.util.Scanner;
class
Circle
{ private double radius; public
   Circle(double radius){
      // set the instance variable radius
     this.radius =radius;
        } public void setRadius(double
   radius){
      // set the radius
      this.radius=radius;

   }
public double getRadius()       {
      // return the radius return
      radius;

   }
   public double calculateArea() { // complete the below statement
      return Math.PI*radius*radius;

   }
public double calculateCircumference()       {
      // complete the statement 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()));
      // invoke the calculatecircumference method
      System.out.println("Circumference = "+String.format("%.2f" ,
c.calculateCircumference()));

      sc.close();
   }
}
```

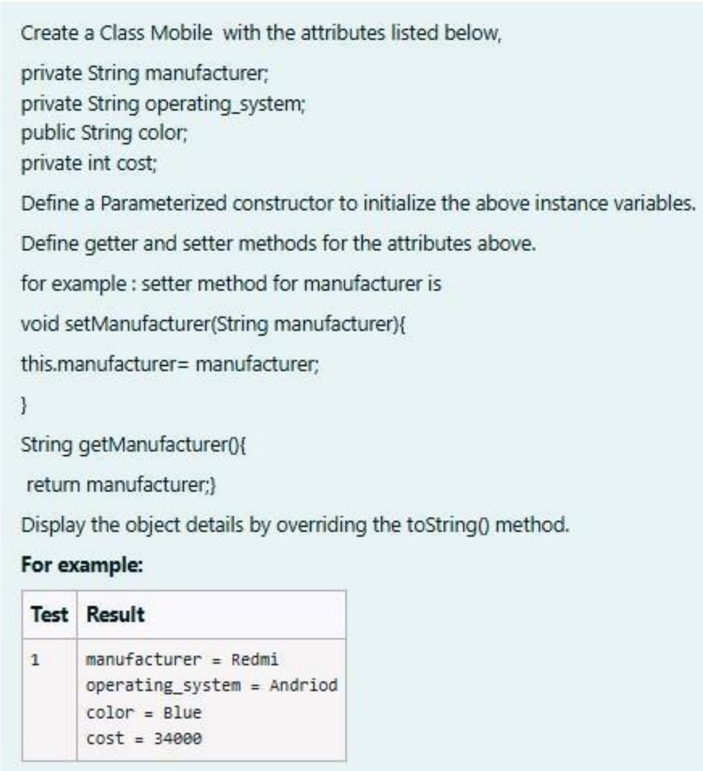
OUTPUT :



	Test	Input	Expected	Got	
✓	1	4	Area = 50.27 Circumference = 25.13	Area = 50.27 Circumference = 25.13	✓
✓	2	6	Area = 113.10 Circumference = 37.70	Area = 113.10 Circumference = 37.70	✓
✓	3	2	Area = 12.57 Circumference = 12.57	Area = 12.57 Circumference = 12.57	✓

Passed all tests! ✓

2.



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:

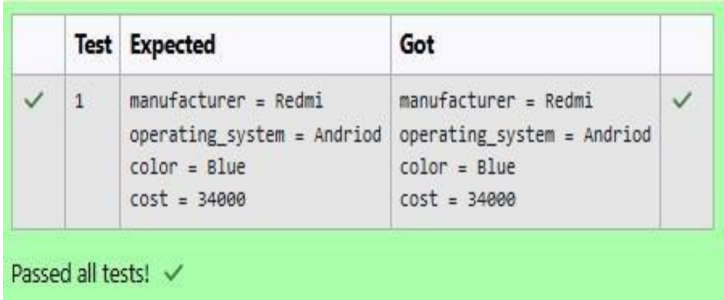
Test	Result

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

SOLUTION :

```
public   class   mobile{
   private   String   man;
   private    String    os;
   public    String    clr;
   private int cost;
   public  mobile(String  man,String  os,String  clr,int  cost){
      this.man=man; this.os=os; this.clr=clr; this.cost=cost;
      } public String toString(){ return "manufacturer = "+man+"\n"+"operating_system =
"+os+"\n"+"color = "+ clr+"\n"+"cost = "+cost;
      }
      public static void main(String[]args){
         mobile mobile=new
      mobile("Redmi","Andriod","Blue",34000);
      System.out.println(mobile); }
}
```

OUTPUT :

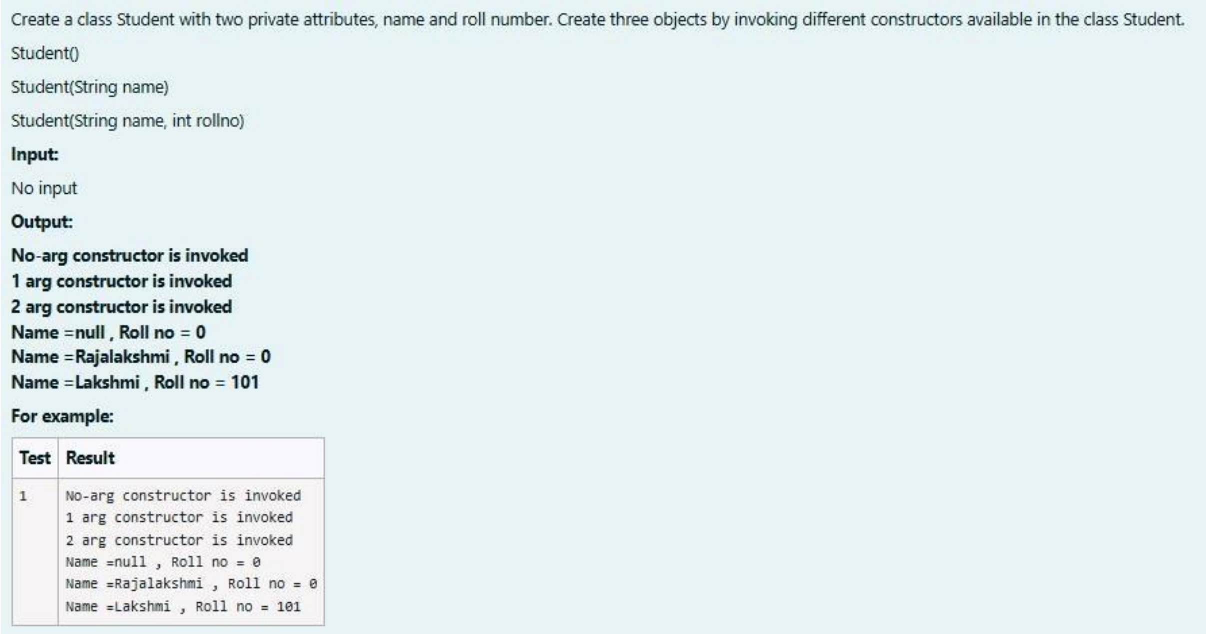


	Test	Expected	Got	

✓	1	manufacturer = Redmi
		operating_system = Andriod
		color = Blue
		cost = 34000
			manufacturer = Redmi
			operating_system = Andriod
			color = Blue
			cost = 34000
				✓

Passed all tests! ✓

3.



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)

Input:

No input

Output:

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

For example:

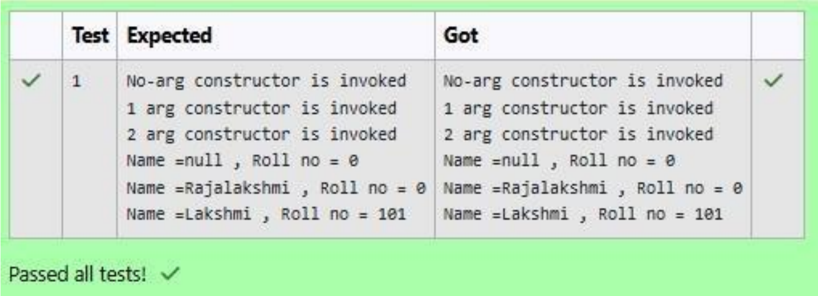
Test	Result

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

SOLUTION :

```
public class stud{ private String name; private int roll; public stud(){
       System.out.println("No-arg constructor is invoked"); name=null; roll=0;

   } public stud(String name){
       System.out.println("1 arg constructor is invoked"); this.name=name; roll=0;
}
public stud(String name,int roll){
   System.out.println("2 arg constructor is invoked"); this.name=name;
   this.roll=roll;

   }

public static void main (String[]args){ stud
         s1=new stud(); stud s2=new
         stud("Rajalakshmi"); stud s3=new
         stud("Lakshmi",101);
         System.out.println("Name ="+s1.name+" , Roll no = "+s2.roll);
         System.out.println("Name ="+s2.name+" , Roll no = "+s2.roll);
         System.out.println("Name ="+s3.name+" , Roll no = "+s3.roll);
      }
   }
```

OUTPUT :



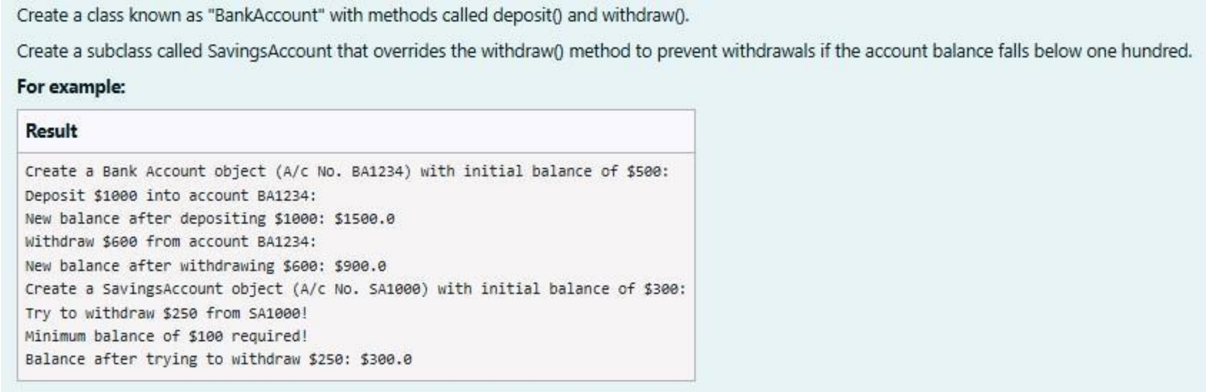
	Test	Expected	Got	

✓	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! ✓

Lab-05-Inheritance

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
```

SOLUTION :

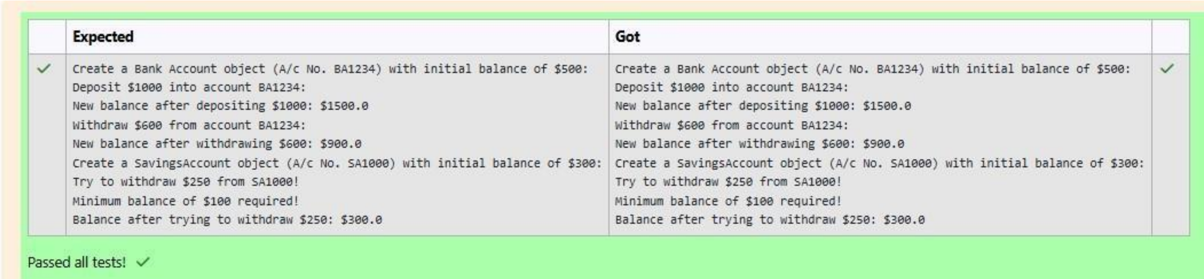
```
class BankAccount {
// Private field to store the account number
private String accountNumber;
// Private field to store the balance
```

```
   private double balance;

   // Constructor to initialize account number and balance
   public BankAccount(String accountNumber,double
   balance){ this.accountNumber=accountNumber;
   this.balance=balance;
   }




   // Method to deposit an amount into the account
   public void deposit(double amount) {
      // Increase the balance by the deposit amount
    balance+=amount;
   }

   // Method to withdraw an amount from the account
   public void withdraw(double amount) {
      // Check if the balance is sufficient for the withdrawal
      if (balance >= amount) {
         // Decrease the balance by the withdrawal amount
         balance -= amount;
      } else {
         // Print a message if the balance is
      insufficient System.out.println("Insufficient
      balance"); }
   }

   // Method to get the current balance
   public double getBalance() { //
   Return the current balance return
      balance;
   }
   public String getAccountNumber(){ return
      accountNumber;
   }
}
class SavingsAccount extends BankAccount {
   // Constructor to initialize account number and balance
   public SavingsAccount(String accountNumber, double balance) {
      // Call the parent class constructor
      super(accountNumber,balance);
   }

   // Override the withdraw method from the parent class
   @Override
```

```
public void withdraw(double amount) {
    // Check if the withdrawal would cause the balance to drop below $100
```

```
        if (getBalance() - amount < 100) {
           // Print a message if the minimum balance requirement is not met
           System.out.println("Minimum balance of $100 required!");
        } else {
           // Call the parent class withdraw method
           super.withdraw(amount);
        }
    }
} public class Main {

    public static void main(String[] args) {
        // Print message to indicate creation of a BankAccount object
        System.out.println("Create a Bank Account object (A/c No. BA1234) with initial
balance of $500:");
        // Create a BankAccount object (A/c No. "BA1234") with initial balance of $500
        BankAccount BA1234 = new BankAccount("BA1234", 500);
        // Print message to indicate deposit action
        System.out.println("Deposit $1000 into account BA1234:");
        // Deposit $1000 into account BA1234
       BA1234.deposit(1000);
        // Print the new balance after deposit
         System.out.println("New balance after depositing $1000: $"+BA1234.getBalance());

        // Print message to indicate withdrawal action
        System.out.println("Withdraw $600 from account BA1234:");
        // Withdraw $600 from account BA1234
      BA1234.withdraw(600);
        // Print the new balance after withdrawal
        System.out.println("New balance after withdrawing $600: $" +
BA1234.getBalance());

        // Print message to indicate creation of another SavingsAccount object
        System.out.println("Create a SavingsAccount object (A/c No. SA1000) with initial
balance of $300:");
        // Create a SavingsAccount object (A/c No. "SA1000") with initial balance of $300
        SavingsAccount SA1000 = new SavingsAccount("SA1000", 300);

        // Print message to indicate withdrawal action
        System.out.println("Try to withdraw $250 from SA1000!");
        // Withdraw $250 from SA1000 (balance falls below $100)
        SA1000.withdraw(250);
        // Print the balance after attempting to withdraw $250
        System.out.println("Balance after trying to withdraw $250: $" +
SA1000.getBalance()); } }
```

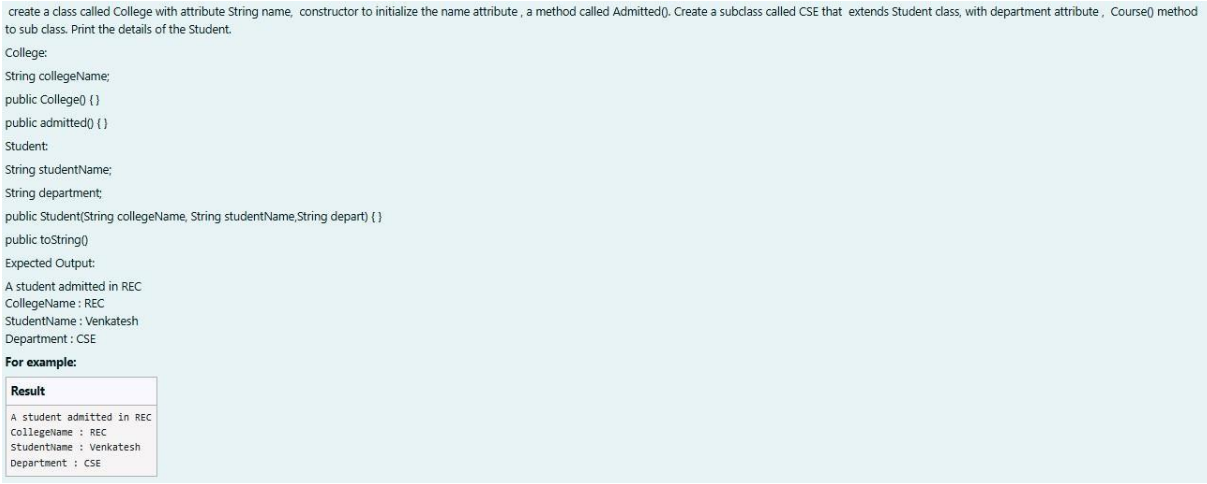
OUTPUT :



	Expected	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	✓

Passed all tests! ✓

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()

Expected Output:

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

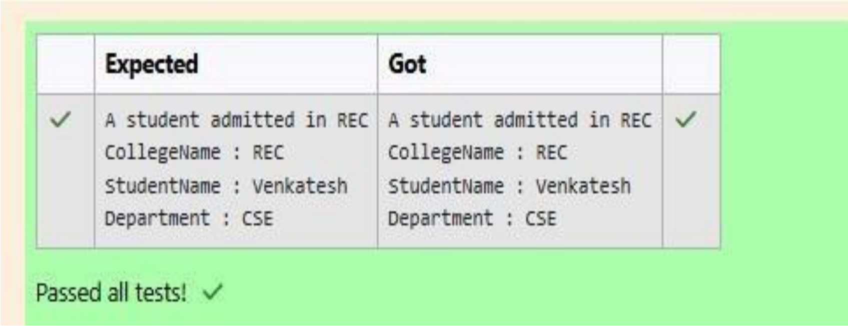
For example:

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

SOLUTION :

```
class College
{
public String collegeName;

public  College(String  collegeName)
    { // initialize the instance variables
    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 department) {
   // initialize the instance variables
   super(collegeName);
   this.studentName=studentName;
   this.department=department;
}

public String toString(){
   // return the details of the student return "CollegeName :
"+collegeName+"\n"+"StudentName :
"+studentName+"\n"+"Department : "+department;
} }
public class Main { public static void main
(String[] args) {
      Student s1 = new Student("REC","Venkatesh","CSE");
       s1.admitted();                       // invoke the admitted() method
      System.out.println(s1.toString());
}
}
```

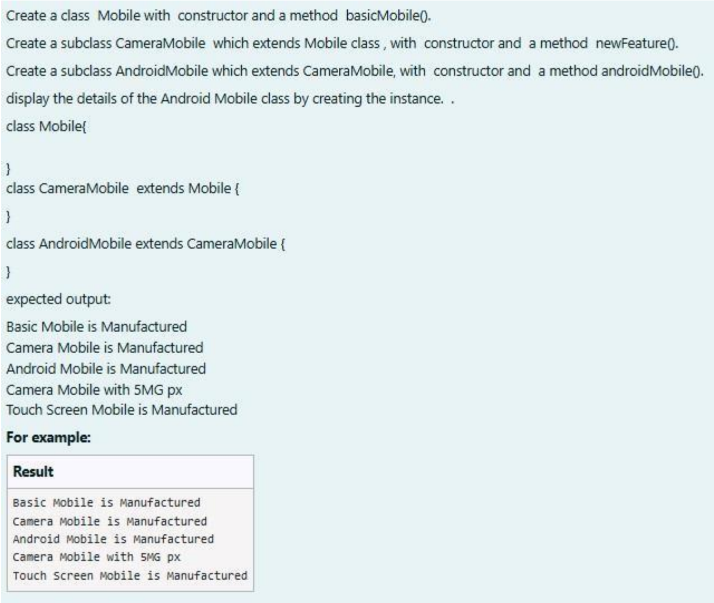
OUTPUT :



	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! ✓

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 {
}
```

expected output:

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

For example:

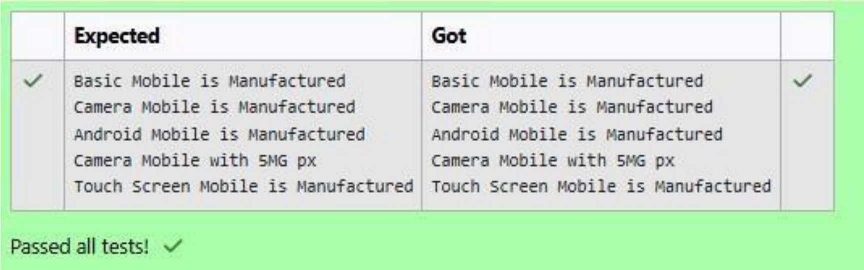
Result
Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured

SOLUTION :

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

```
    }
    void basmob(){
        System.out.println("Basic Mobile is Manufactured");
    }
}
class cam extends mob{ cam(){ super();
        System.out.println("Camera Mobile is Manufactured");
    }
    void newm(){
        System.out.println("Camera Mobile with 5MG px");

    }
}
class and extends cam{ and(){ super();
    System.out.println("Android Mobile is Manufactured");
    }
    void andmob(){
        System.out.println("Touch Screen Mobile is Manufactured");
    }
    } public class Main{ public static void main(String[]args){
and andmob=new and(); andmob.newm(); andmob.andmob();
    }

}
```

OUTPUT :

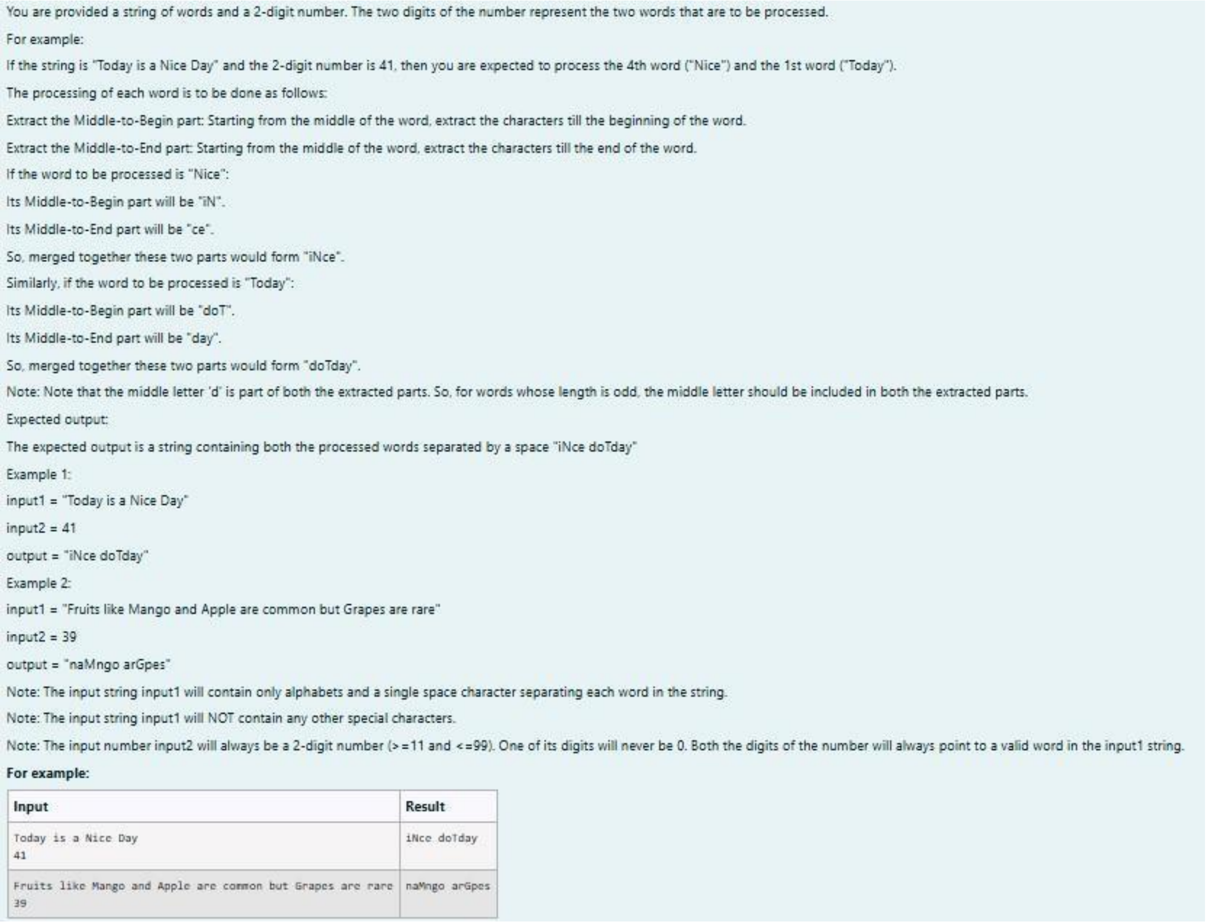


	Expected	Got	
✓	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile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

Passed all tests! ✓

Lab-06-String, StringBuffer

1.



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"

Example 1:

input1 = "Today is a Nice Day"

input2 = 41

output = "iNce doTday"

Example 2:

input1 = "Fruits like Mango and Apple are common but Grapes are rare"

input2 = 39

output = "naMngo arGpes"

Note: The input string input1 will contain only alphabets and a single space character separating each word in the string.

Note: The input string input1 will NOT contain any other special characters.

Note: The input number input2 will always be a 2-digit number (>=11 and <=99). One of its digits will never be 0. Both the digits of the number will always point to a valid word in the input1 string.

For example:

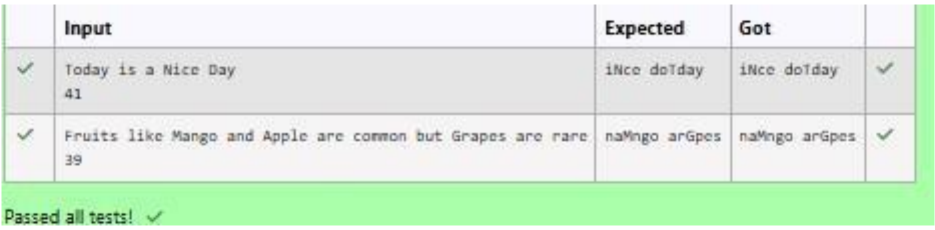
Input	Result
Today is a Nice Day 41	iNce doTday
Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes

SOLUTION :

```
import java.util.*; public class mix{
public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String g = scan.nextLine(); int n =
        scan.nextInt(),ones,flag = 0; StringBuffer
        temp = new StringBuffer(); StringBuffer
        temp1 = new StringBuffer(); int space =
        0; while (n > 0){ ones = (n %10) - 1;
              for(int i = 0; i < g.length();i++){
              if (g.charAt(i) == ' '){ space
                  = space + 1;
                  }
                  else if(space == ones && flag == 0){
                        temp.append(Character.toString(g.charAt(i)));
                  }
                  else if(space == ones && flag == 1){
                        temp1.append(Character.toString(g.charAt(i)));
                  }
```

```
            } space =
            0 ; flag =
            1; n = n
            /10;
        }
        rew m = new rew();
        System.out.println(m.r(temp1.toString()) + " " + m.r(temp.toString()));
    }
}
class rew{
    String r(String a){ int le
        = a.length(),n,q;
        StringBuffer temp3 = new StringBuffer(); if(le % 2 ==
        1){ n = ((int)(le/2)); q = ((int)(le/2));
        } else{ n =
        ((int)(le/2)) - 1; q = ((int)(le/2));
        } for(int i = n;i >= 0;i--){ temp3.append(Character.toString(a.charAt(i)));
            } for(int i = q;i < le;i++){ temp3.append(Character.toString(a.charAt(i)));
        }
        return temp3.toString(); }
}
```

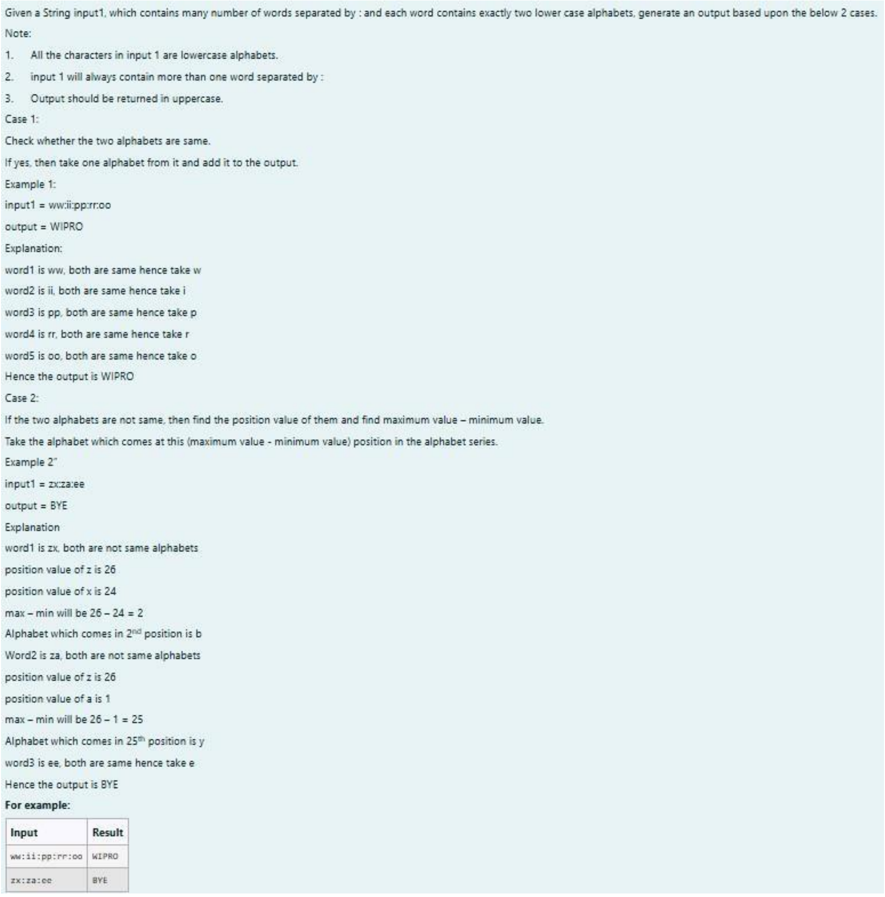
OUTPUT :



	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! ✓

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.

Case 1:

Check whether the two alphabets are same.

If yes, then take one alphabet from it and add it to the output.

Example 1:

input1 = ww:ii:pp:rr:oo

output = WIPRO

Explanation:

word1 is ww, both are same hence take w

word2 is ii, both are same hence take i

word3 is pp, both are same hence take p

word4 is rr, both are same hence take r

word5 is oo, both are same hence take o

Hence the output is WIPRO

Case 2:

If the two alphabets are not same, then find the position value of them and find maximum value – minimum value.

Take the alphabet which comes at this (maximum value - minimum value) position in the alphabet series.

Example 2"

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:

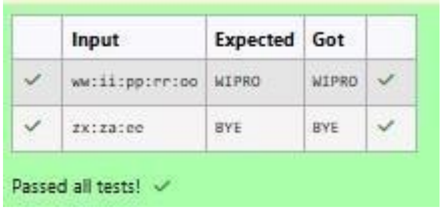
Input	Result
ww:ii:pp:rr:oo	WIPRO
zx:za:ee	BYE

SOLUTION :

```
import java.util.*; class diff{ char different(char a, char b){ if ((int)a !=
(int)b) return
(char)((int)'a' + ((int)a-(int)b) - 1); return a;
      }
}
public class Main{ public static void main(String[] args){ Scanner scan =
   new
   Scanner(System.in); diff z = new diff();
      String q = scan.nextLine();
      StringBuffer ans = new StringBuffer();
      StringBuffer temp = new
      StringBuffer(); for(int i = 0;i <
      q.length();i++){ if(q.charAt(i) == ':'){ temp.append(" ");
         } else{
         temp.append(Character.toString(q.charAt(i))); }
```

```
        }
        String h = temp.toString(); for(int i
        = 0;i < temp.length();i++){ if(i%3
        ==                                                          0){
            ans.append(Character.toString(z.different(h.charAt(i),h.charAt(i+1))));
          }
        }
        System.out.print(ans.toString().toUpperCase());
    }
}
```

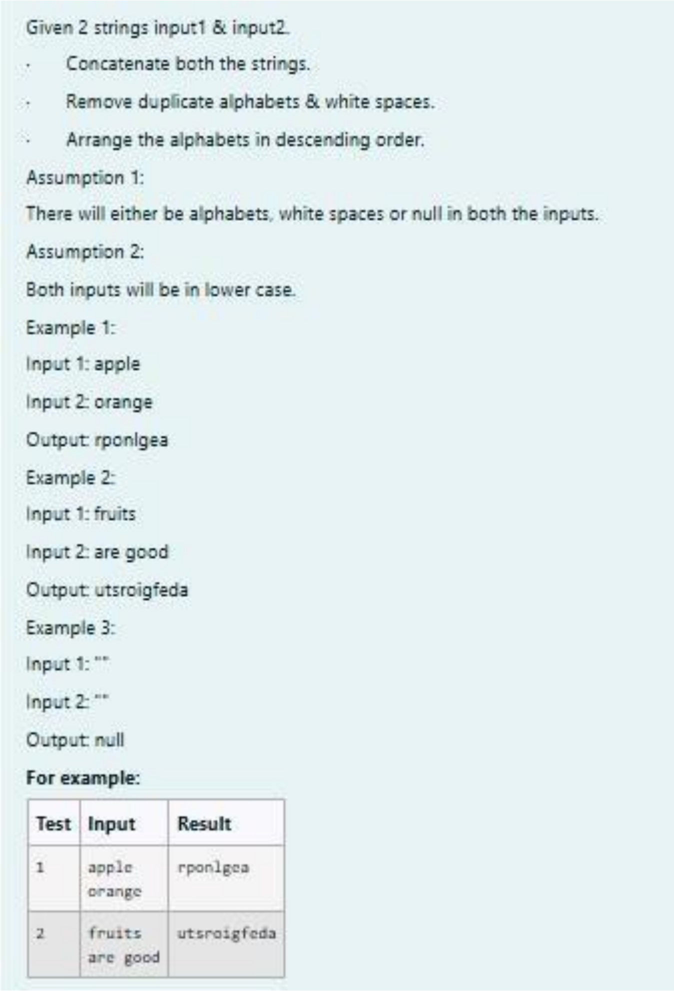
OUTPUT :



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

Passed all tests! ✓

3.



Given 2 strings input1 & input2.

- Concatenate both the strings.
- Remove duplicate alphabets & white spaces.
- Arrange the alphabets in descending order.

Assumption 1:

There will either be alphabets, white spaces or null in both the inputs.

Assumption 2:

Both inputs will be in lower case.

Example 1:

Input 1: apple

Input 2: orange

Output: rponlgea

Example 2:

Input 1: fruits

Input 2: are good

Output: utsroigfeda

Example 3:

Input 1: ""

Input 2: ""

Output: null

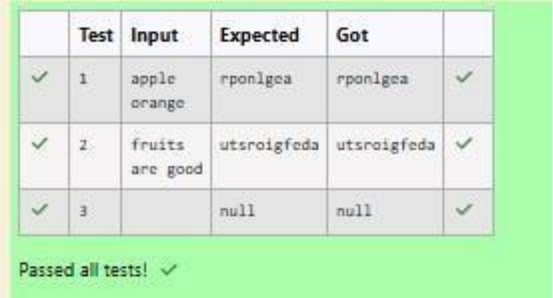
For example:

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

SOLUTION :

```
import java.util.*;


public class HelloWorld { public static void
    main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String a = scan.nextLine();
        String b = scan.nextLine();
        StringBuffer ab = new StringBuffer();
        if(a.trim().isEmpty() && b.trim().isEmpty()){
        System.out.print("null");
        } else{ for(int i = 0;i < a.length();i++){ if
        (a.charAt(i)
            != ' ') {
            ab.append(Character.toString(a.charAt(i))); }
        } for(int i = 0;i < b.length();i++){ if
        (b.charAt(i)
            != ' '){
            ab.append(Character.toString(b.charAt(i))); }
        } char[] d =
        ab.toString().toCharArray();
        Arrays.sort(d);
        for(int i = d.length - 1;i >= 1;i--){ if(d[i]
            != d[i-1])
            System.out.print(d[i]);
        }
        System.out.print(d[0]);
        }

    }
}
```

OUTPUT :

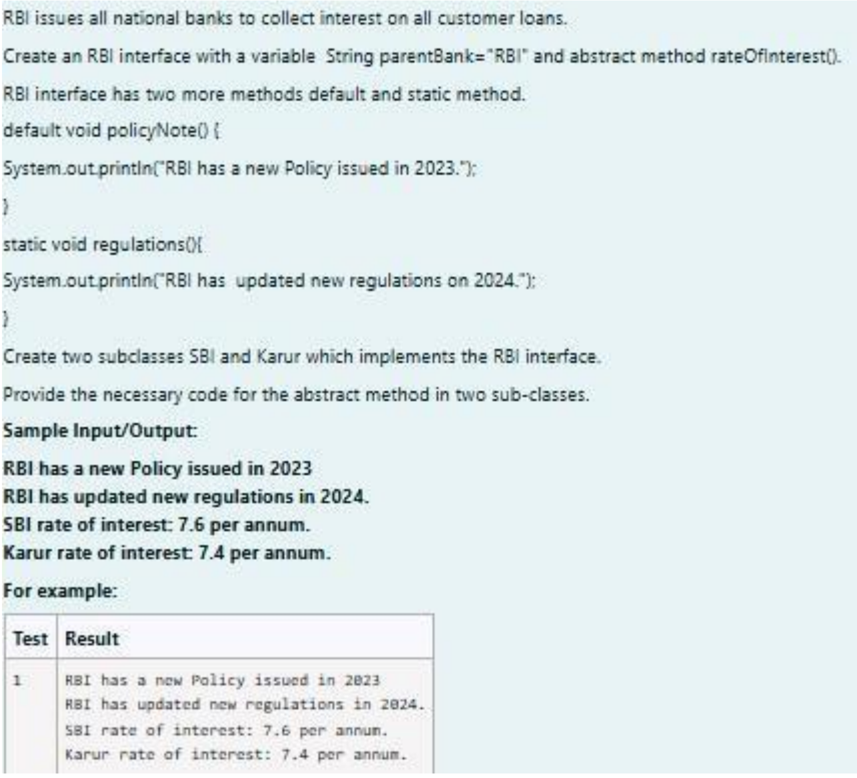


	Test	Input	Expected	Got	
✓	1	apple orange	rponlgea	rponlgea	✓
✓	2	fruits are good	utsroigfeda	utsroigfeda	✓
✓	3		null	null	✓

Passed all tests! ✓

Lab-07-Interfaces

1.



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.

Sample Input/Output:

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.

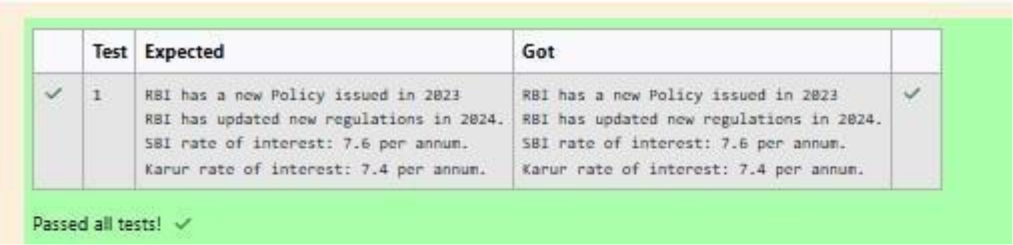
For example:

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.

SOLUTION :

```
// Define the RBI interface
interface RBI {
    // Variable declaration
    String parentBank = "RBI";

    // Abstract method
    double rateOfInterest();

    // Default method
    default void policyNote() {
        System.out.println("RBI has a new Policy issued in 2023"); }

    // Static method
    static void regulations() {
        System.out.println("RBI has updated new regulations in 2024.");
    }
}

// SBI class implementing RBI interface
class SBI implements RBI {
    // Implementing the abstract method
    public double rateOfInterest() {
        return 7.6;
    }
}

// Karur class implementing RBI interface
class Karur implements RBI { //
Implementing the abstract method public
double rateOfInterest() { return 7.4;
    }
}

// Main class to test the functionality
public class Main { public static void
main(String[] args) {
        // RBI policies and regulations
        RBI rbi = new SBI(); // Can be any class implementing RBI
        rbi.policyNote(); // Default method RBI.regulations(); //
        Static method

        // SBI bank details
        SBI sbi = new SBI();
        System.out.println("SBI rate of interest: " + sbi.rateOfInterest() + " per annum.");

        // Karur bank details
        Karur karur = new Karur();
        System.out.println("Karur rate of interest: " + karur.rateOfInterest() + " per annum.");
    }
}
```

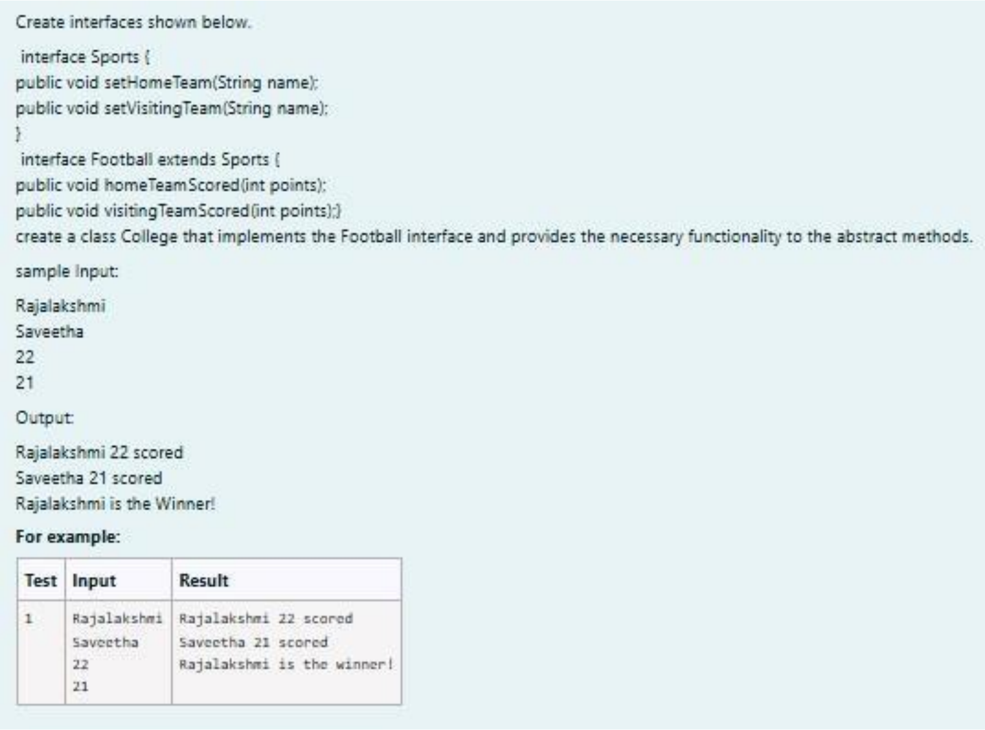
OUTPUT :



	Test	Expected	Got	
✓	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.	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.	✓

Passed all tests! ✓

2.



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.

sample Input:

```
Rajalakshmi
Saveetha
22
21
```

Output:

```
Rajalakshmi 22 scored
Saveetha 21 scored
Rajalakshmi is the Winner!
```

For example:

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

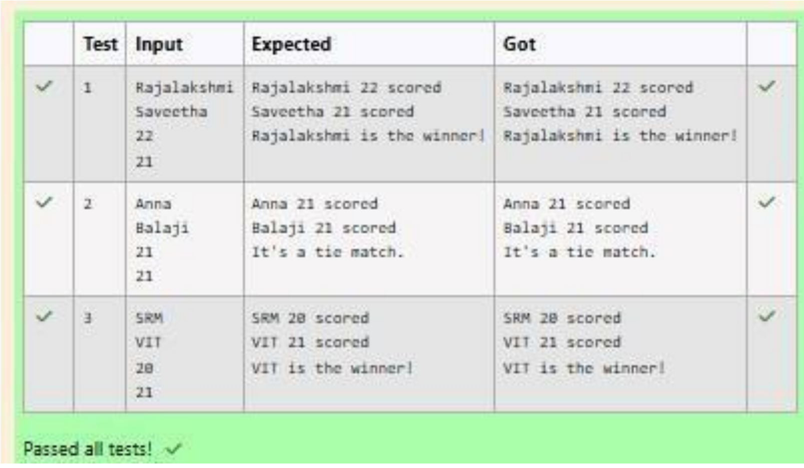
SOLUTION :

```
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);

        // Get home team name
        String hname = sc.nextLine();

        // Get visiting team name
        String vteam = sc.nextLine();

        // Create College object College
        match = new College();
        match.setHomeTeam(hname);
        match.setVisitingTeam(vteam);

        // Get points scored by home team
        int htpoints = sc.nextInt();
        match.homeTeamScored(htpoints);

        // Get points scored by visiting team
        int vtpoints = sc.nextInt();
        match.visitingTeamScored(vtpoints);

        // Determine and print the winning team
        match.winningTeam();

        sc.close();
```

```
}
}
```

```
}
```

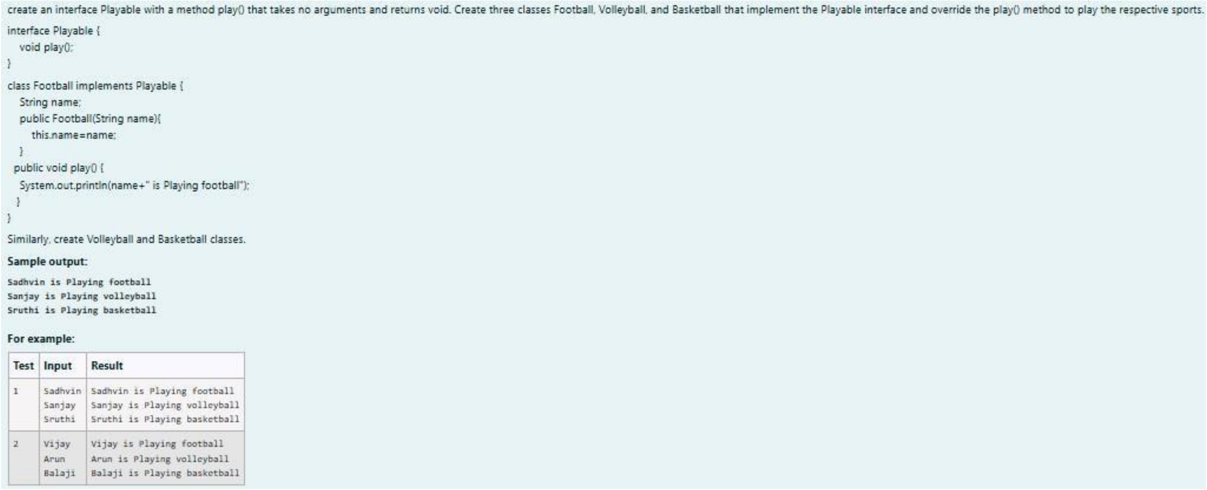
OUTPUT :



	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!	✓
✓	2	Anna Balaji 21 21	Anna 21 scored Balaji 21 scored It's a tie match.	Anna 21 scored Balaji 21 scored It's a tie match.	✓
✓	3	SRM VIT 20 21	SRM 20 scored VIT 21 scored VIT is the winner!	SRM 20 scored VIT 21 scored VIT is the winner!	✓

Passed all tests! ✓

3.



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.

Sample output:

```
Sadhvin is Playing football
Sanjay is Playing volleyball
Sruthi is Playing basketball
```

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

SOLUTION :

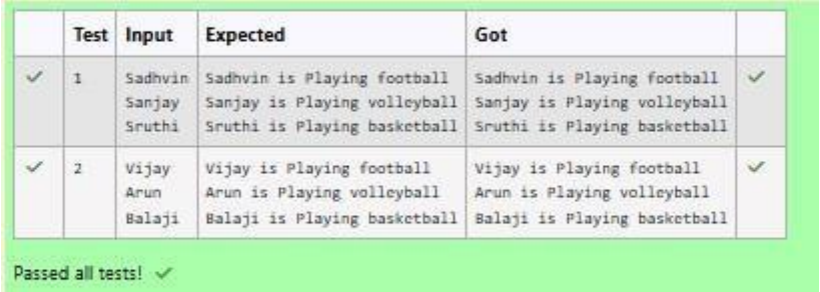
```
import java.util.Scanner;

// Define the Playable interface
interface Playable {
   // Abstract method to play the respective sport
   void play();
}

// Football class implementing Playable interface
class Football implements Playable { String
name;

   // Constructor
   public Football(String name) {
      this.name = name;
   }

   // Override the play method
```

```
    public void play() {
        System.out.println(name + " is Playing football");
    }
}

// Volleyball class implementing Playable interface
class Volleyball implements Playable { String
name;

    // Constructor
    public Volleyball(String name) {
        this.name = name;
    }

    // Override the play method
    public void play() {
        System.out.println(name + " is Playing volleyball");
    }
}

// Basketball class implementing Playable interface
class Basketball implements Playable { String
name;

    // Constructor
    public Basketball(String name) { this.name
        = name;
    }

    // Override the play method
    public void play() {
        System.out.println(name + " is Playing basketball");
    }
}

// Main class to test the functionality
public class Main { public static void
main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input for Football player

        String footballPlayerName = scanner.nextLine();
        Football footballPlayer = new Football(footballPlayerName);

        // Input for Volleyball player
```

```
String volleyballPlayerName = scanner.nextLine();
Volleyball volleyballPlayer = new Volleyball(volleyballPlayerName);
```

```
        // Input for Basketball player

        String basketballPlayerName = scanner.nextLine();
        Basketball basketballPlayer = new Basketball(basketballPlayerName);

        // Call the play method for each player
        footballPlayer.play();
        volleyballPlayer.play();
        basketballPlayer.play();

        scanner.close();
    }
}
```

OUTPUT :

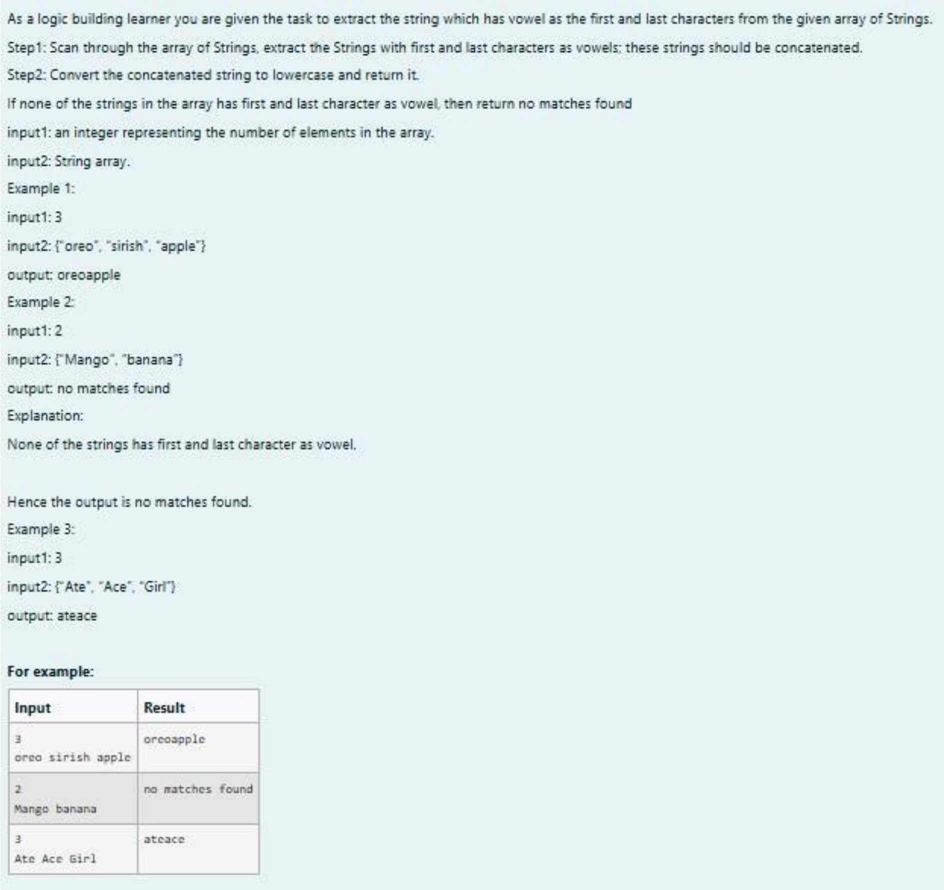


	Test	Input	Expected	Got	
✓	1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	✓
✓	2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji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

Passed all tests! ✓

Lab-08 - Polymorphism, Abstract Classes, final Keyword

1.



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

input1: an integer representing the number of elements in the array.

input2: String array.

Example 1:

input1: 3

input2: {"oreo", "sirish", "apple"}

output: oreoapple

Example 2:

input1: 2

input2: {"Mango", "banana"}

output: no matches found

Explanation:

None of the strings has first and last character as vowel.

Hence the output is no matches found.

Example 3:

input1: 3

input2: {"Ate", "Ace", "Girl"}

output: ateace

For example:

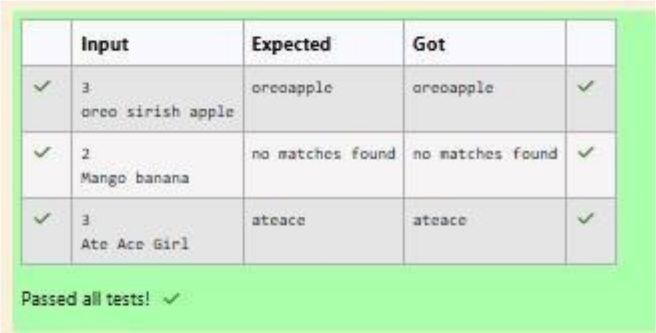
Input	Result
3 oreo sirish apple	oreoapple
2 Mango banana	no matches found
3 Ate Ace Girl	ateace

SOLUTION :

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

VowelStringExtractor {

   // Method to extract strings with vowels as first and last characters
   public static String extractVowelStrings(String[] stringArray) {
       StringBuilder result = new StringBuilder();
       String vowels = "aeiouAEIOU"; // String containing all vowels

       // Iterate through the array of strings for
       (String s : stringArray) {
           // Check if the string is not empty and if both the first and last characters are vowels
if (s.length() > 0 && vowels.indexOf(s.charAt(0)) != -1 &&
vowels.indexOf(s.charAt(s.length() - 1)) != -1) { result.append(s); // Append matching
string to the result }
       }

       // Return the concatenated string in lowercase or "no matches found"
       return result.length() > 0 ? result.toString().toLowerCase() : "no matches found"; }
```

```
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Input for the number of strings

    int n = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

    // Input for the strings in one line

    String input = scanner.nextLine();
    String[] strings = input.split(" "); // Split input into an array

    // Process and output the result
    String result = extractVowelStrings(strings); System.out.println(result);

    scanner.close(); // Close the scanner }
}
```

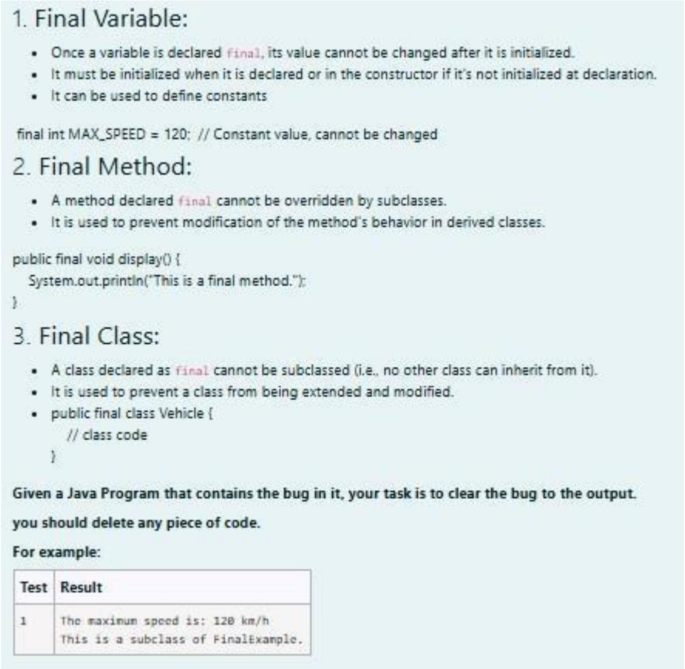
OUTPUT :



	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! ✓

2.



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
  }

Given a Java Program that contains the bug in it, your task is to clear the bug to the output.

you should delete any piece of code.

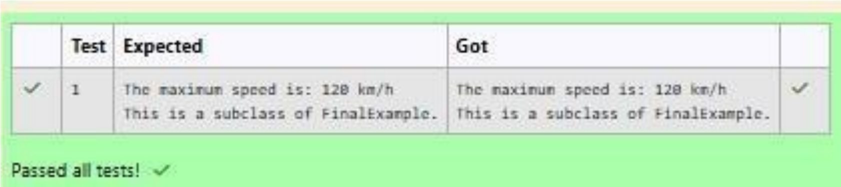
For example:

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

SOLUTION :

```
// Final class definition
final class FinalExample {
    // Final variable
    final int MAX_SPEED = 120; // Constant value

    // Final method public final
    void display() {
        System.out.println("The maximum speed is: " + MAX_SPEED + " km/h");
    }
}

// Main class to test the final class public
class Test { public static void
main(String[] args) {
        // Create an instance of FinalExample
        FinalExample example = new FinalExample();
        example.display();

        // Uncommenting the following line will result in a compile-time error //
        because FinalExample is a final class and cannot be subclassed. // class
        SubclassExample extends FinalExample { }

        System.out.println("This is a subclass of FinalExample.");
    }
}
```

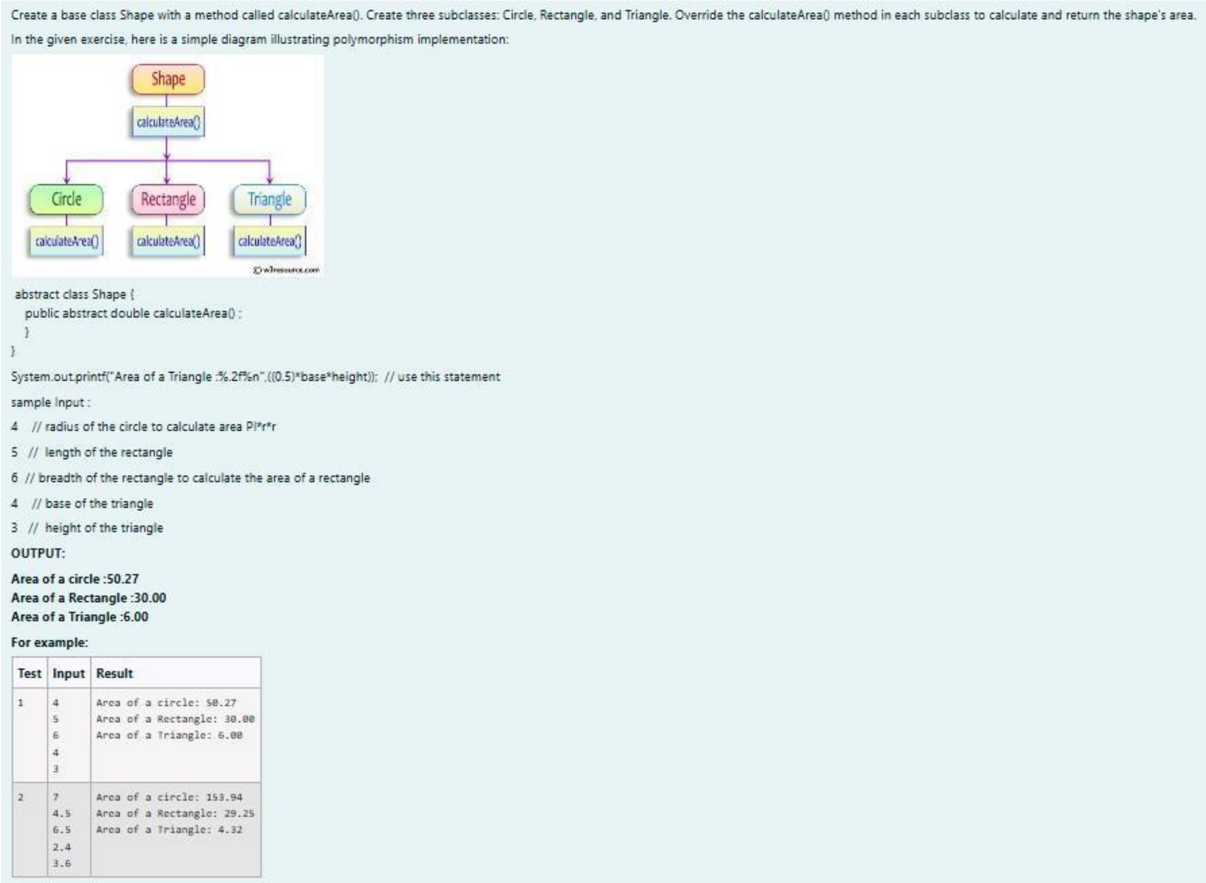
OUTPUT :



	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! ✓

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

sample Input :

4 // radius of the circle to calculate area PI*r*r

5 // length of the rectangle

6 // breadth of the rectangle to calculate the area of a rectangle

4 // base of the triangle

3 // height of the triangle

OUTPUT:

Area of a circle :50.27
Area of a Rectangle :30.00
Area of a Triangle :6.00

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

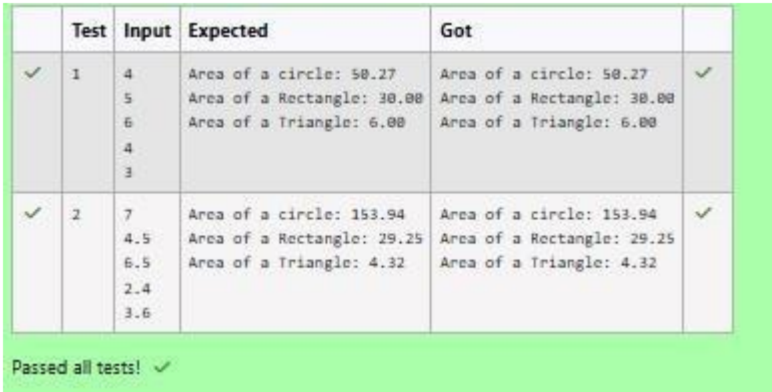
SOLUTION :

```
import java.util.Scanner;

// Abstract class Shape abstract
class Shape { public abstract double
calculateArea(); }

// Circle class
class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
```

```
   public double calculateArea() { return Math.PI * radius * radius; // Area
   of circle: πr² }
}

// Rectangle class
class Rectangle extends Shape { private double
   length; private double breadth;

   public Rectangle(double length, double breadth) { this.length = length;
      this.breadth = breadth;
   }

   @Override
   public double calculateArea() { return length * breadth; // Area of rectangle:
      length * breadth
   }
}

// Triangle class
class Triangle extends Shape { private double
   base; private double height;

   public Triangle(double base, double height) { this.base = base;
      this.height = height;
   }

   @Override
   public double calculateArea() { return 0.5 * base * height; // Area of triangle: 0.5 *
      base * height
   }
}

// Main class to test the shapes public class ShapeTest
{ public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Input for Circle

      double radius = scanner.nextDouble();
      Circle circle = new Circle(radius);
      System.out.printf("Area of a circle: %.2f%n", circle.calculateArea());

      // Input for Rectangle
```

```
        double length = scanner.nextDouble();

        double breadth = scanner.nextDouble();
        Rectangle rectangle = new Rectangle(length, breadth);
        System.out.printf("Area of a Rectangle: %.2f%n", rectangle.calculateArea());

        // Input for Triangle double base =

        scanner.nextDouble();

        double height = scanner.nextDouble();
        Triangle triangle = new Triangle(base, height);
        System.out.printf("Area of a Triangle: %.2f%n", triangle.calculateArea());

        scanner.close();
    }
}
```

OUTPUT :

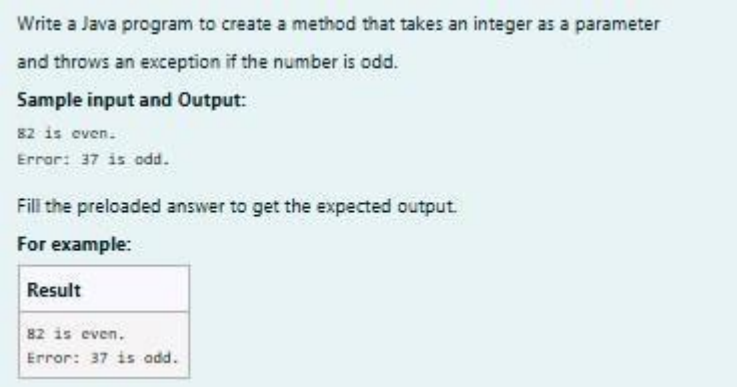


	Test	Input	Expected	Got	
✓	1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	✓
✓	2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	Area of a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

Passed all tests! ✓

Lab-09-Exception Handling

1.



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

Sample input and Output:

```
82 is even.
Error: 37 is odd.
```

Fill the preloaded answer to get the expected output.

For example:

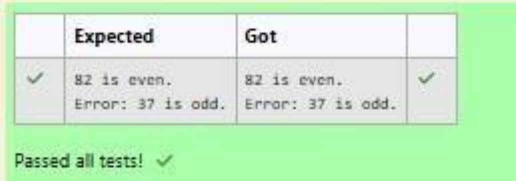
Result
82 is even. Error: 37 is odd.

SOLUTION :

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

```
        int n = 82; trynumber(n); n = 37;
        trynumber(n); // Call the
        trynumber(n);
    }

    public static void trynumber(int n) { try {
        checkEvenNumber(n); // Call the checkEvenNumber()
        System.out.println(n + " is even.");
        } catch (Exception e) { // Catch the exception
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void checkEvenNumber(int number) { if (number % 2 != 0) { throw new
        RuntimeException(number + " is odd."); // Throw a RuntimeException }
    }
}
```

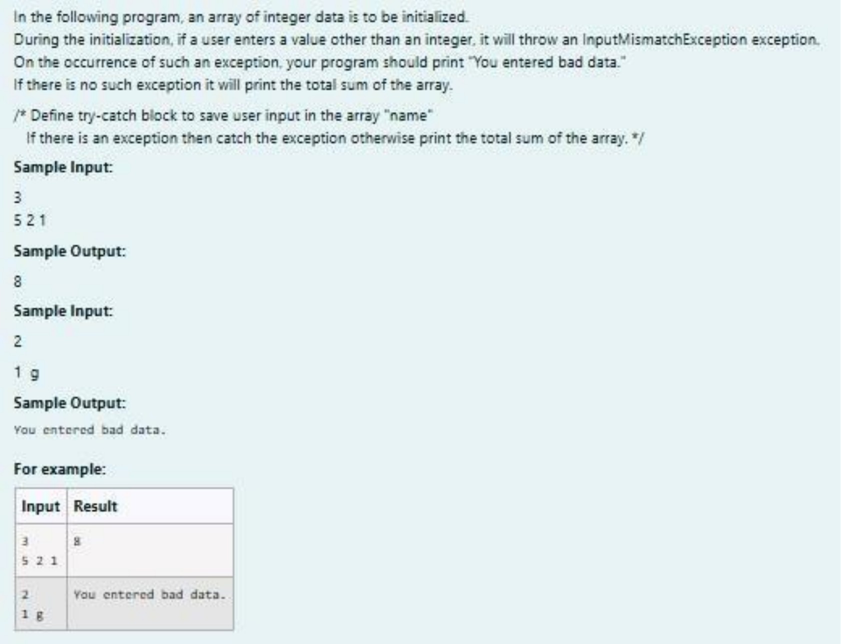
OUTPUT :



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

Passed all tests! ✓

2.



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. */

Sample Input:

3
5 2 1

Sample Output:

8

Sample Input:

2
1 g

Sample Output:

```
You entered bad data.
```

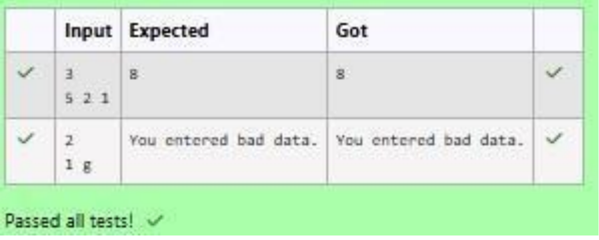
For example:

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

SOLUTION :

```
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 sum = 0; // save
      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. */
      try { for (int i = 0; i < length; i++) { name[i]
         = sc.nextInt(); // save user input in the
         array }

         // Calculate the total sum
         for (int num : name) {
         sum += num;
         }

         // Print the total sum
         System.out.println(sum);
      } catch (InputMismatchException e) {
         System.out.println("You entered bad data.");
      }

      sc.close(); // Close the scanner
   }
}
```

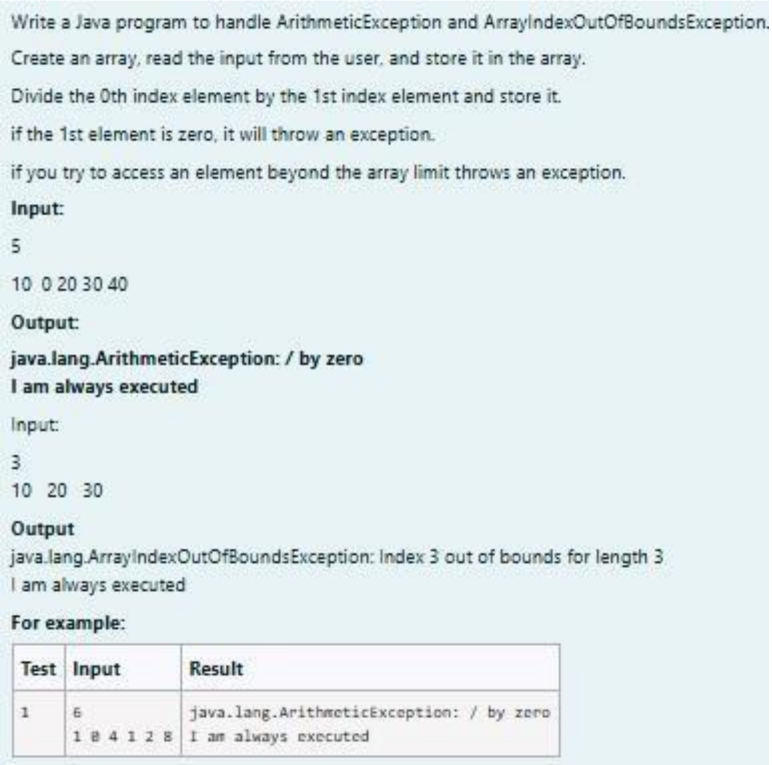
OUTPUT :



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

Passed all tests! ✓

3.



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.

Input:

```
5
10 0 20 30 40
```

Output:

```
java.lang.ArithmeticException: / by zero
I am always executed
```

Input:

```
3
10  20  30
```

Output

```
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
I am always executed
```

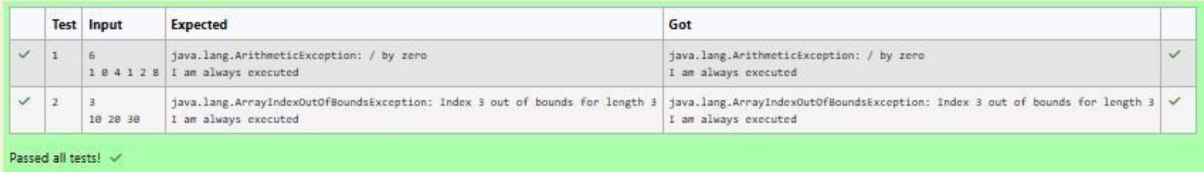
For example:

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

SOLUTION :

```
import java.util.Scanner;

public class ExceptionHandlingExample { public
    static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read the size of the array
        int size = scanner.nextInt();

        // Initialize the array int[]
        numbers = new int[size];

        // Read the elements into the array
        for (int i = 0; i < size; i++) {
        numbers[i] = scanner.nextInt();
        }

        try {
            // Attempt to perform division int result = numbers[0] / numbers[1]; // This may
            cause an ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println(e); // Catch division by zero
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(e); // Catch accessing out of bounds
        } catch (Exception e) {
            System.out.println(e); // Catch any other exceptions
        } finally {
            // This block is always executed
        }

        try {
            // Attempt to access an out-of-bounds index int outOfBoundsValue =
            numbers[3]; // This will trigger
ArrayIndexOutOfBoundsException if size < 4
        } catch (ArrayIndexOutOfBoundsException e) { System.out.println(e);
        } finally {
            // This block is always executed for the second try System.out.println("I am
            always executed");
        }

        scanner.close();
    }
}
```

OUTPUT :

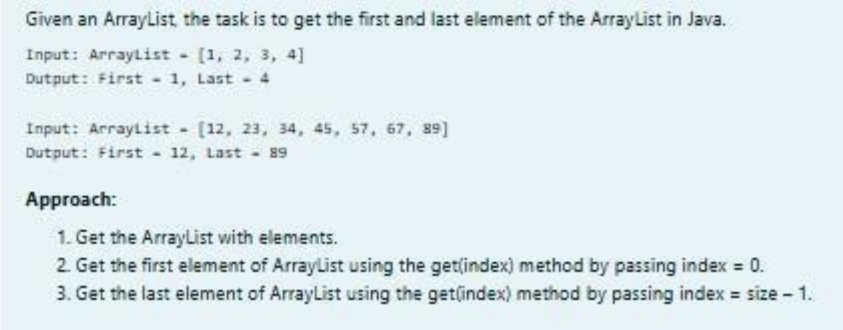


	Test	Input	Expected	Got	
✓	1	6 1 0 4 1 2 8	java.lang.ArithmeticException: / by zero I am always executed	java.lang.ArithmeticException: / by zero I am always executed	✓
✓	2	3 10 20 30	java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3 I am always executed	java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3 I am always executed	✓

Passed all tests! ✓

Lab-10- Collection- List

1.



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

```
Input: ArrayList = [1, 2, 3, 4]
Output: First = 1, Last = 4

Input: ArrayList = [12, 23, 34, 45, 57, 67, 89]
Output: First = 12, Last = 89
```

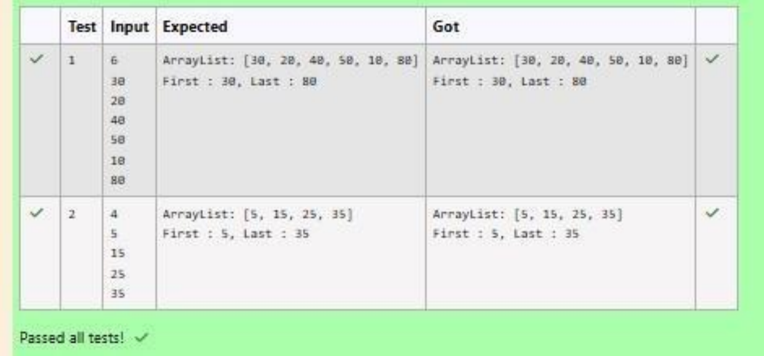
Approach:

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.

SOLUTION :

```
import java.util.ArrayList;
import java.util.Scanner;

public class FirstAndLastElement { public
    static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Create an ArrayList
        ArrayList<Integer> numbers = new ArrayList<>();
        int numElements = scanner.nextInt();

        for (int i = 0; i < numElements; i++) { int
            number = scanner.nextInt();
            numbers.add(number);
        }
        System.out.println("ArrayList: " + numbers);

        // Get the first element int firstElement
        = numbers.get(0);

        //   Get   the   last   element   int   lastElement   =
        numbers.get(numbers.size() - 1);

        // Print the results
        System.out.print("First : " + firstElement);
        System.out.println(", Last : " + lastElement);
    }
}
```

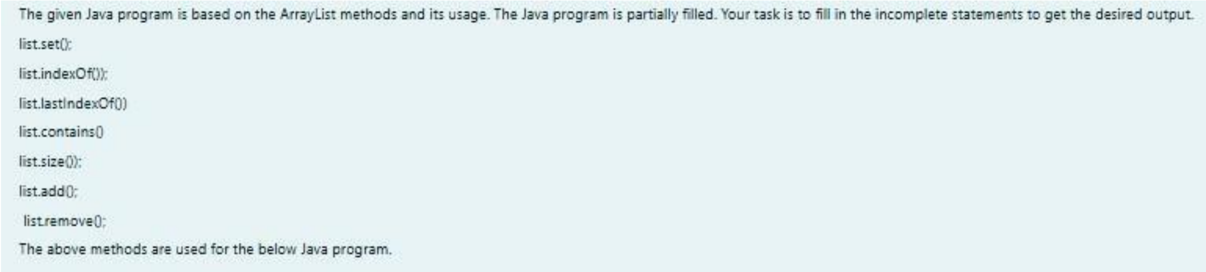
OUTPUT :



	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! ✓

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();

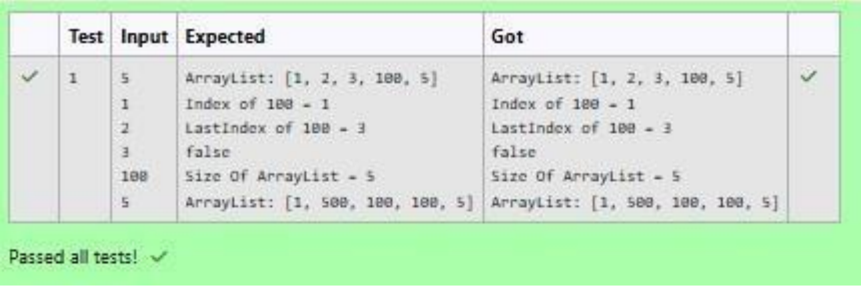
list.remove();

The above methods are used for the below Java program.

SOLUTION :

```
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());

 // printing initial value ArrayList
 System.out.println("ArrayList: " + list);

//Replacing the element at index 1 with 100 list.set(1,100);

 //Getting the index of first occurrence of 100
System.out.println("Index of 100 = "+ list.indexOf(100)              );

//Getting the index of last occurrence of 100
 System.out.println("LastIndex of 100 = "+ list.lastIndexOf(100));
// Check whether 200 is in the list or not
System.out.println(list.contains(200)); //Output : false
 // Print ArrayList size
 System.out.println("Size Of ArrayList = "+list.size() );
//Inserting 500 at index 1
list.add(1,500);                              // code here
 //Removing an element from position 3
list.remove(3);                               // code here
  System.out.print("ArrayList: " + list);
 }
}
```

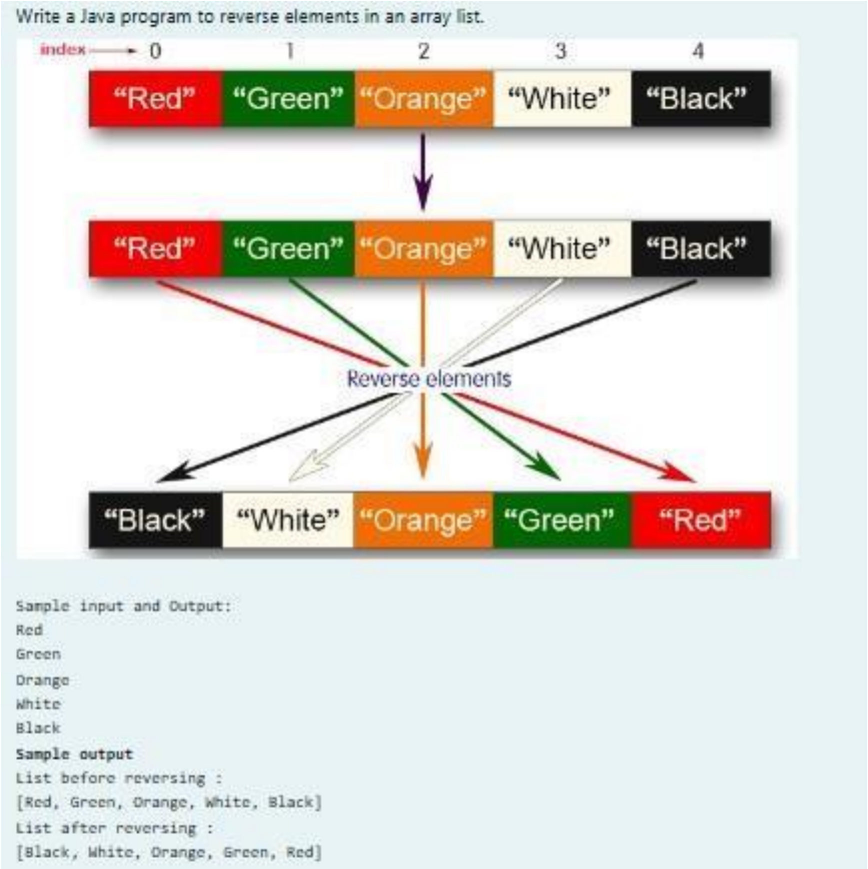
OUTPUT :



	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! ✓

3.



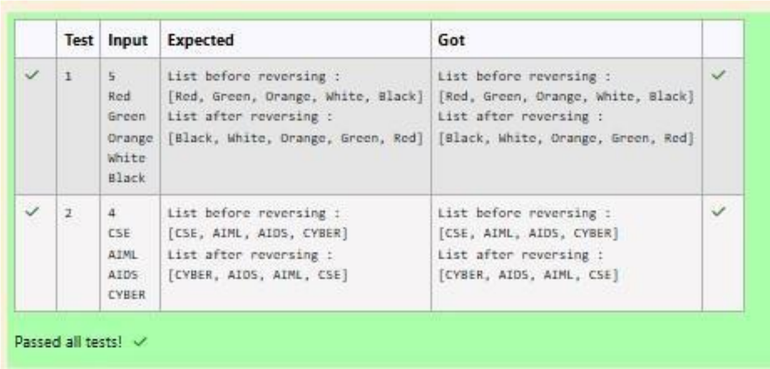
Write a Java program to reverse elements in an array list.

```
Sample input and Output:
Red
Green
Orange
White
Black
Sample output
List before reversing :
[Red, Green, Orange, White, Black]
List after reversing :
[Black, White, Orange, Green, Red]
```

SOLUTION :

```
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> list = new ArrayList<>();
      int n = scanner.nextInt();

      for (int i = 0; i < n; i++) {
         String element = scanner.next();
         list.add(element);
      }

      System.out.println("List before reversing : ");
      System.out.println(list);

      Collections.reverse(list);

      System.out.println("List after reversing : ");
      System.out.println(list);
}
}
```

OUTPUT :

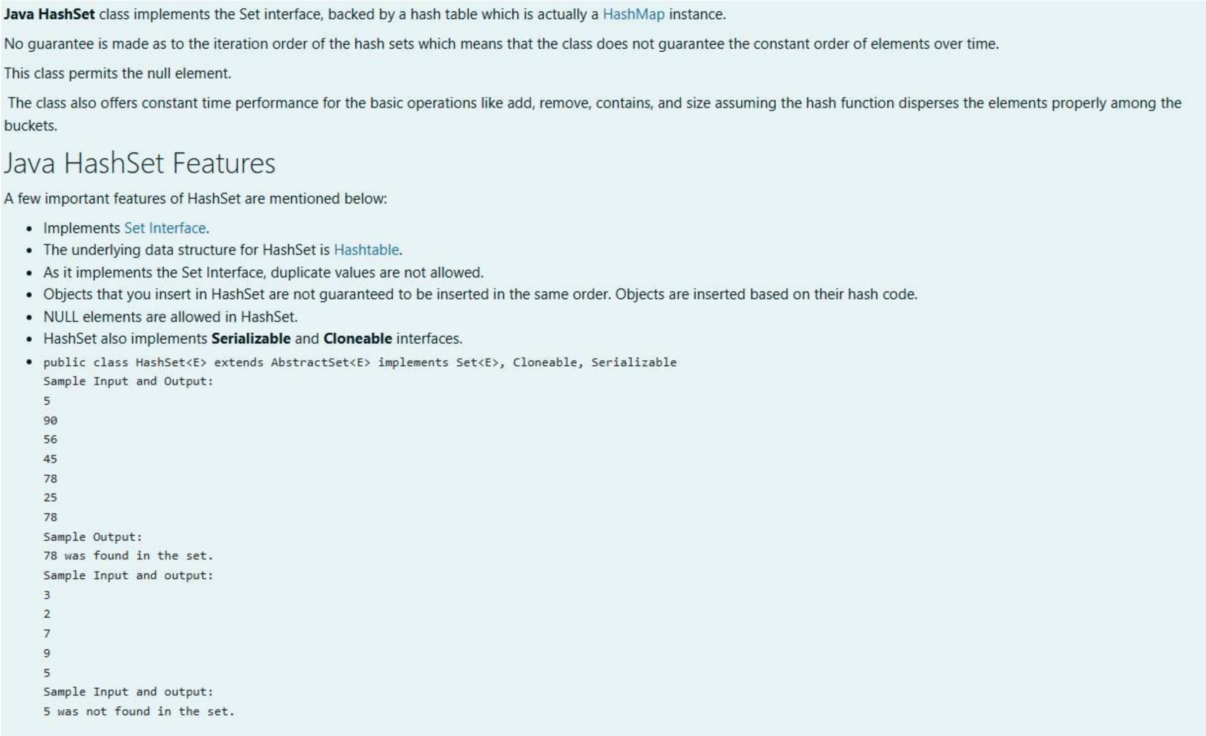


	Test	Input	Expected	Got	
✓	1	5 Red Green Orange White Black	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	✓
✓	2	4 CSE AIML AIDS CYBER	List before reversing : [CSE, AIML, AIDS, CYBER] List after reversing : [CYBER, AIDS, AIML, CSE]	List before reversing : [CSE, AIML, AIDS, CYBER] List after reversing : [CYBER, AIDS, AIML, CSE]	✓

Passed all tests! ✓

Lab-11-Set, Map

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:

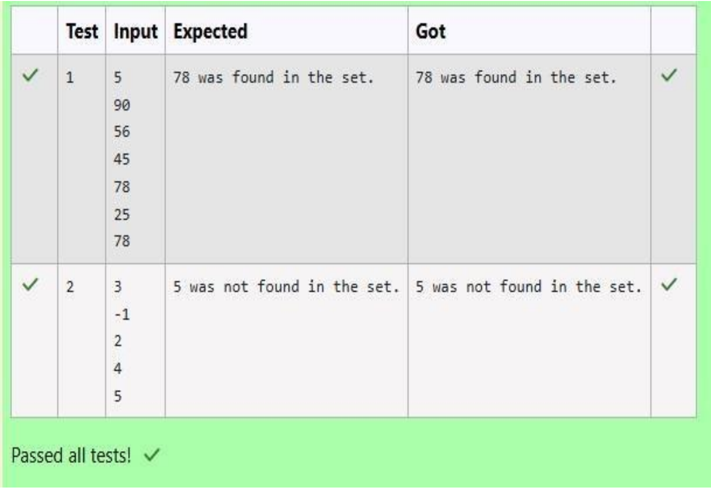
- Implements Set Interface.
- The underlying data structure for HashSet is Hashtable.
- As it implements the Set Interface, duplicate values are not allowed.
- Objects that you insert in HashSet are not guaranteed to be inserted in the same order. Objects are inserted based on their hash code.
- NULL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

```
Sample Input and Output:
5
90
56
45
78
25
78
Sample Output:
78 was found in the set.
Sample Input and output:
3
2
7
9
5
Sample Input and output:
5 was not found in the set.
```

SOLUTION :

```
import java.util.HashSet;
import java.util.Scanner;

public class Prog { public static void
    main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Read the number of elements
        int n = sc.nextInt();
        // Create a HashSet object to store numbers
        HashSet<Integer> numbers = new HashSet<>();

        // Add numbers to the HashSet for
        (int i = 0; i < n; i++) {
        numbers.add(sc.nextInt());
        }

        // Read the search key
        int skey = sc.nextInt();

        // Check if skey is present in the HashSet
        if (numbers.contains(skey)) {
          System.out.println(skey + " was found in the set.");
        } else {
          System.out.println(skey + " was not found in the set.");
        }

        // Close the scanner
        sc.close();
    }
}
```

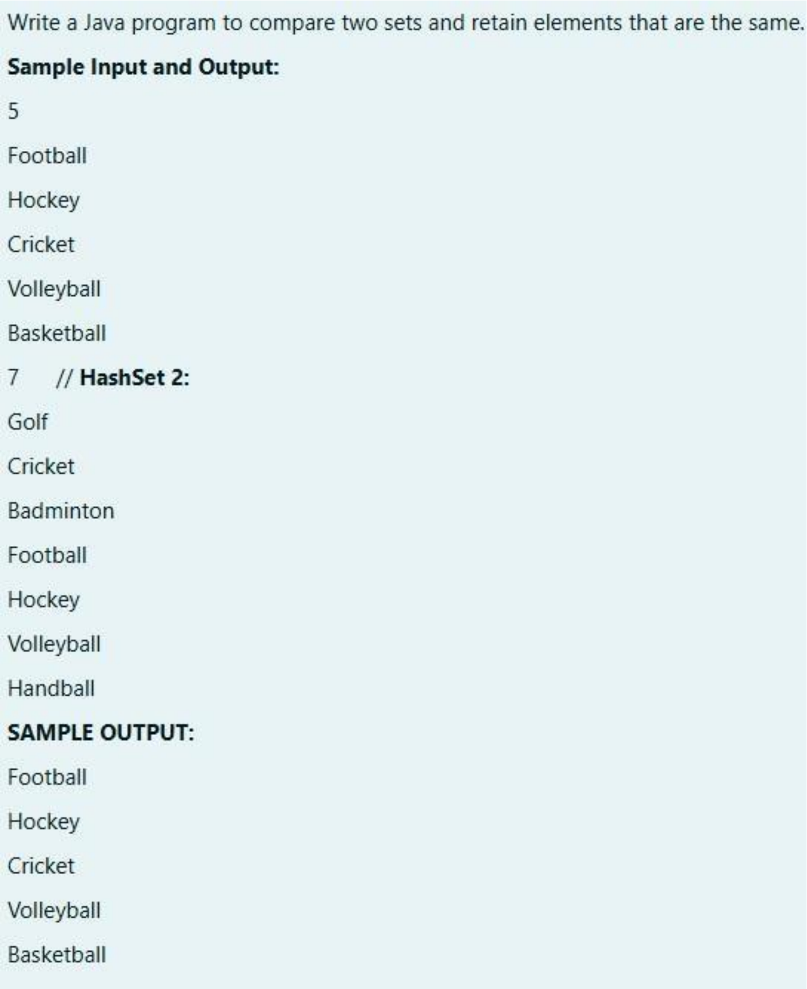
OUTPUT :



	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! ✓

2.



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

Sample Input and Output:

5

Football

Hockey

Cricket

Volleyball

Basketball

7 // **HashSet 2:**

Golf

Cricket

Badminton

Football

Hockey

Volleyball

Handball

SAMPLE OUTPUT:

Football

Hockey

Cricket

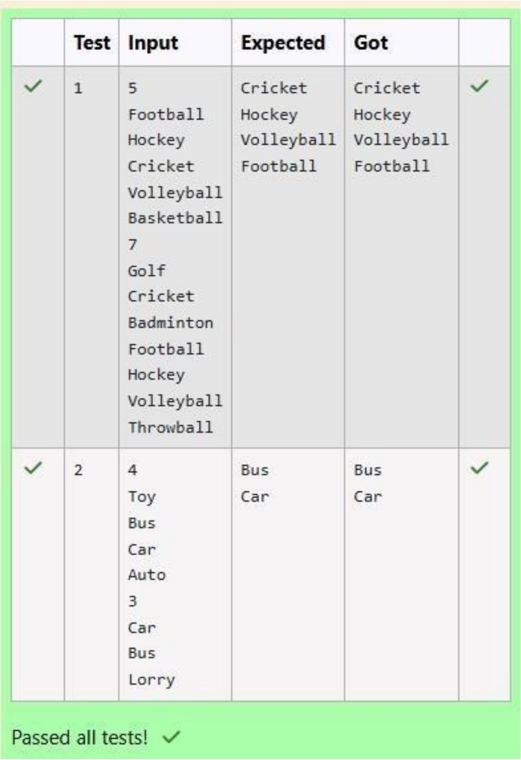
Volleyball

Basketball

SOLUTION :

```
import java.util.HashSet; import
java.util.Scanner; import
java.util.Set;

public class CompareSets { public static void
    main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read the size of the first set
        int size1 = Integer.parseInt(scanner.nextLine());

        // Create a HashSet to store the first set of elements
        Set<String> set1 = new HashSet<>();

        // Read elements for the first set for (int
        i = 0; i < size1; i++) {
        set1.add(scanner.nextLine());
        }

        // Read the size of the second set
        int size2 = Integer.parseInt(scanner.nextLine());

        // Create a HashSet to store the second set of elements
        Set<String> set2 = new HashSet<>();

        // Read elements for the second set for
        (int i = 0; i < size2; i++) {
        set2.add(scanner.nextLine());
        }

        // Retain common elements using the retainAll() method
        set1.retainAll(set2);

        // Print the common elements
        for (String element : set1) {
            System.out.println(element);
        }

        scanner.close();
    }
}
```

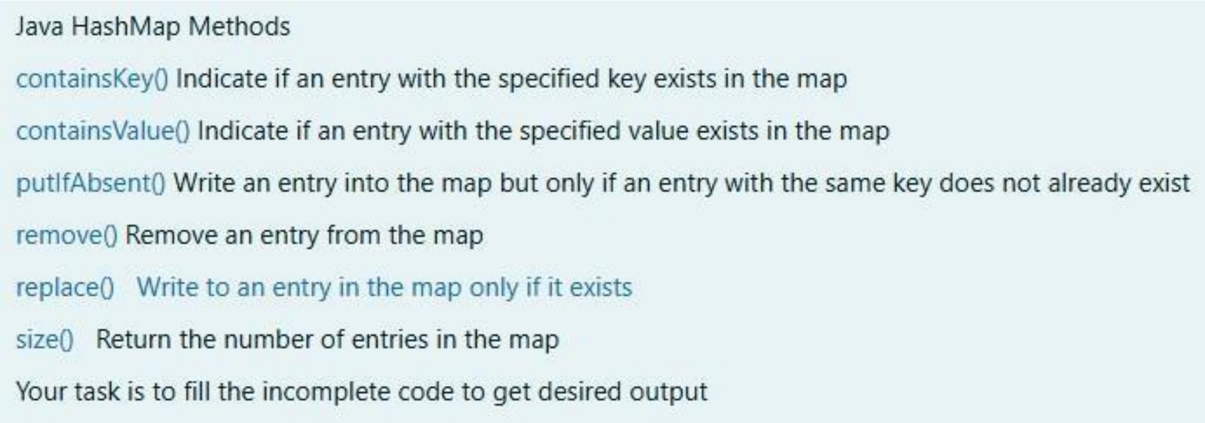
OUTPUT :



	Test	Input	Expected	Got	
✓	1	5 Football Hockey Cricket Volleyball Basketball 7 Golf Cricket Badminton Football Hockey Volleyball Throwball	Cricket Hockey Volleyball Football	Cricket Hockey Volleyball Football	✓
✓	2	4 Toy Bus Car Auto 3 Car Bus Lorry	Bus Car	Bus Car	✓

Passed all tests! ✓

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

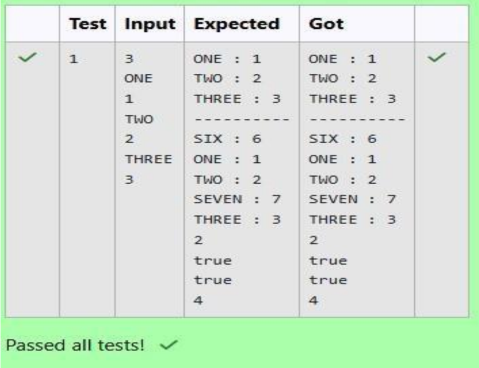
SOLUTION :

```
import java.util.HashMap;
import
java.util.Map.Entry;
import java.util.Scanner;
import java.util.Set; public
class Prog {

  public static void main(String[] args) {
    // Creating HashMap with default initial capacity and load factor
    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);
    }

    // Printing key-value pairs
    Set<Entry<String, Integer>> entrySet = map.entrySet();

    for (Entry<String, Integer> entry : entrySet) {
      System.out.println(entry.getKey() + " : " + entry.getValue());
    }
    System.out.println("----------");

    // Creating another HashMap
    HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();
```

```
// Inserting key-value pairs to anotherMap using put() method anotherMap.put("SIX",
6);
```

```
        anotherMap.put("SEVEN", 7);

        // Inserting key-value pairs of map to anotherMap using putAll() method
        anotherMap.putAll(map); // This line fills in the missing code

        // Printing key-value pairs of anotherMap entrySet
        = anotherMap.entrySet();

        for (Entry<String, Integer> entry : entrySet) {
           System.out.println(entry.getKey() + " : " + entry.getValue());
        }

        // Adds key-value pair 'FIVE-5' only if it is not present in map
        map.putIfAbsent("FIVE", 5);

        // Retrieving a value associated with key 'TWO' int
        value = map.get("TWO");
        System.out.println(value); // Prints the value associated with key "TWO" (if it
exists)

        // Checking whether key 'ONE' exists in map
        System.out.println(map.containsKey("ONE")); // Prints true if "ONE" is a key,
false otherwise

        // Checking whether value '3' exists in map
        boolean valueExists = map.containsValue(3); // You can use a variable to store
the result
        System.out.println(valueExists); // Prints true if value 3 exists in the map, false
otherwise

        // Retrieving the number of key-value pairs present in map
        System.out.println(map.size()); // Prints the number of entries in the map
    }
}
```

OUTPUT :

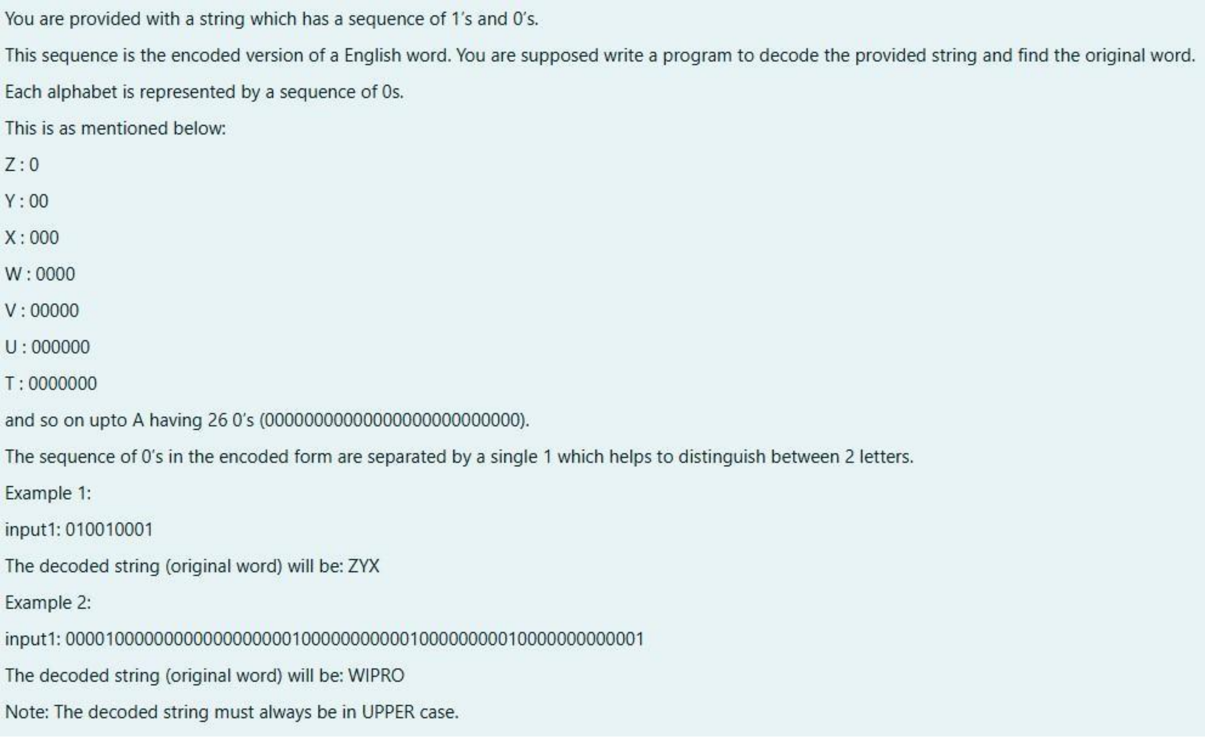


	Test	Input	Expected	Got	
✓	1	3 ONE 1 TWO 2 THREE 3	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	✓

Passed all tests! ✓

Lab-12-Introduction to I/O, I/O Operations, Object Serialization

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.

Example 1:

input1: 010010001

The decoded string (original word) will be: ZYX

Example 2:

input1: 0000100000000000000000010000000000010000000001000000000000001

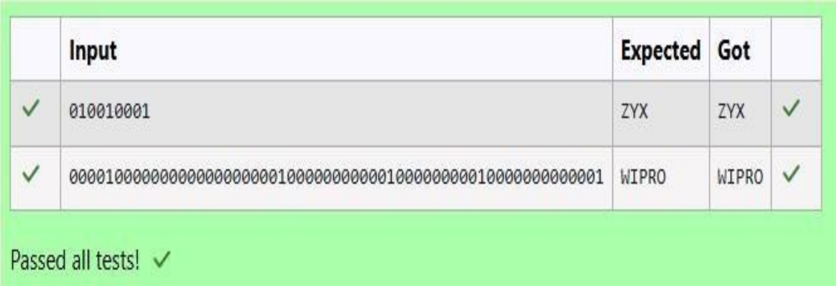
The decoded string (original word) will be: WIPRO

Note: The decoded string must always be in UPPER case.

SOLUTION :

```
import java.util.Scanner;

public class DecodeString { public static
   void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      String encodedString = scanner.nextLine();

      StringBuilder decodedString = new StringBuilder();
      int count = 0;

      for (int i = 0; i < encodedString.length(); i++) {
         if (encodedString.charAt(i) == '0') {
         count++;
         } else { char decodedChar = (char) ('Z' - count +
            1); decodedString.append(decodedChar);
            count = 0;
         }
      }

      System.out.println(decodedString.toString());
   }
}
```

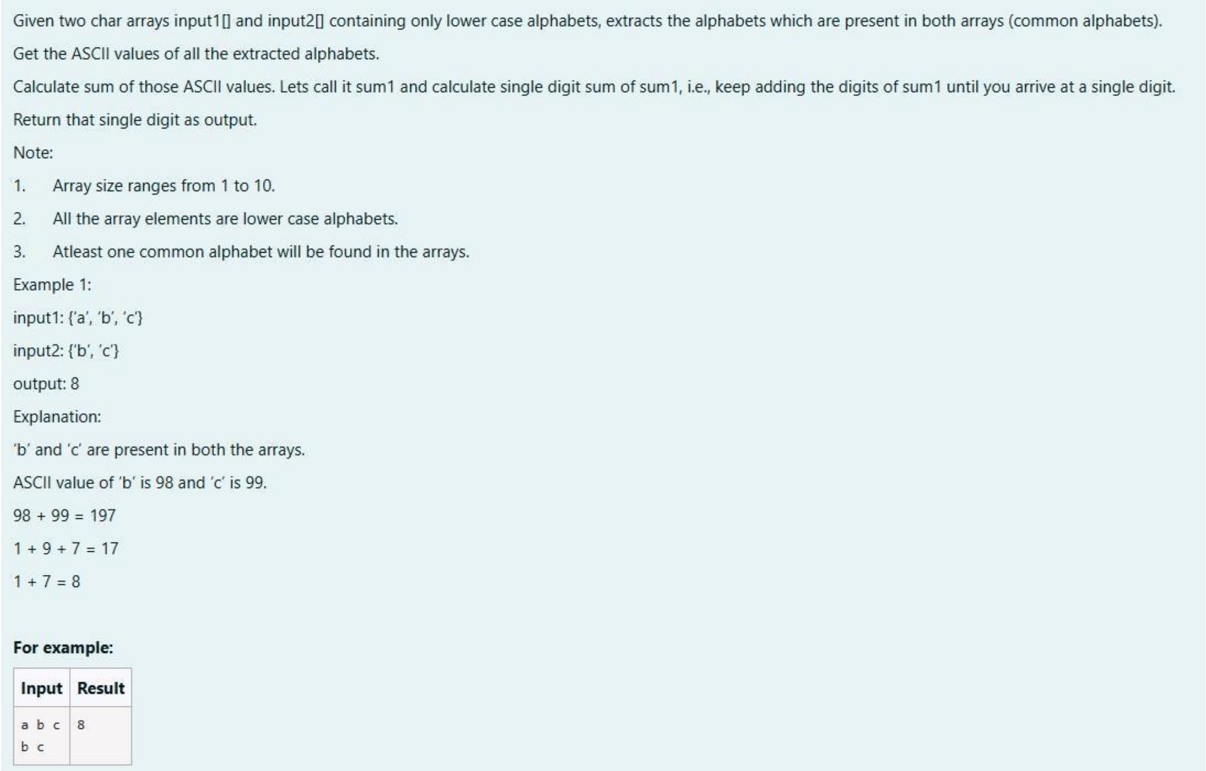
OUTPUT :



	Input	Expected	Got	
✓	010010001	ZYX	ZYX	✓
✓	00001000000000000000000001000000000001000000000100000000000001	WIPRO	WIPRO	✓

Passed all tests! ✓

2.



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.

Example 1:

input1: {'a', 'b', 'c'}

input2: {'b', 'c'}

output: 8

Explanation:

'b' and 'c' are present in both the arrays.

ASCII value of 'b' is 98 and 'c' is 99.

98 + 99 = 197

1 + 9 + 7 = 17

1 + 7 = 8

For example:

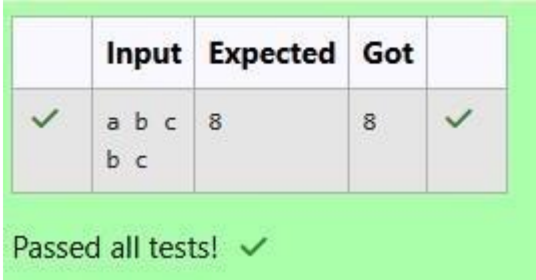
Input	Result
a b c b c	8

SOLUTION :

```
import java.util.HashSet; import java.util.Set; public class

CommonAlphabetSum {

   public static int singleDigitSum(int num) { int sum = 0;
      while (num > 0) { sum += num % 10; num /= 10;
      }
      if (sum > 9) { return singleDigitSum(sum); }
```

```
        return sum;
    }

    public static int calculateCommonAlphabetSum(char[] input1, char[] input2) {
        Set<Character> set1 = new HashSet<>(); for (char c : input1) { set1.add(c);
        }

        int sum = 0; for
        (char  c  :  input2)  {  if
            (set1.contains(c))    {
            sum += c;
            }
        }

        return singleDigitSum(sum);
    }

    public static void main(String[] args)
        { char[] input1 = {'a', 'b', 'c'};
        char[] input2 = {'b', 'c', 'd'};

        int result = calculateCommonAlphabetSum(input1, input2);
    System.out.println(result); }
}
```

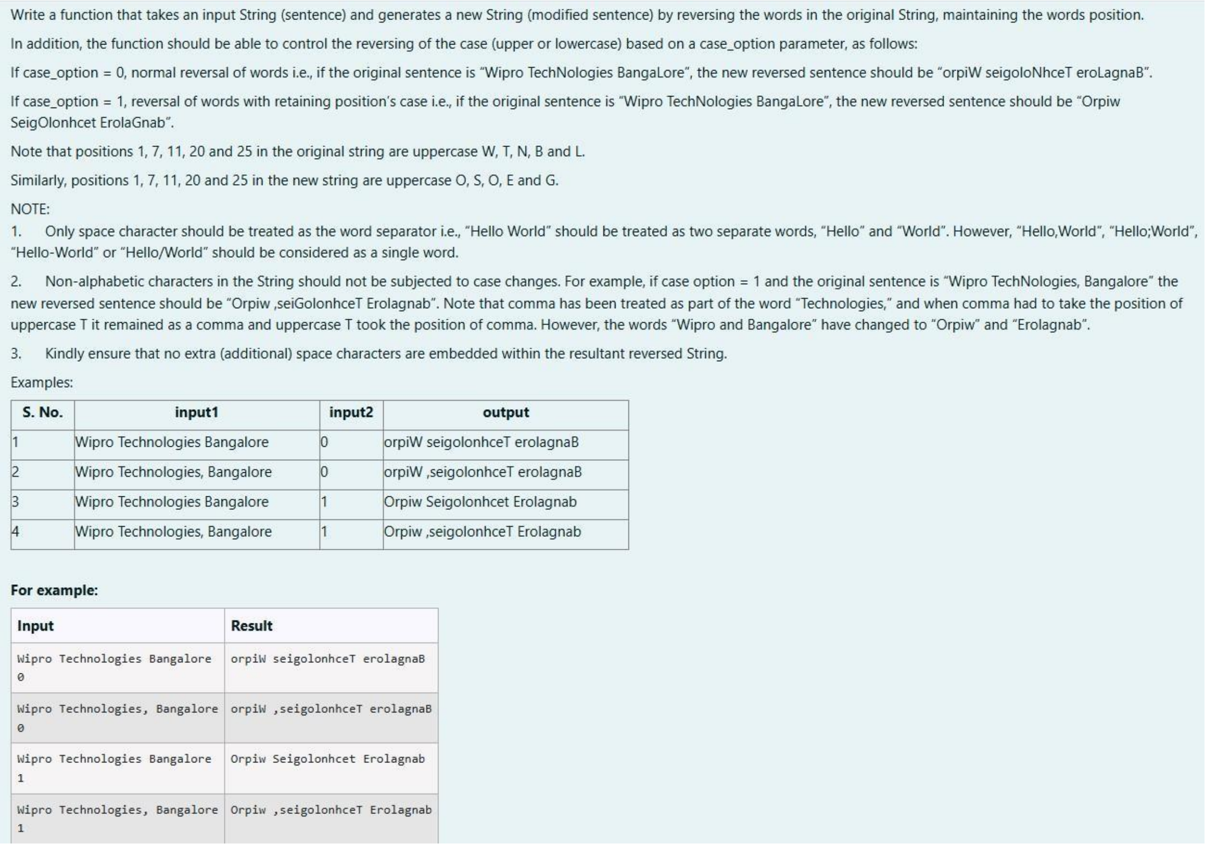
OUTPUT :



	Input	Expected	Got	
✓	a b c b c	8	8	✓

Passed all tests! ✓

3.



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.

Examples:

S. No.	input1	input2	output
1	Wipro Technologies Bangalore	0	orpiW seigolonhceT erolagnaB
2	Wipro Technologies, Bangalore	0	orpiW ,seigolonhceT erolagnaB
3	Wipro Technologies Bangalore	1	Orpiw Seigolonhcet Erolagnab
4	Wipro Technologies, Bangalore	1	Orpiw ,seigolonhceT Erolagnab

For example:

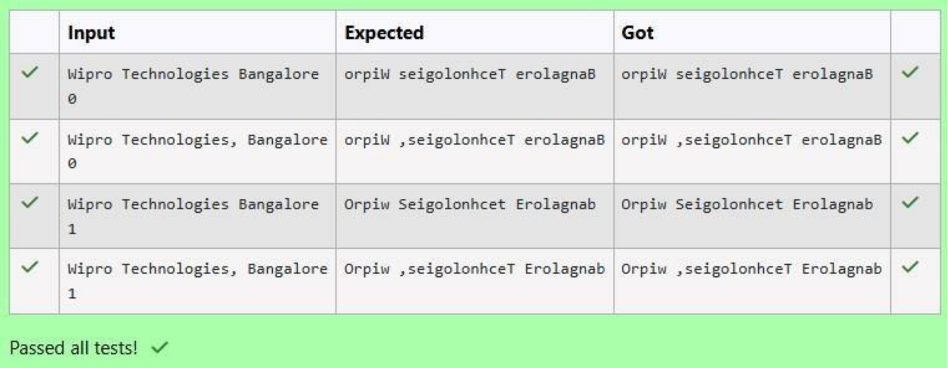
Input	Result
Wipro Technologies Bangalore 0	orpiW seigolonhceT erolagnaB
Wipro Technologies, Bangalore 0	orpiW ,seigolonhceT erolagnaB
Wipro Technologies Bangalore 1	Orpiw Seigolonhcet Erolagnab
Wipro Technologies, Bangalore 1	Orpiw ,seigolonhceT Erolagnab

SOLUTION :

```
import java.util.Scanner; public

class WordReverser {

  public static String reverseWordsWithCase(String sentence, int caseOption) {
    // Split the sentence into words based on spaces
    String[] words = sentence.split(" ");

    // StringBuilder to store the result
    StringBuilder result = new StringBuilder();

    // Process each word for
    (String word : words) {
      // Reverse the word
      String reversedWord = new StringBuilder(word).reverse().toString();

      if (caseOption == 0) {
        // If caseOption is 0, no case conversion, just reverse the word
        result.append(reversedWord).append(" ");
      } else if (caseOption == 1) {
        // If caseOption is 1, adjust the case while maintaining original letter
positions
```

```
        result.append(applyCaseConversion(reversedWord, word)).append(" ");
      }
    }

    // Remove the trailing space and return the result return
    result.toString().trim();
  }

  private static String applyCaseConversion(String reversedWord, String
originalWord) {
    // StringBuilder to store the adjusted word
    StringBuilder adjustedWord = new StringBuilder();

    // Iterate over each character in the reversed word
    for (int i = 0; i < reversedWord.length(); i++) { char
    reversedChar = reversedWord.charAt(i); char
    originalChar = originalWord.charAt(i);

      if (Character.isLowerCase(originalChar)) {
        // If the original character was lowercase, the reversed character should be
uppercase adjustedWord.append(Character.toLowerCase(reversedChar));
      } else if (Character.isUpperCase(originalChar)) {
        // If the original character was uppercase, the reversed character should be
lowercase adjustedWord.append(Character.toUpperCase(reversedChar));
      } else {
        // Non-alphabetic characters remain unchanged
      adjustedWord.append(reversedChar); }
    }

    return adjustedWord.toString();
  }

  public static void main(String[] args) {
    // Create a Scanner object to get input from the user Scanner
    scanner = new Scanner(System.in);

    // Get sentence input from the user

    String sentence = scanner.nextLine(); //

    Get case option input from the user int

    caseOption = scanner.nextInt();

    // Validate the case option
    if (caseOption != 0 && caseOption != 1) {
            System.out.println("Invalid case option. Please enter 0 or 1.");
        } else {
            // Call the function and print the result
            String result = reverseWordsWithCase(sentence, caseOption);
            System.out.println(result);
        }

        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT :



	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! ✓

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 - 231001059

SWETHA A - 231001226

Of

BACHELOR OFTECHNOLOGY

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 : **SIGNATURE:**

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

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

1. 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.

2. 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.

3. 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.

4. 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 andAcknowledgement**

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 for typical 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:** o 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

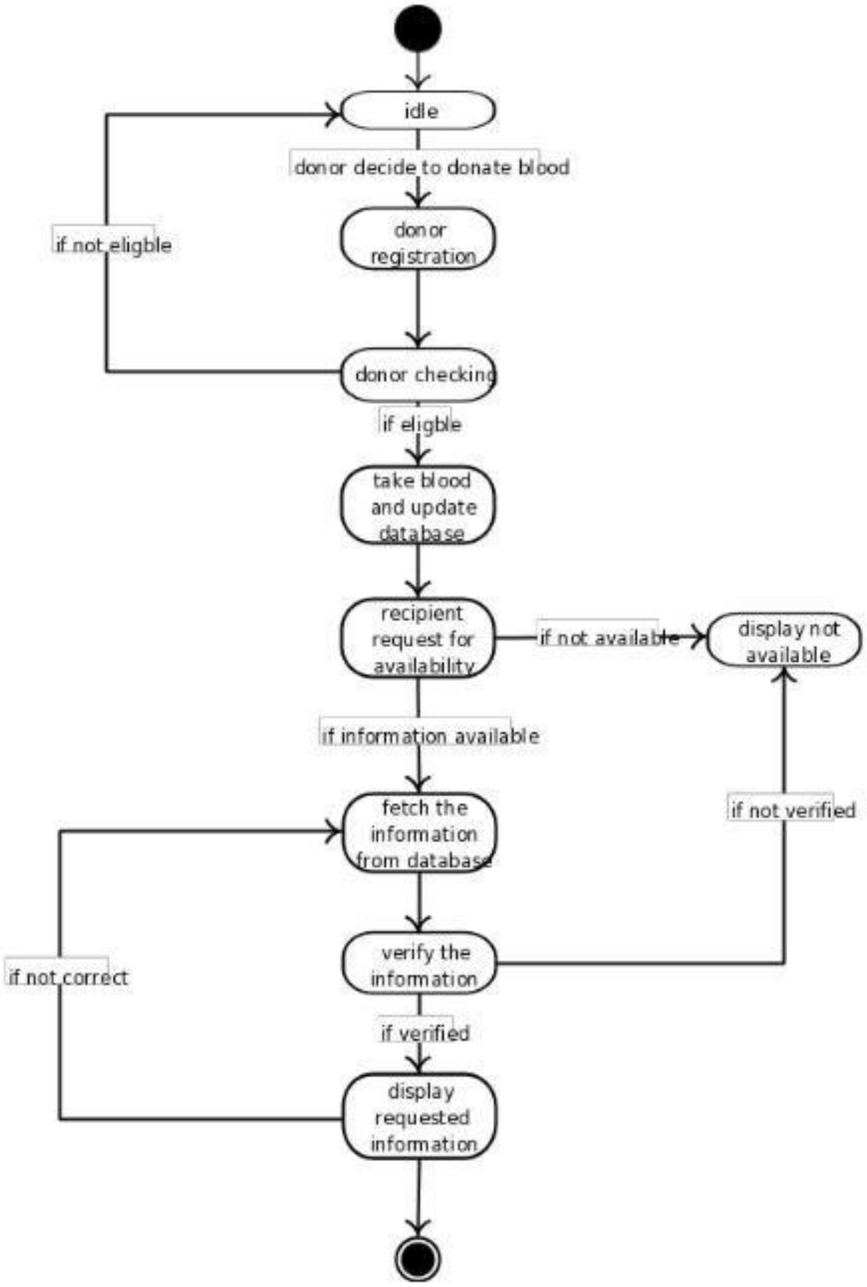


Figure 2.1 Use Case Diagrams

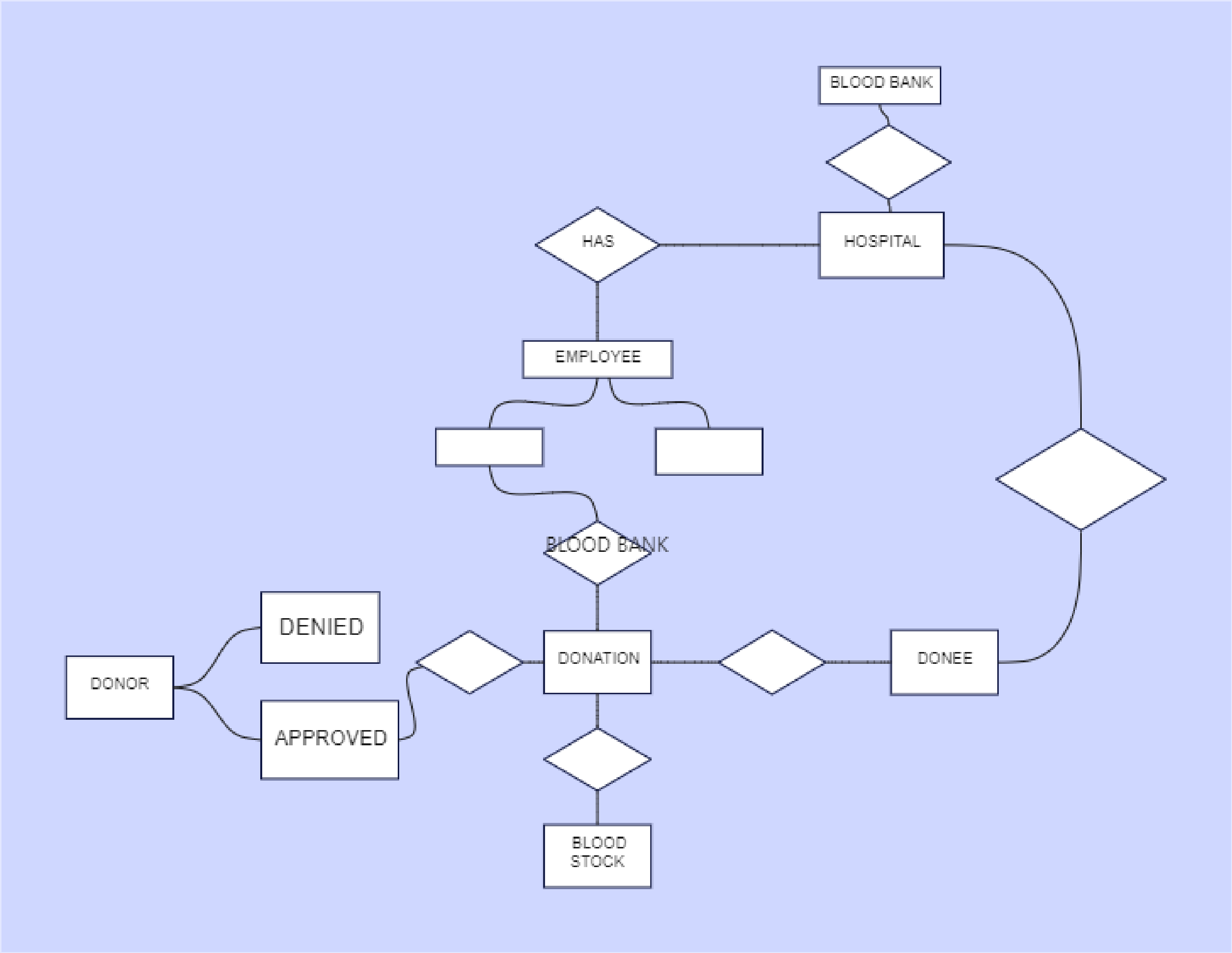


Figure 2.2 Entity Relationship Diagram

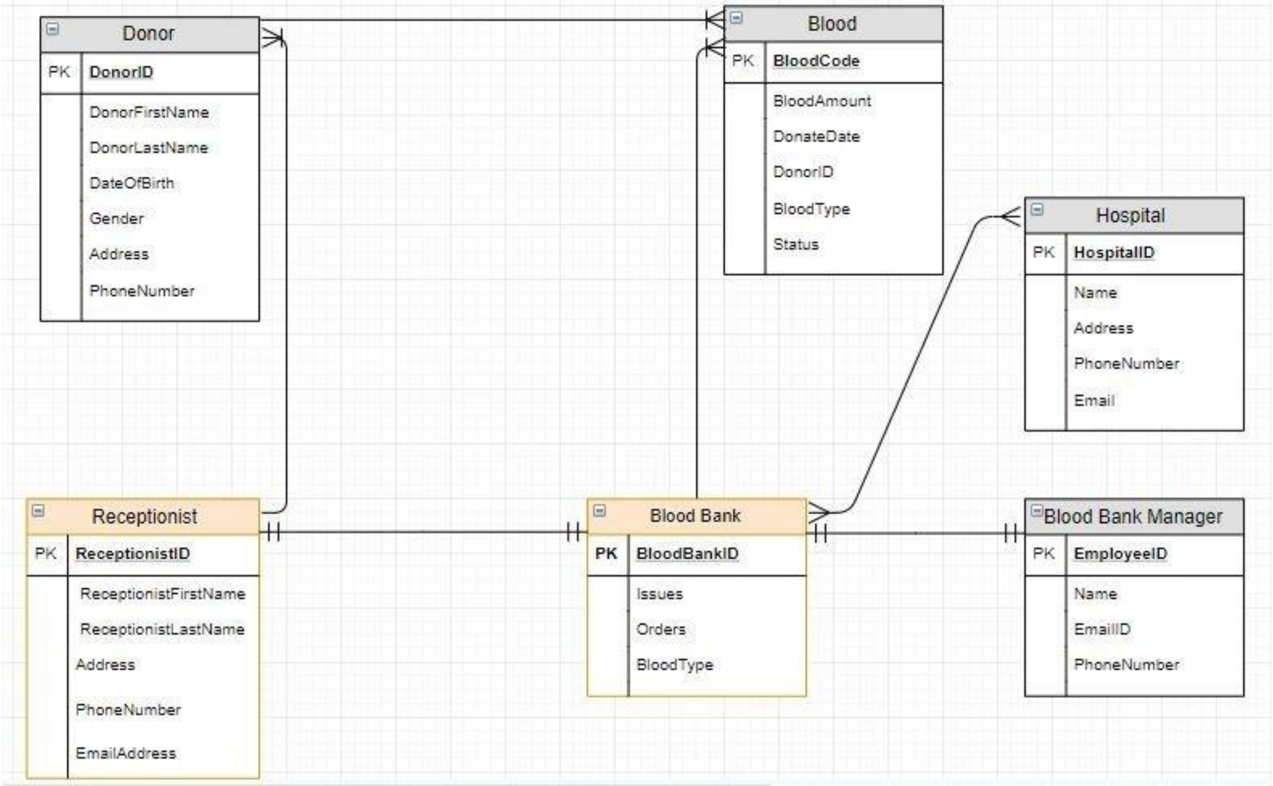


Figure 2.3 Data-flow diagram

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 Sol or Elasticsearch for integrating efficient search functionalities into the system.

Analytics Tools: Google Analytics for tracking user activity and be-havior 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

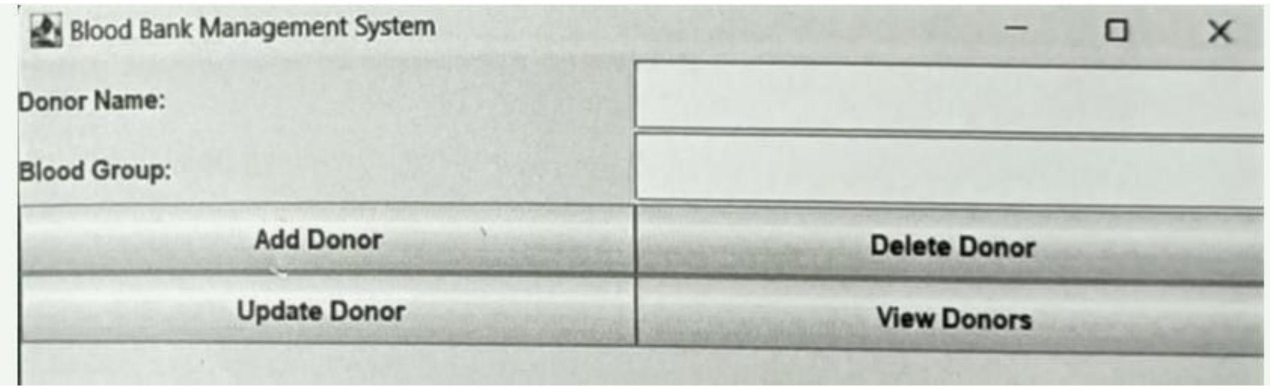
□ Tools Used:

- 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:** o **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:

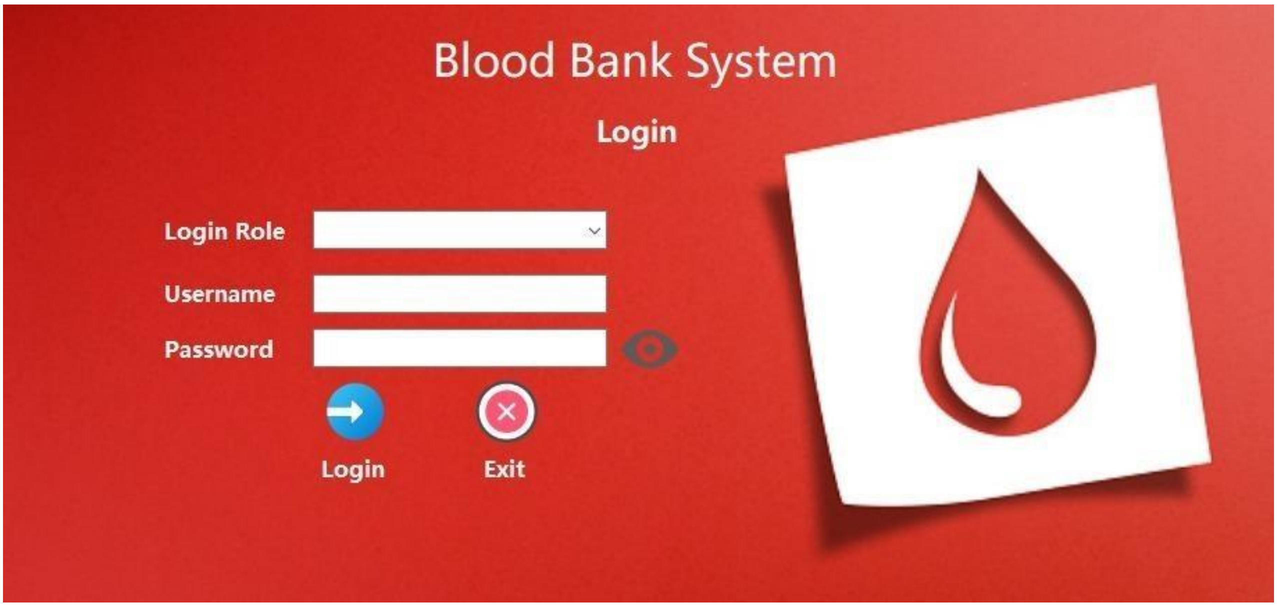


Blood Bank Management System

Donor Name:

Blood Group:

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.

o email: The email address of the donor. o phone: Contact

number of the donor. o blood_type: The blood type of the

donor (e.g., A+, O-, etc.). o last_donation_date: The date of the

most recent blood donation.

o 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).

---

3) 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. o name: The name of the admin. o email: The admin's email address. o 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);
```

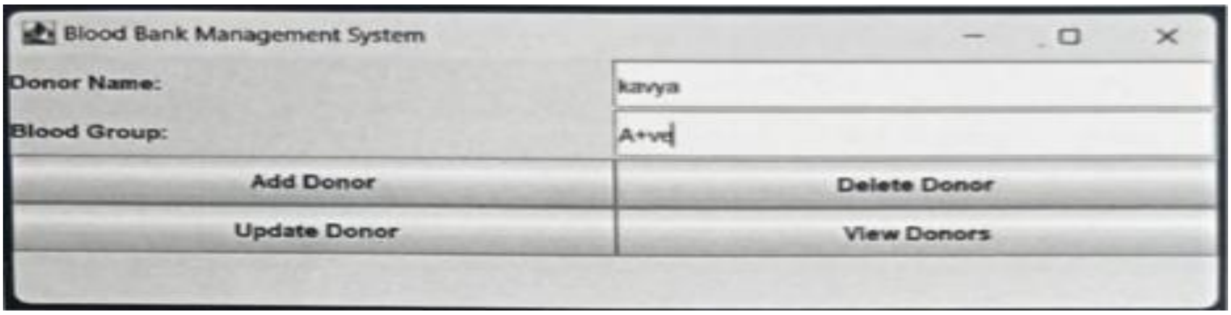
```
contentPane.add(textField_1);
                          bloodGroupLabel = new JLabel("Blood Group: ");
bloodGroupLabel.setFont(new Font("Tahoma",
Font.BOLD, 16));              bloodGroupLabel.setBounds(37,
302, 140, 111);
contentPane.add(bloodGroupLabel);


                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 :

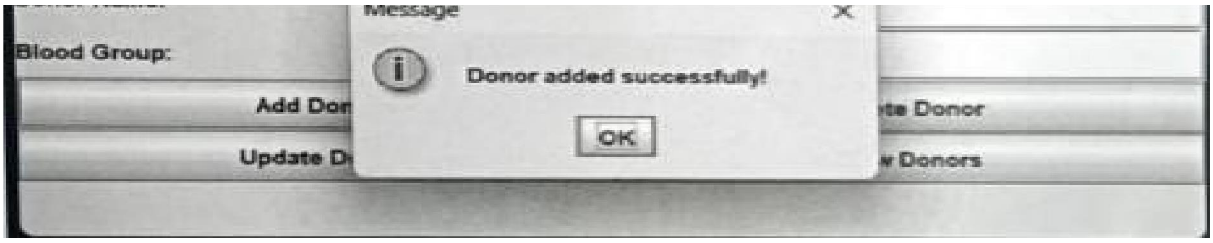
ADDING DONOR AND VIEWING DONOR DETAILS:


Blood Bank Management System

Donor Name: kavya

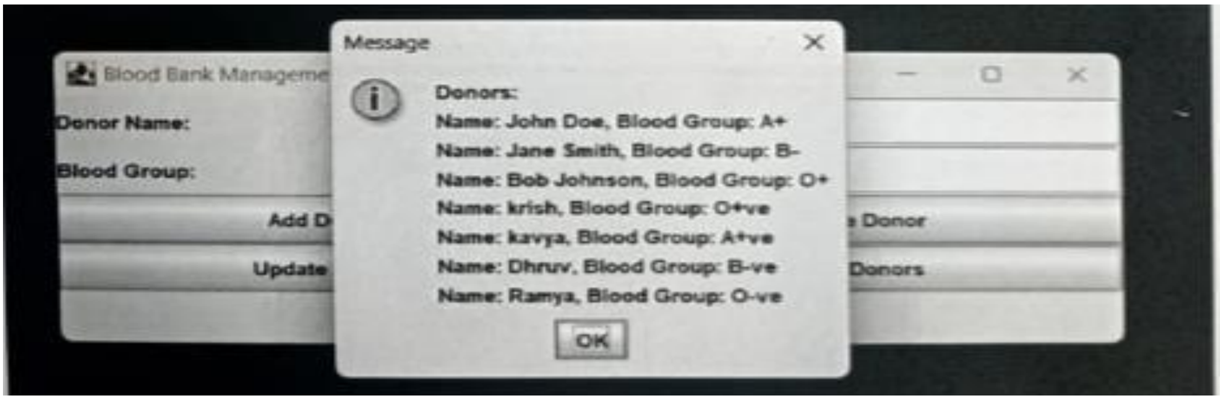
Blood Group: A+ve

FIG.4.3.1 ADDING DONOR DETAIL


Message

Donor added successfully!

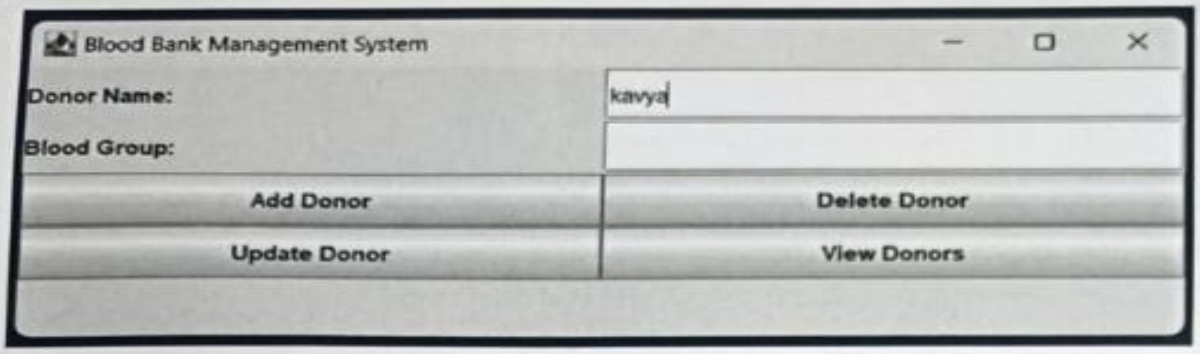
FIG.4.3.2 DONOR DETAIL ADDED SUCCESSFULLY


Message

Donors:
Name: John Doe, Blood Group: A+
Name: Jane Smith, Blood Group: B-
Name: Bob Johnson, Blood Group: O+
Name: krish, Blood Group: O+ve
Name: kavya, Blood Group: A+ve
Name: Dhruv, Blood Group: B-ve
Name: Ramya, Blood Group: O-ve

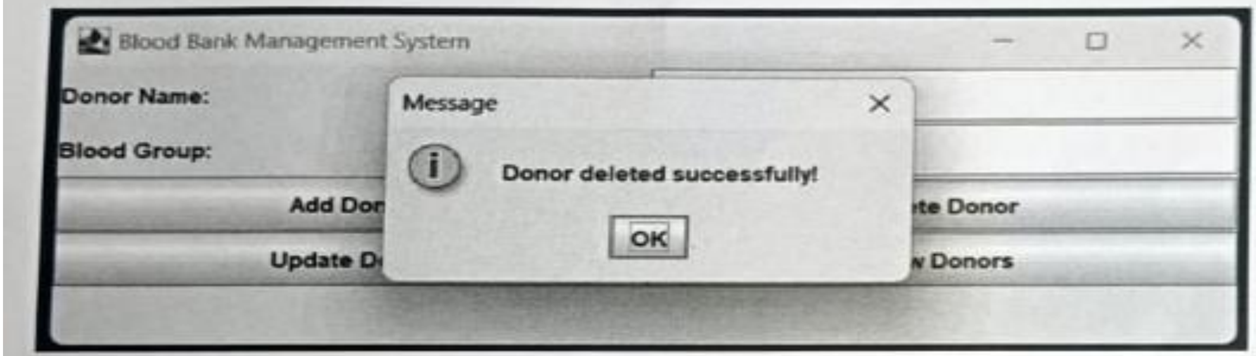
FIG.4.3.3 VIEWING DETAILS OF DONOR AFTER ADDITION

» DELETING DONOR AND VIEWING DONOR DETAILS:



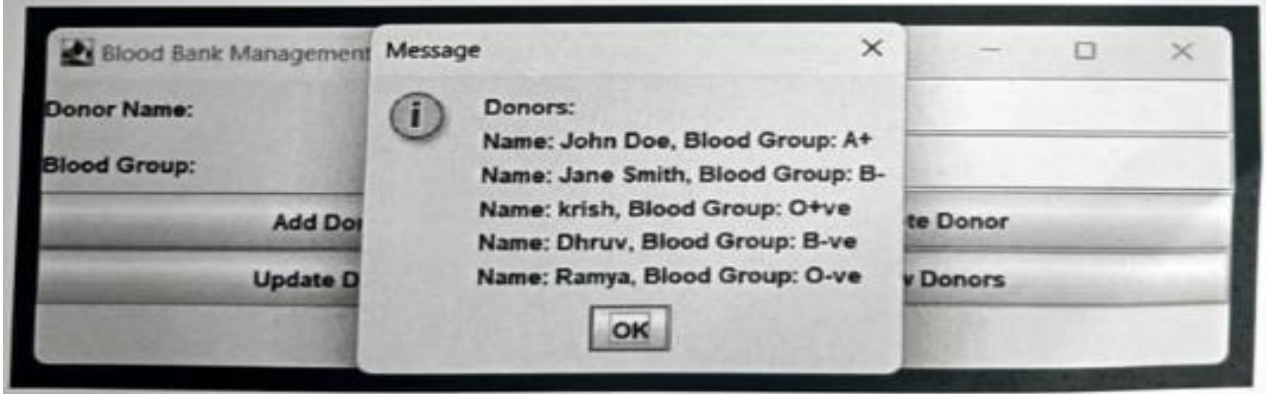
Donor Name: kavya

Fig.4.3.4 DELETING DONOR DETAIL



Donor deleted successfully!

Fig.4.3.5 DONOR DELETED SUCCESSFULLY



Donors:
Name: John Doe, Blood Group: A+
Name: Jane Smith, Blood Group: B-
Name: krish, Blood Group: O+ve
Name: Dhruv, Blood Group: B-ve
Name: Ramya, Blood Group: O-ve

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**: o 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**: o Deploy the application on a server using a platform like **Heroku**, **AWS**, or **Azure**. o 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 o Beginner-friendly Java programming guide.
2. Effective Java by Joshua Bloch o Best practices for writing clean, efficient, and maintainable Java code.
3. Java: The Complete Reference by Herbert Schildt o 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.

○