RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602 105

CS23333 OBJECT ORIENTED PROGRAMING USING JAVA
Laboratory Record Note Book

Name : .Vaishali S· · · · · · · · · · · · · · · ·

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

College Roll No. : . . . . 231001235……. . . . .

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

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

RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name : . .VAISHALI . S .. . .

Academic Year : 2024-2025 Semester:. III Branch : IT-D

Register No. 2116231001235

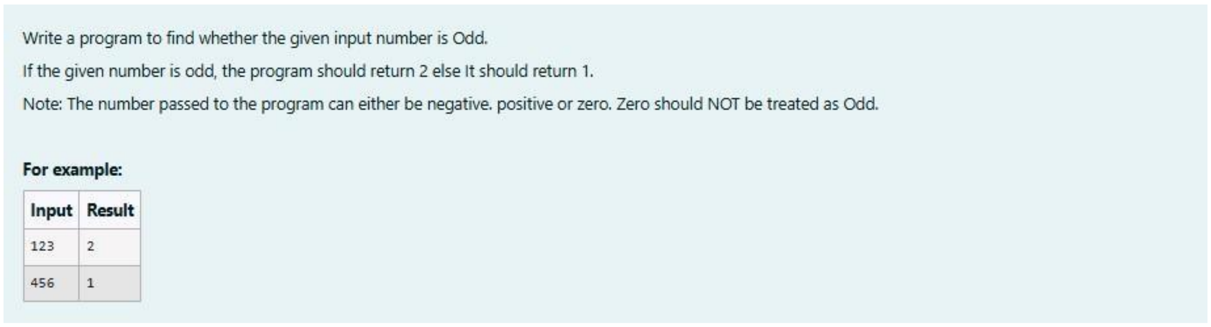
Certified that this is the bonafide record of work done by the above student in the CS23333 –Object Oriented Programming using JAVA during the year 2024 - 2025.

Signature of Faculty in-charge

Submitted for the Practical Examination held on . . . .

Internal Examiner **External Examiner**

INDEX


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.

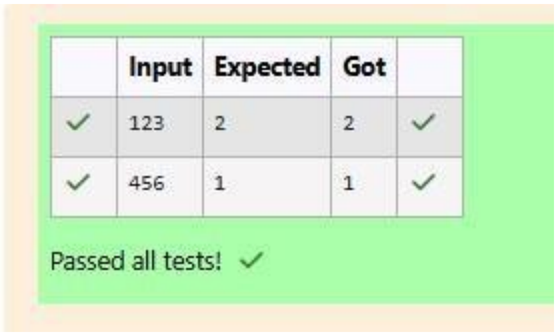
For example:

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);
}
}
}
```

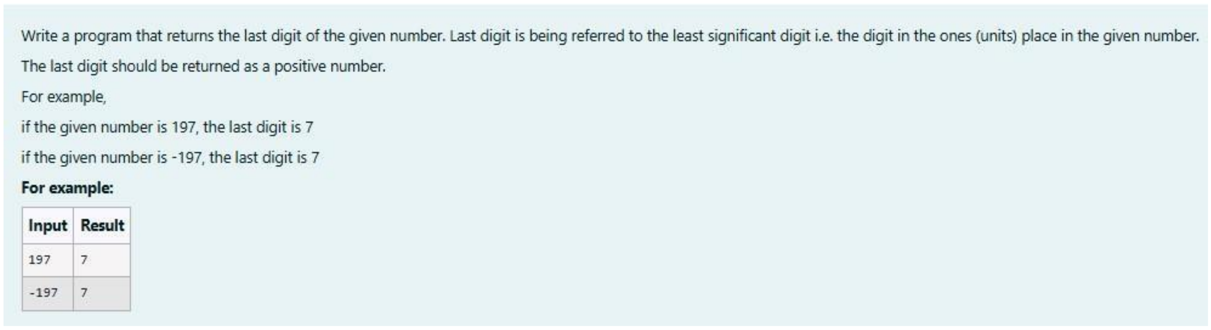
OUTPUT :



	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

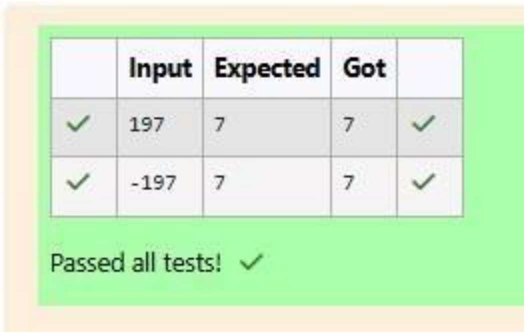
For example:

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);
    }
    }
```

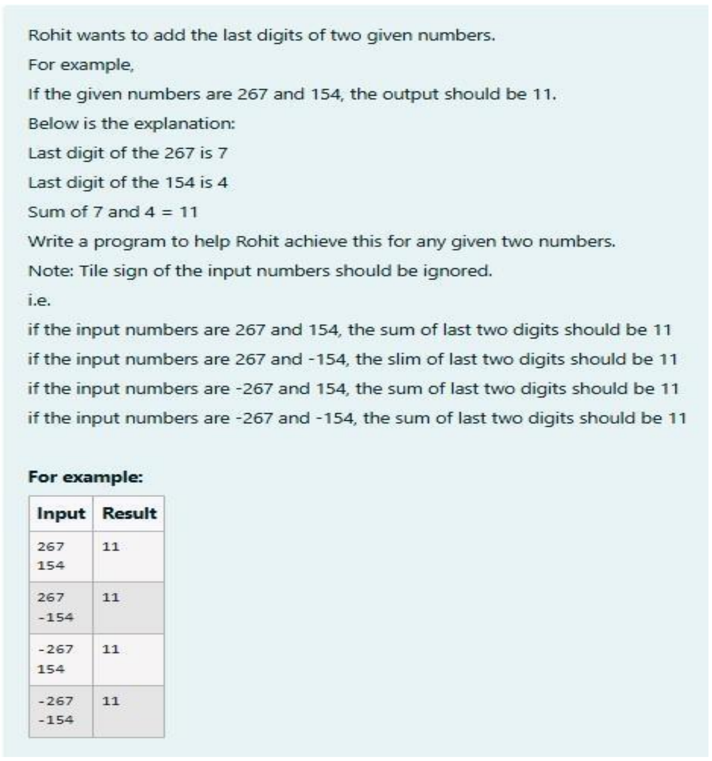
OUTPUT :



	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

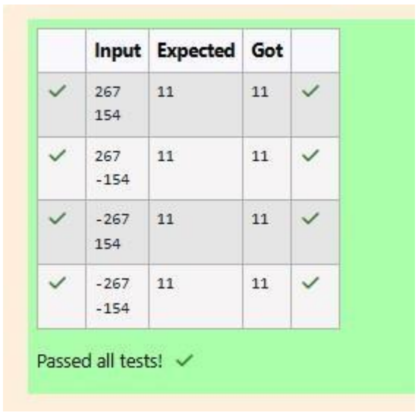
For example:

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:

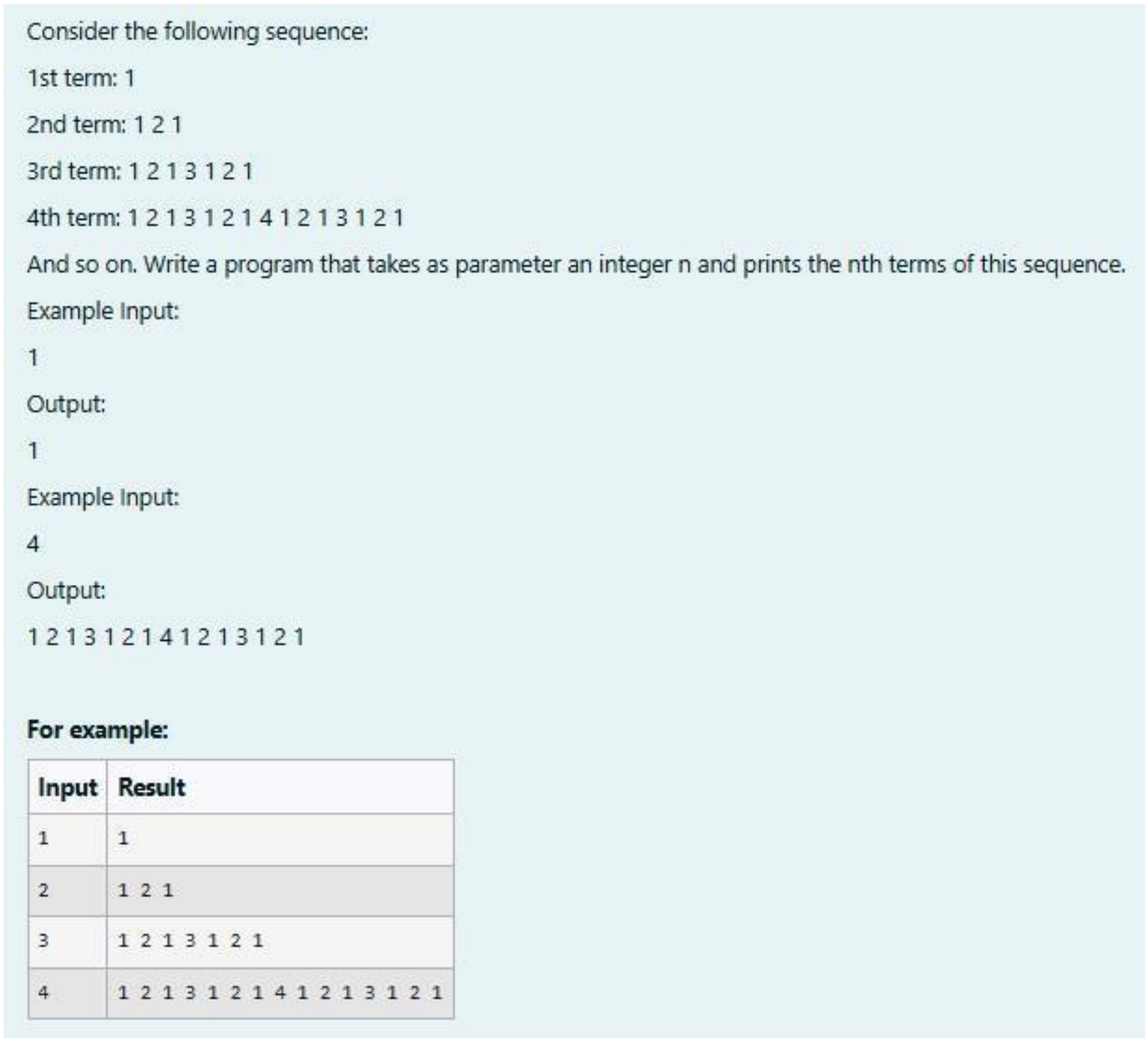


	Input	Expected	Got	
✓	267 154	11	11	✓
✓	267 -154	11	11	✓
✓	-267 154	11	11	✓
✓	-267 -154	11	11	✓

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

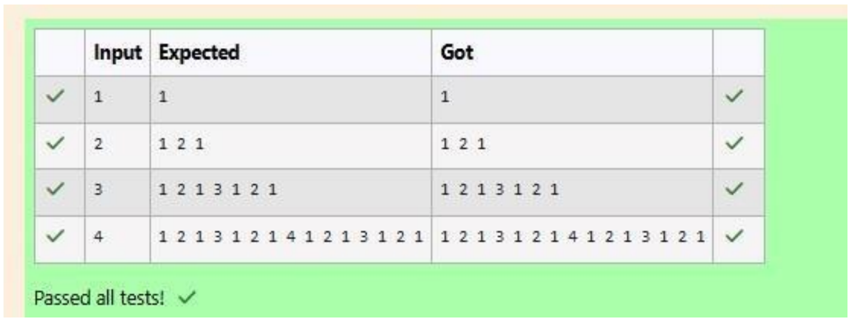
For example:

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();
}
}
```

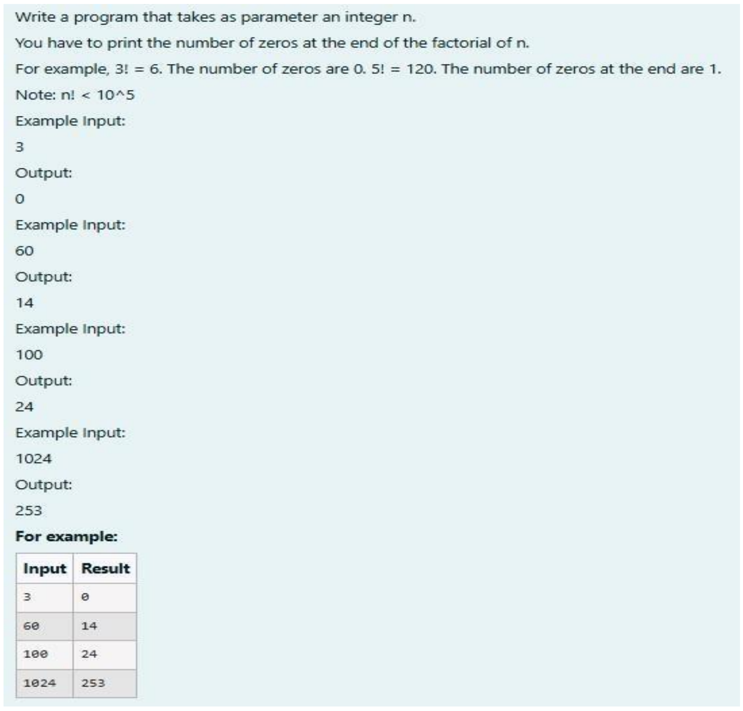
OUTPUT :



	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:

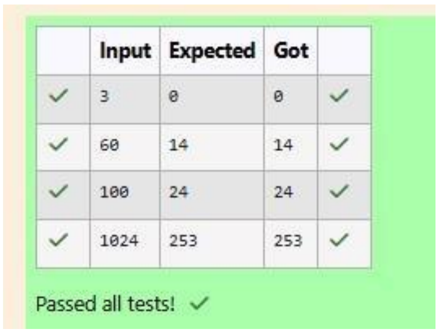
Input	Result
3	0
60	14
100	24
1024	253

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); }
}
```

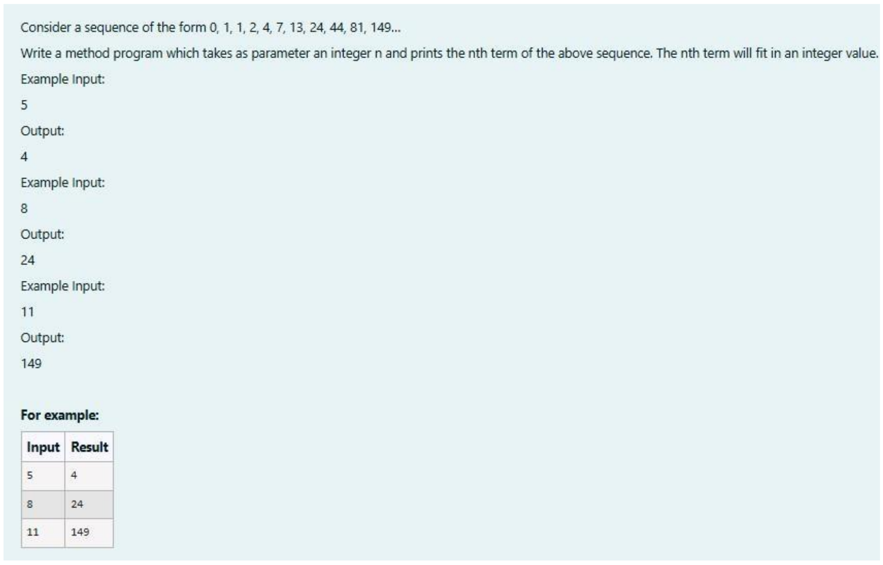
OUTPUT :



	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

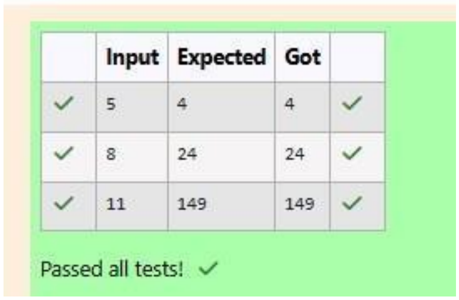
For example:

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 :

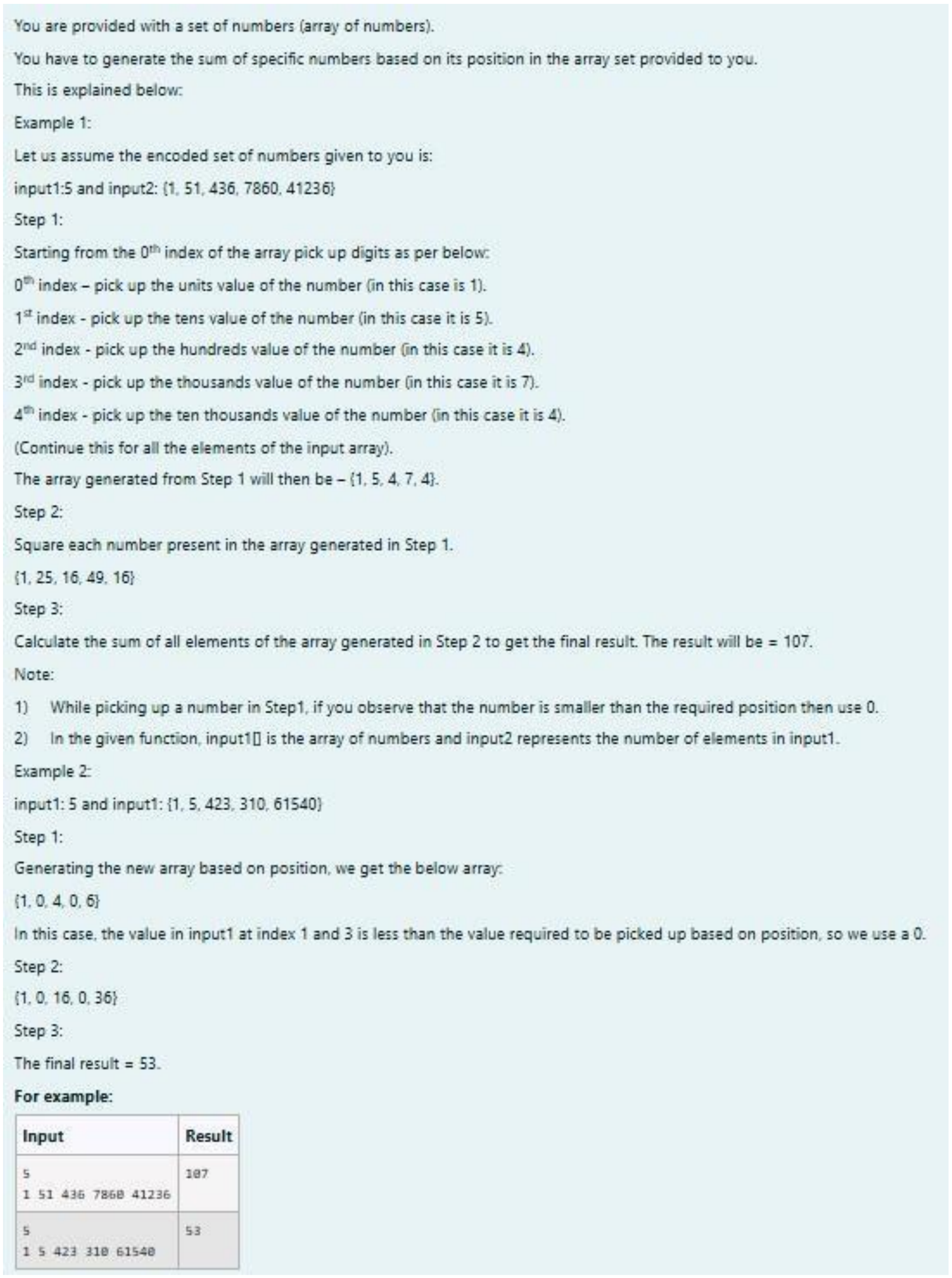


	Input	Expected	Got	
✓	5	4	4	✓
✓	8	24	24	✓
✓	11	149	149	✓

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:

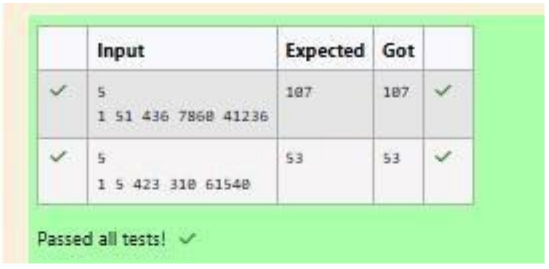
Input	Result
5 1 51 436 7860 41236	107
5 1 5 423 310 61540	53

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);
    }
}
```

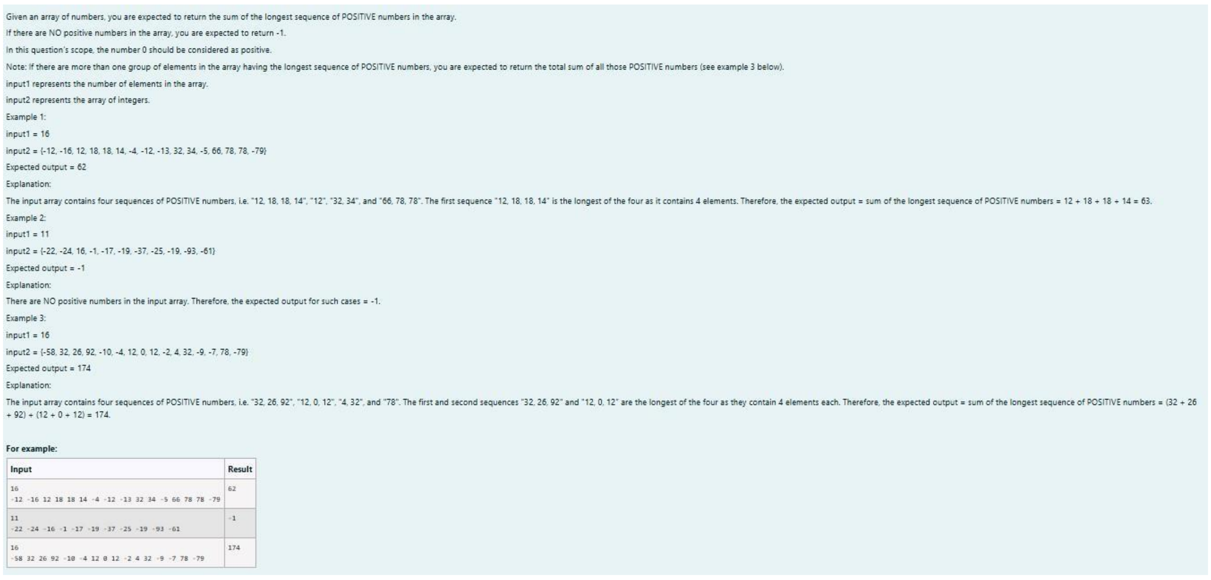
OUTPUT :



	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.

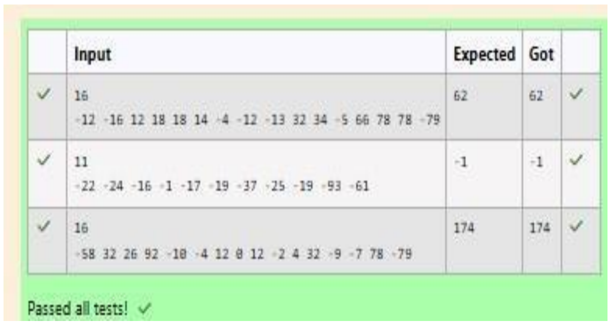
For example:

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);
  }
  }
```

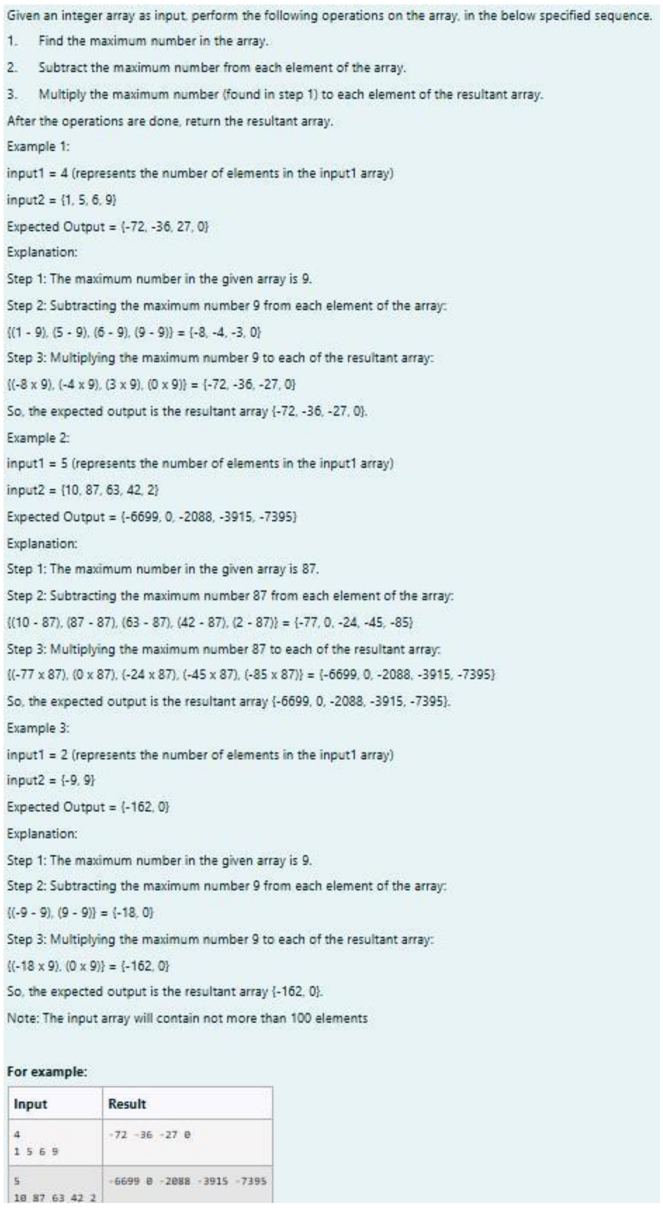
OUTPUT :



	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

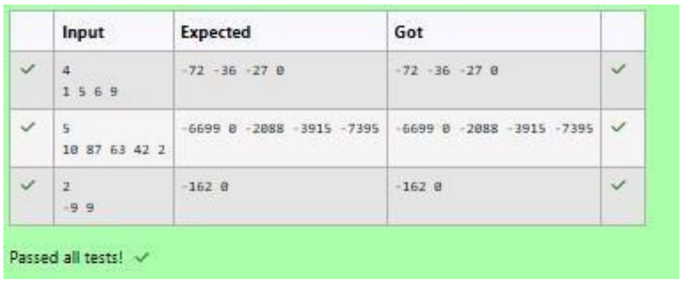
For example:

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 :

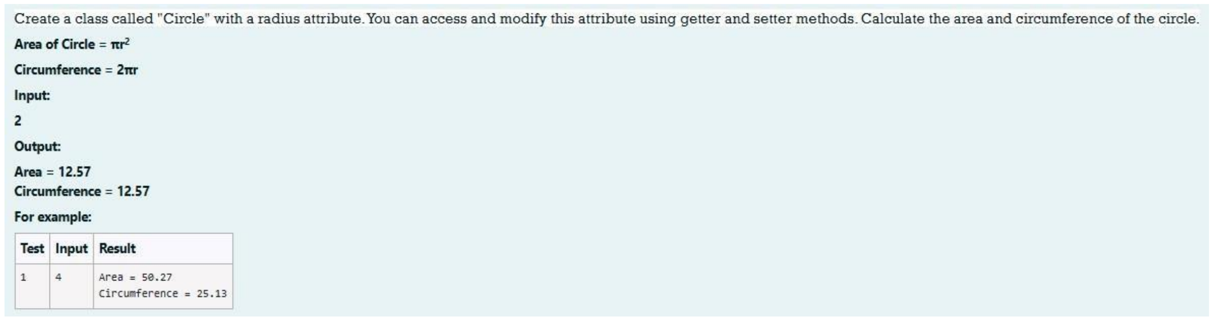


	Input	Expected	Got	
✓	4 1 5 6 9	-72 -36 -27 0	-72 -36 -27 0	✓
✓	5 10 87 63 42 2	-6699 0 -2088 -3915 -7395	-6699 0 -2088 -3915 -7395	✓
✓	2 -9 9	-162 0	-162 0	✓

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:

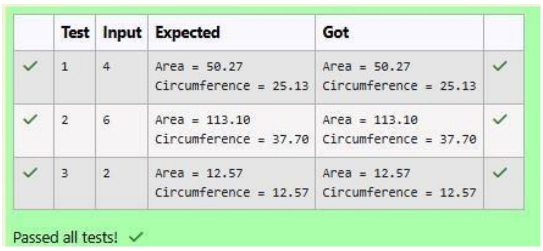
Test	Input	Result

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();
   }
}
```

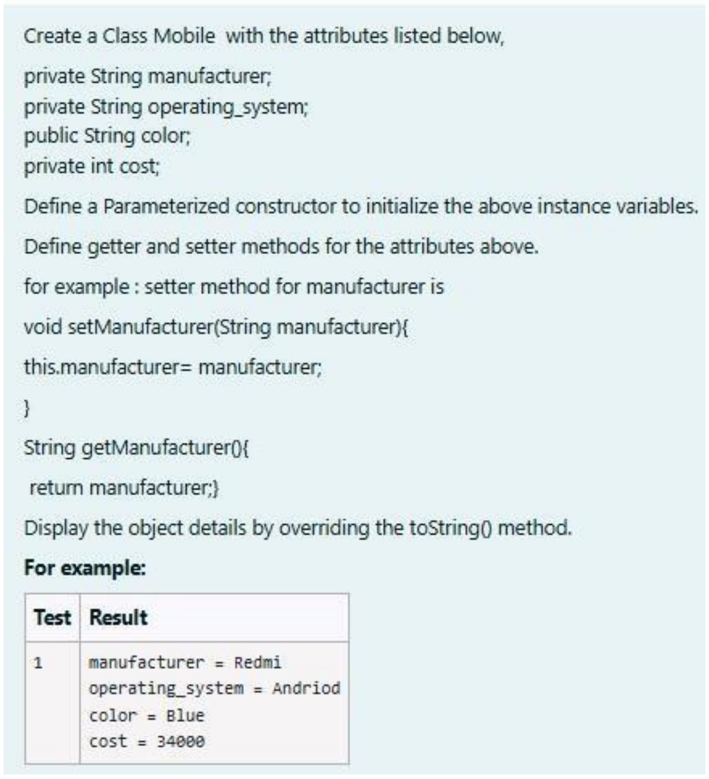
OUTPUT :



	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:

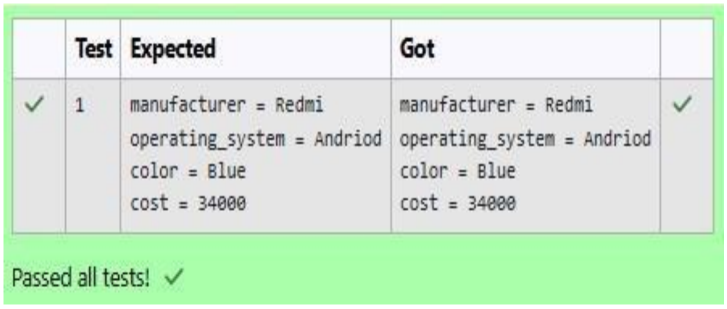
Test	Result

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 :

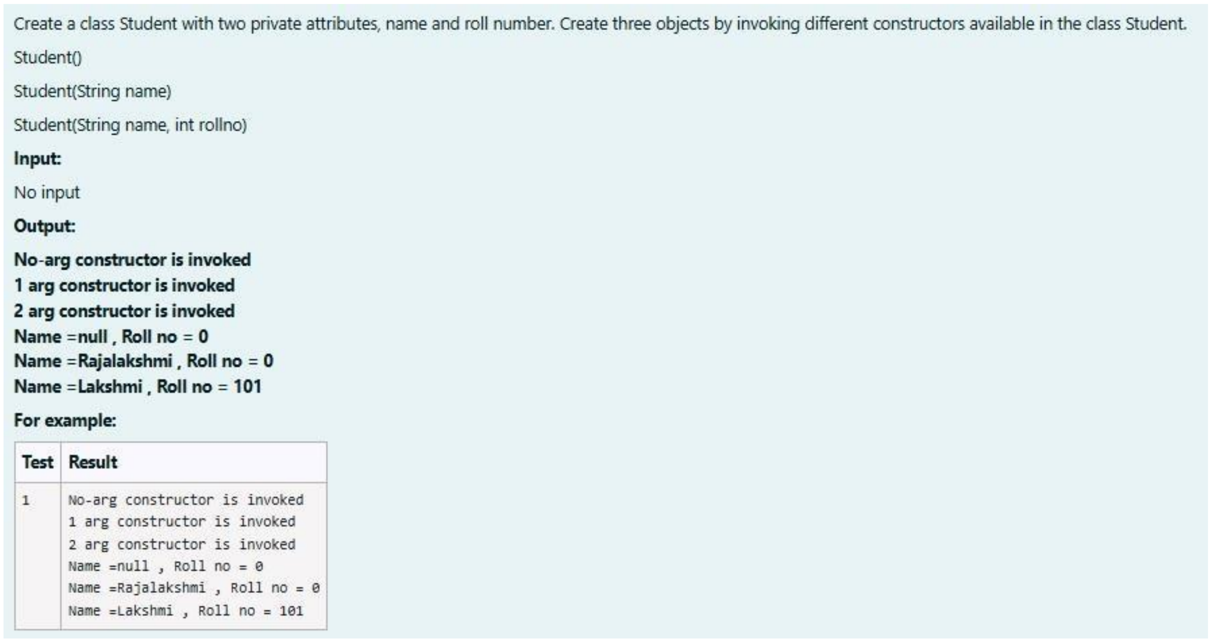


	Test	Expected	Got	

✓	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:

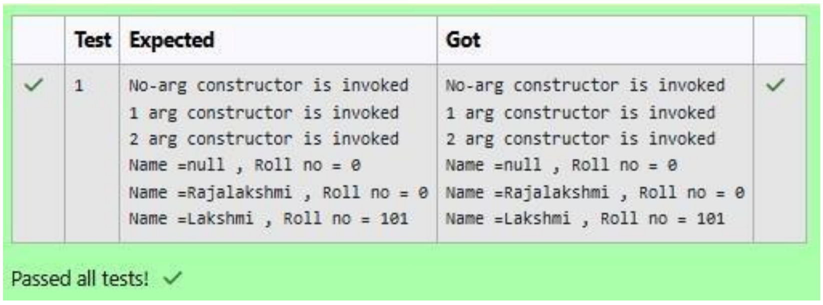
Test	Result

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 :



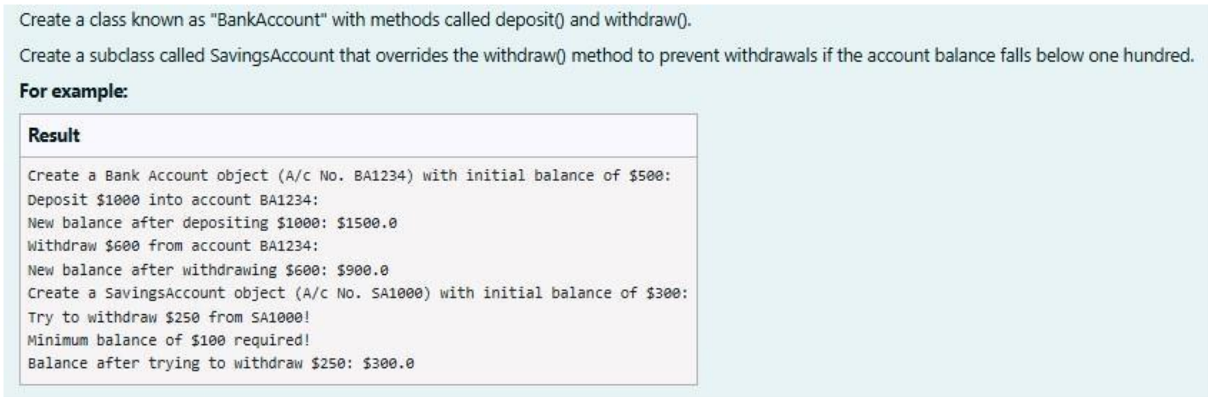
	Test	Expected	Got	

✓	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 :

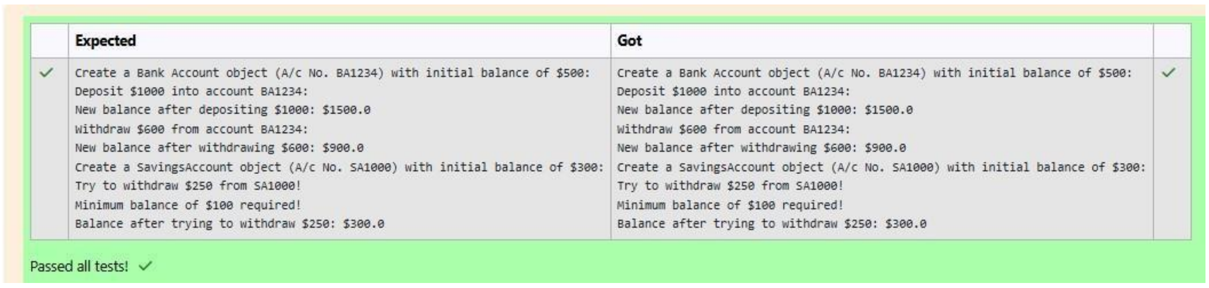
```
class BankAccount {
// Private field to store the account number
private String accountNumber;
// Private field to store the balance
```

```
   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()); } }
```

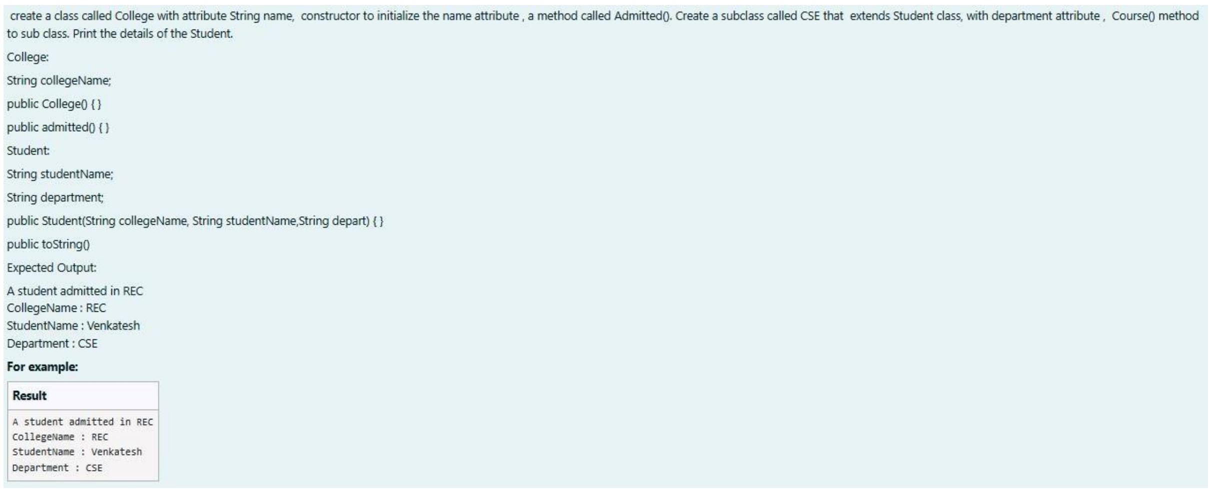
OUTPUT :



	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

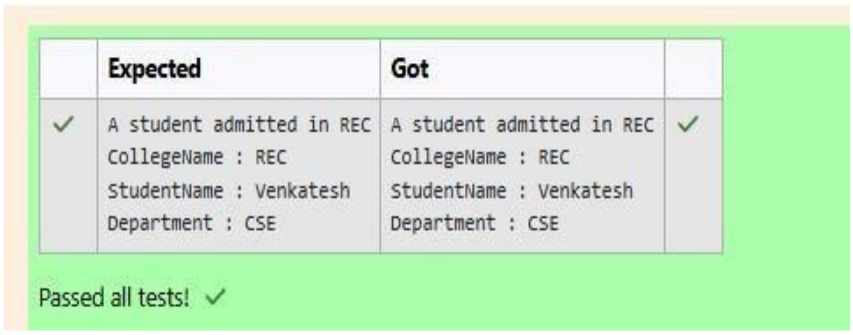
For example:

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());
}
}
```

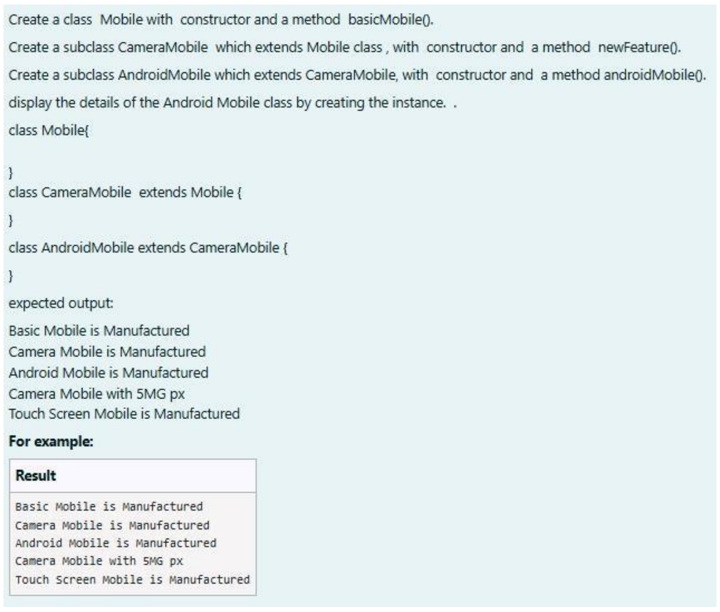
OUTPUT :



	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:

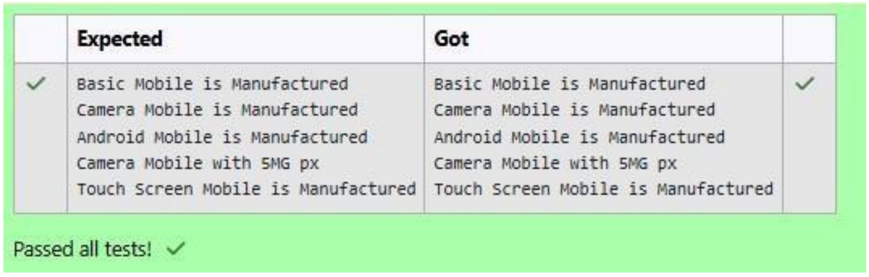
Result
Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured

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 :

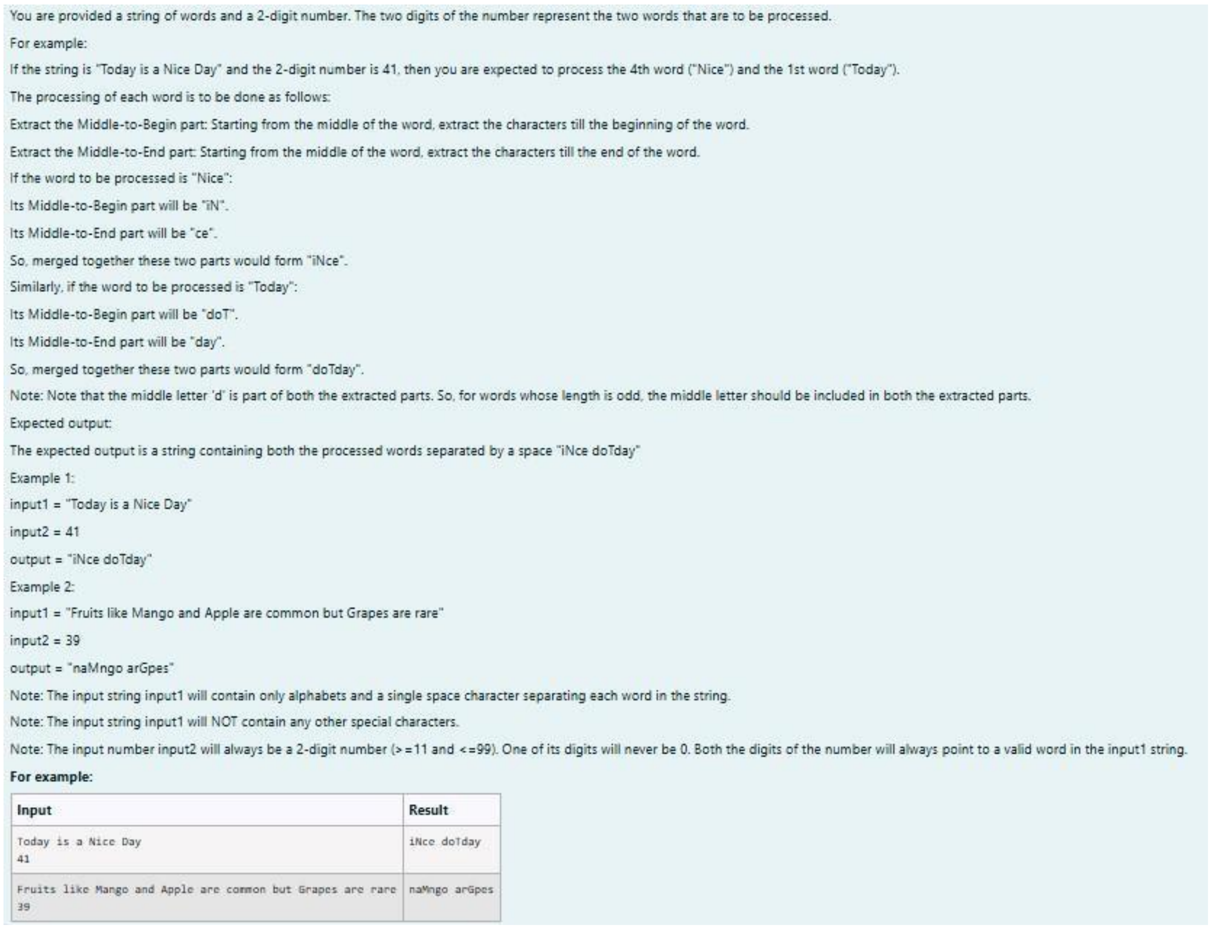


	Expected	Got	
✓	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured	Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile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

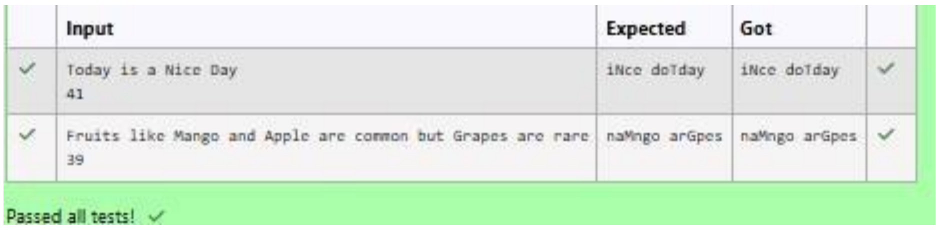
Input	Result
Today is a Nice Day 41	iNce doTday
Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes

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(); }
}
```

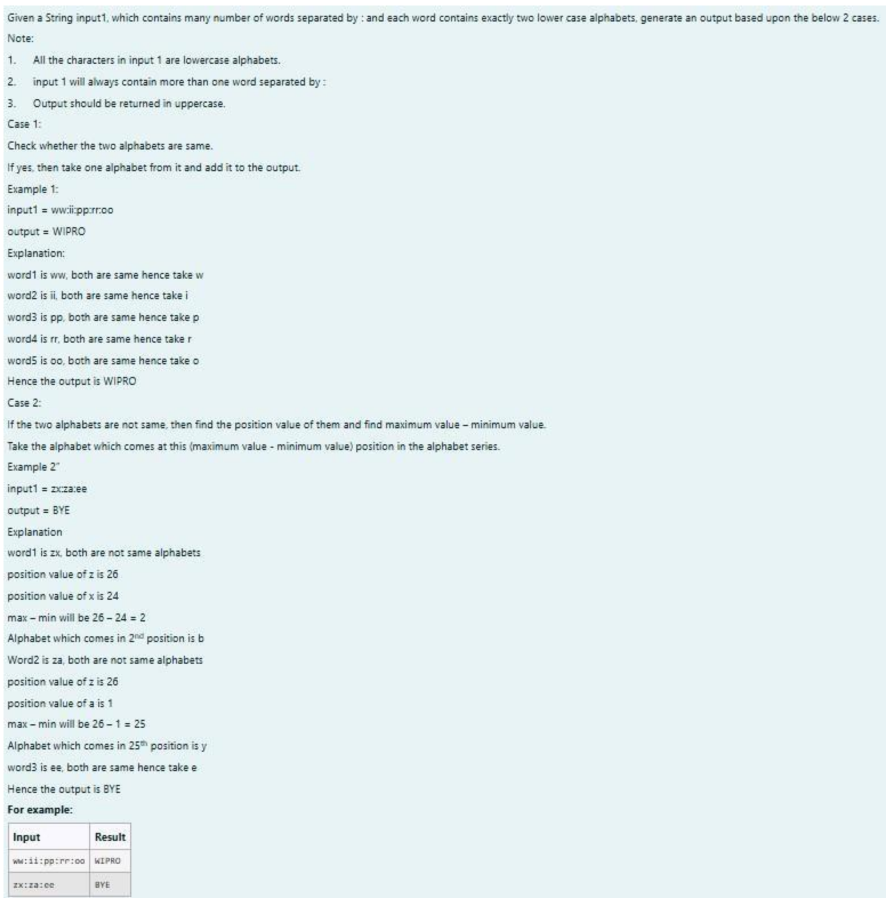
OUTPUT :



	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

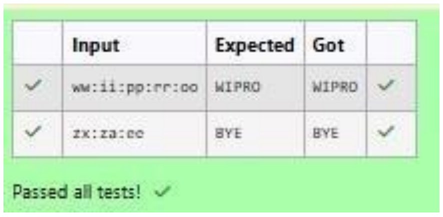
For example:

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());
    }
}
```

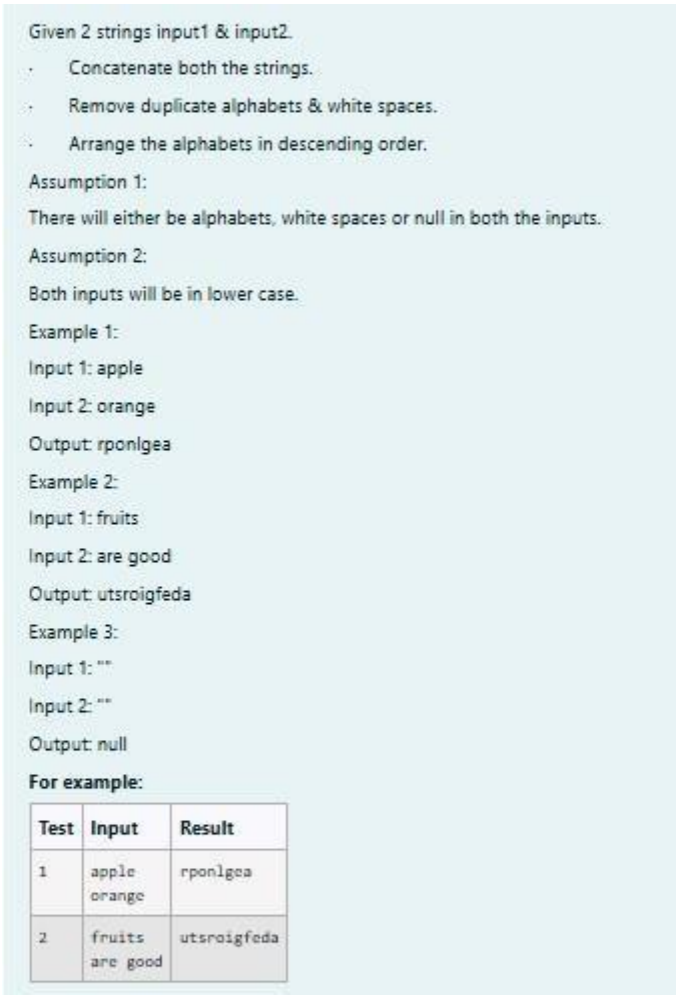
OUTPUT :



	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

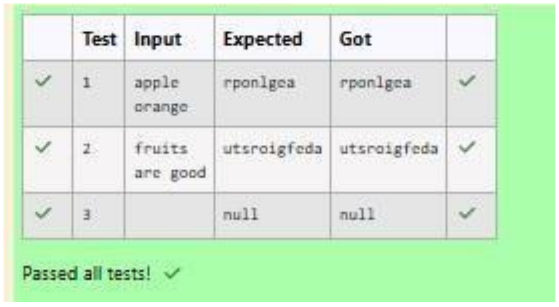
For example:

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 :

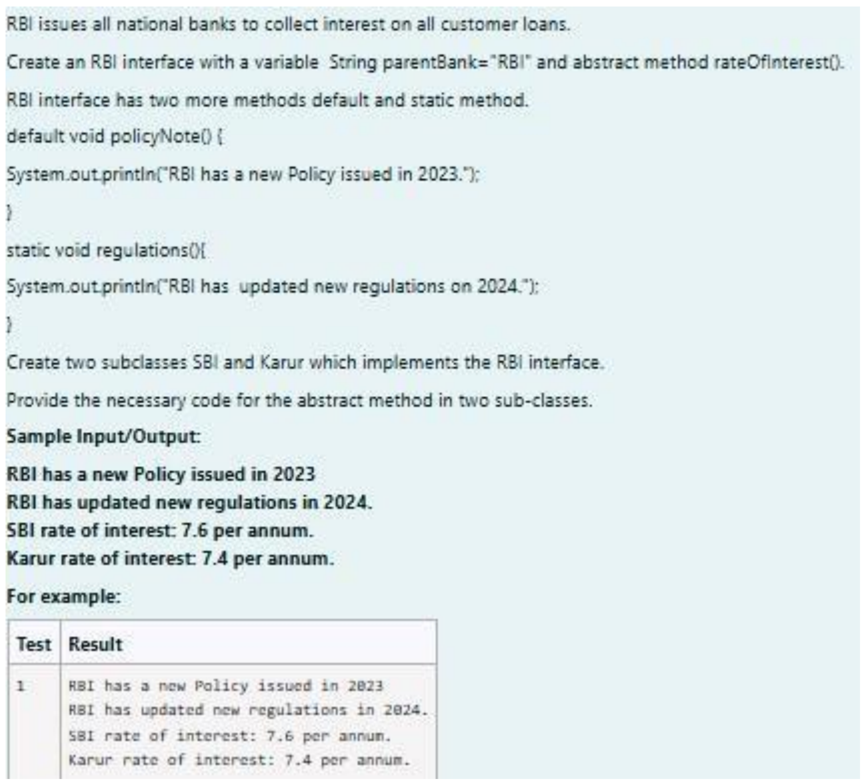


	Test	Input	Expected	Got	
✓	1	apple orange	rponlgea	rponlgea	✓
✓	2	fruits are good	utsroigfeda	utsroigfeda	✓
✓	3		null	null	✓

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.

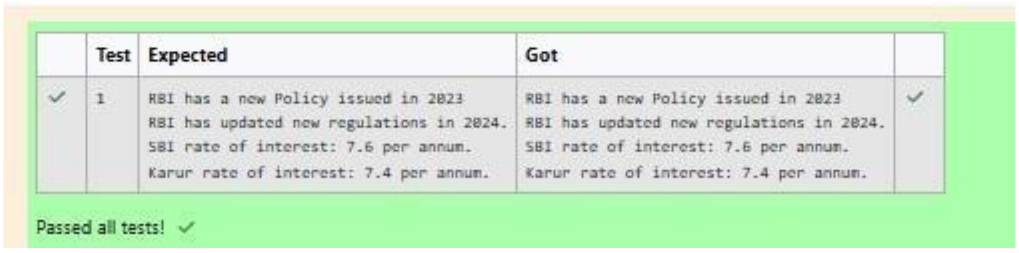
For example:

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.");
    }
}
```

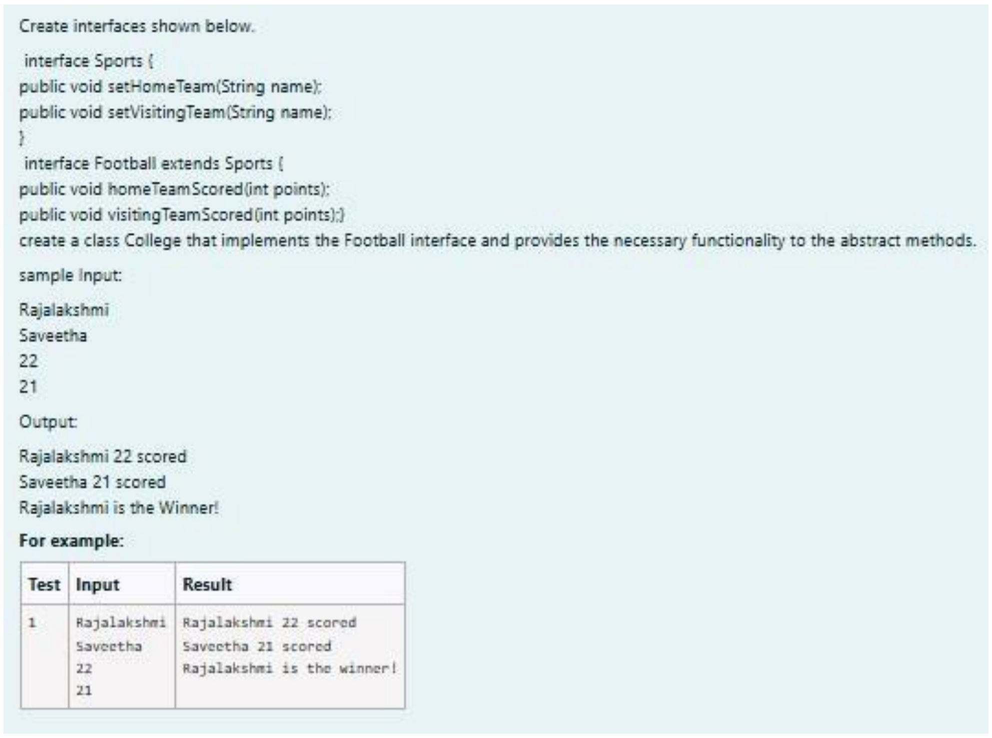
OUTPUT :



	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:

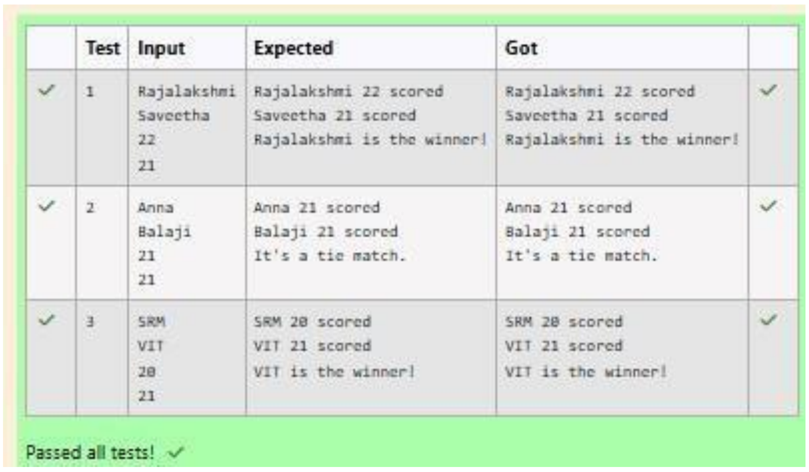
Test	Input	Result
1	Rajalakshmi Saveetha 22 21	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!

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();
    }
```

```
}
```

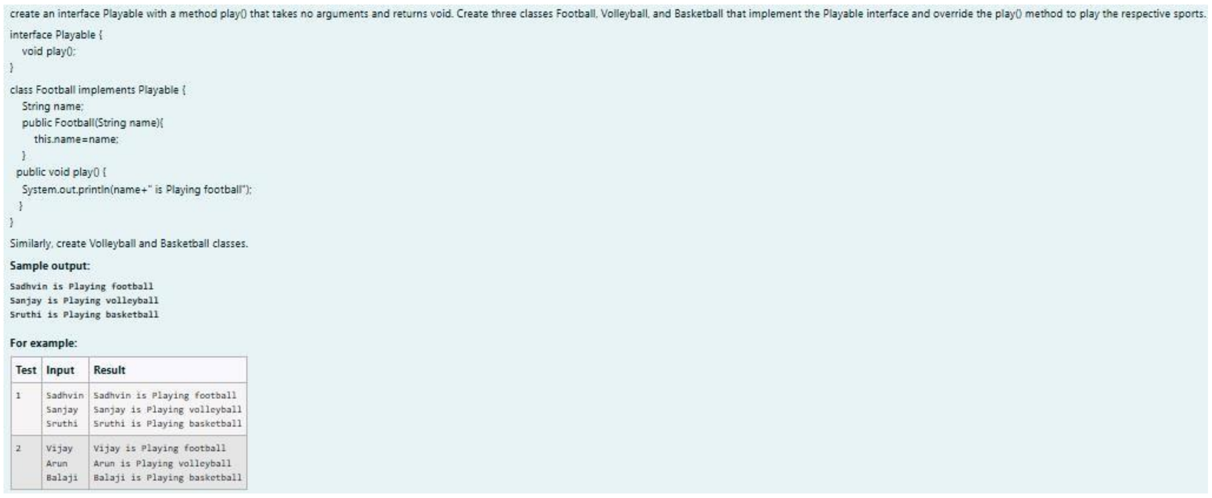
OUTPUT :



	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 :

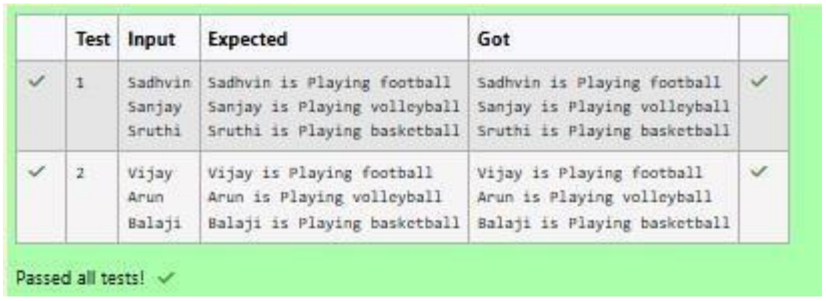
```
import java.util.Scanner;

// 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 :

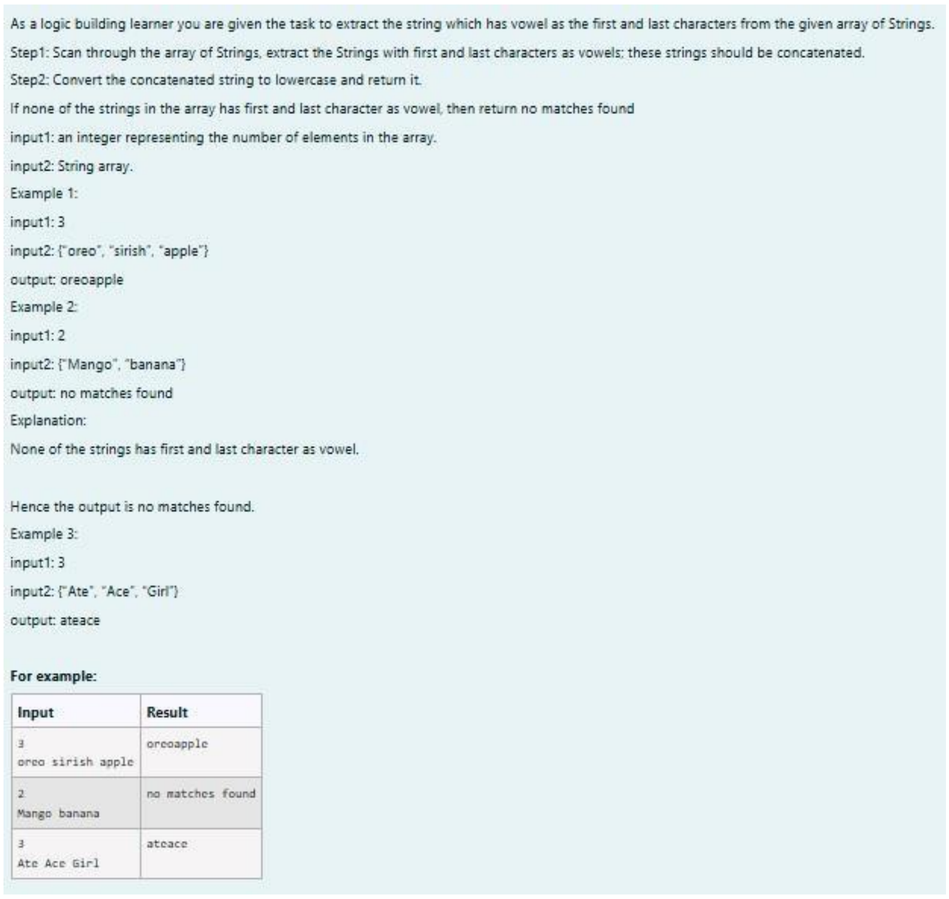


	Test	Input	Expected	Got	
✓	1	Sadhvin Sanjay Sruthi	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	Sadhvin is Playing football Sanjay is Playing volleyball Sruthi is Playing basketball	✓
✓	2	Vijay Arun Balaji	Vijay is Playing football Arun is Playing volleyball Balaji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

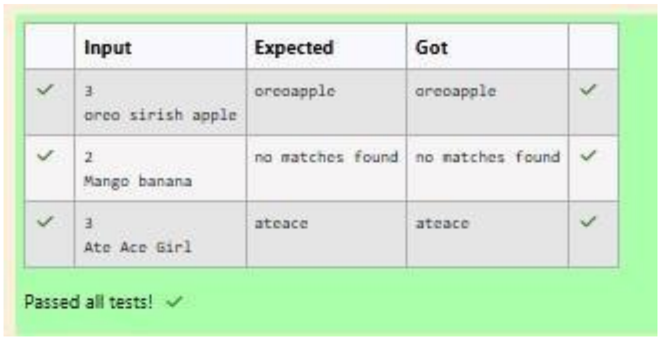
For example:

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 }
}
```

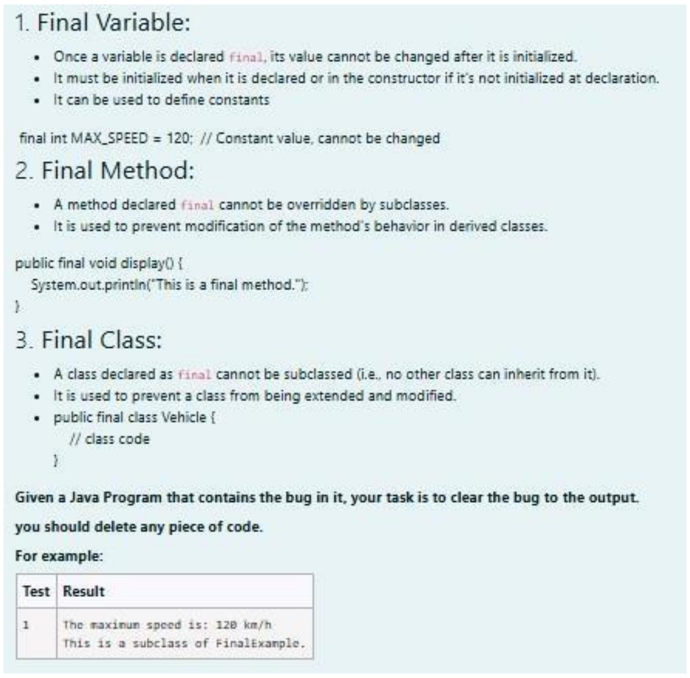
OUTPUT :



	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.

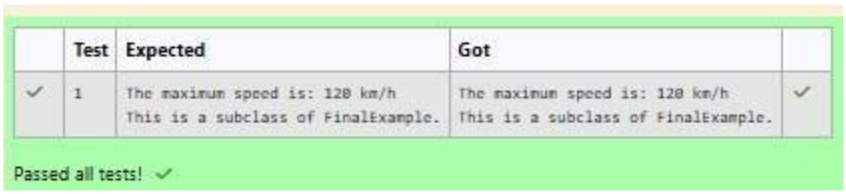
For example:

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.");
    }
}
```

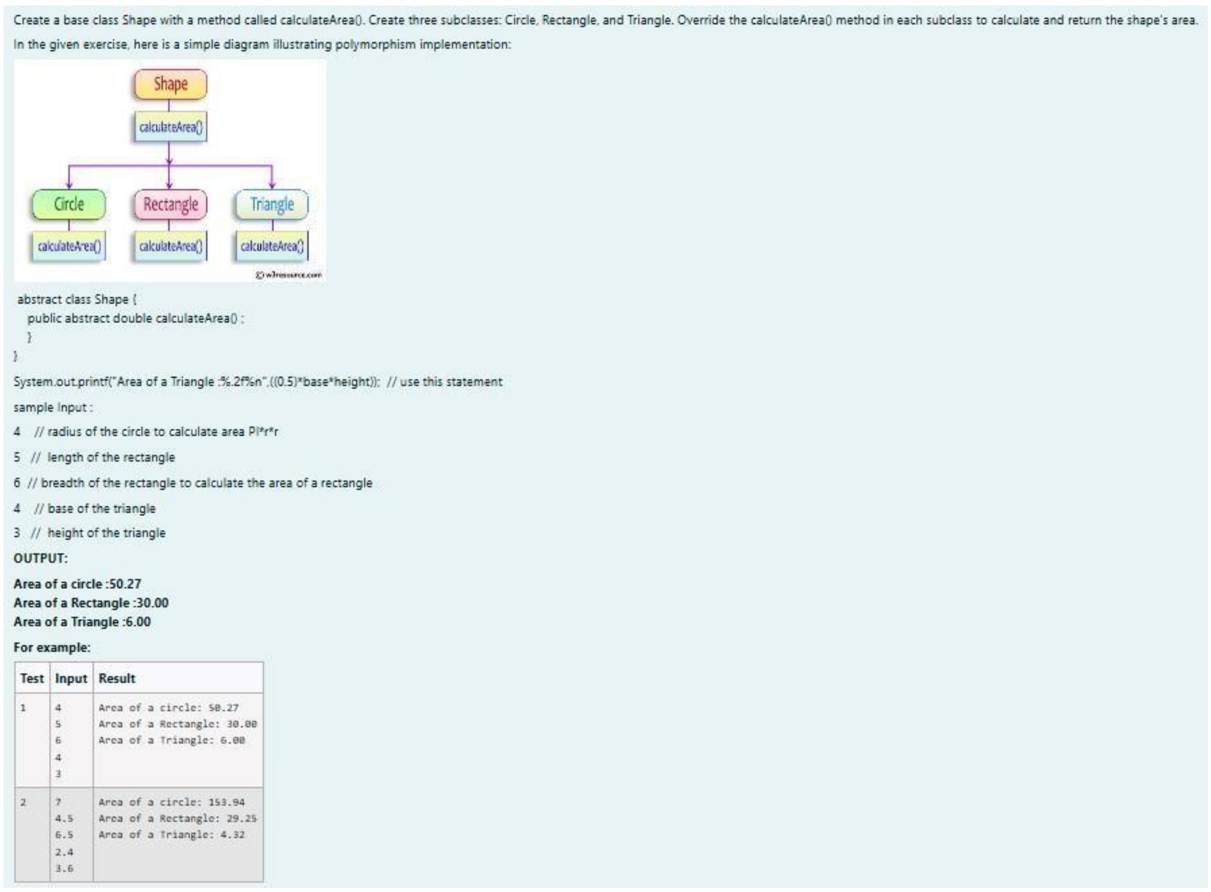
OUTPUT :



	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.	✓

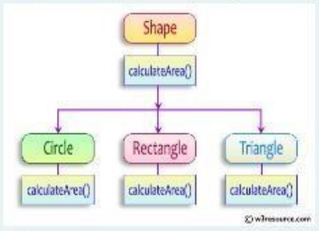
Passed all tests! ✓

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:

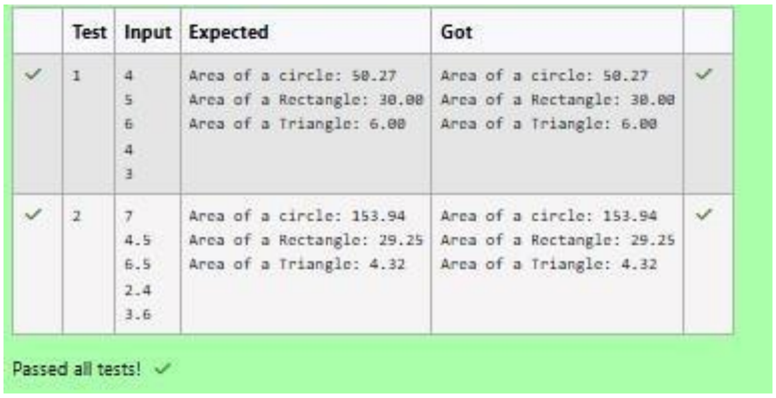
Test	Input	Result
1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00
2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32

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 :

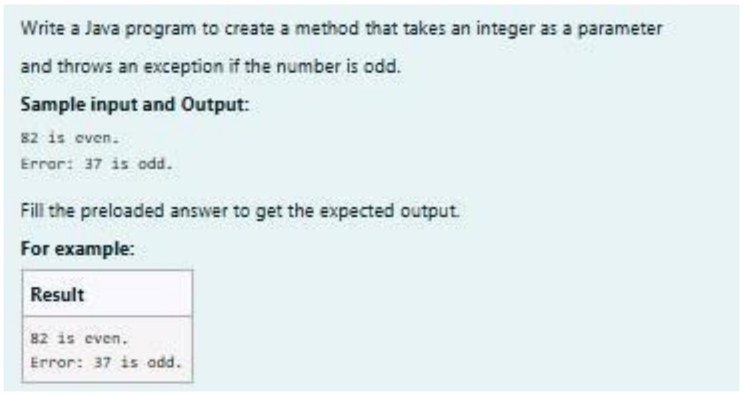


	Test	Input	Expected	Got	
✓	1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	✓
✓	2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	Area of a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

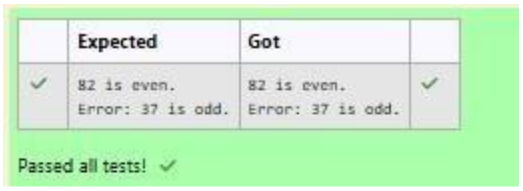
Result
82 is even. Error: 37 is odd.

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 }
    }
}
```

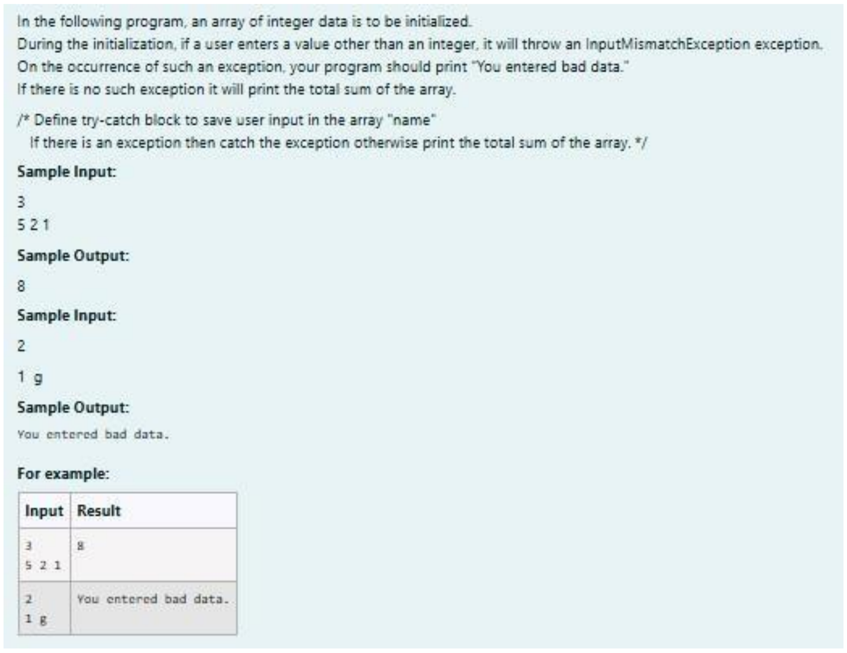
OUTPUT :



	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.
```

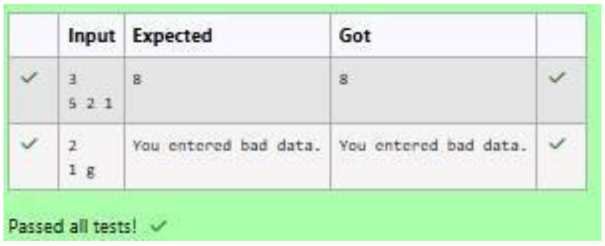
For example:

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 }
}
```

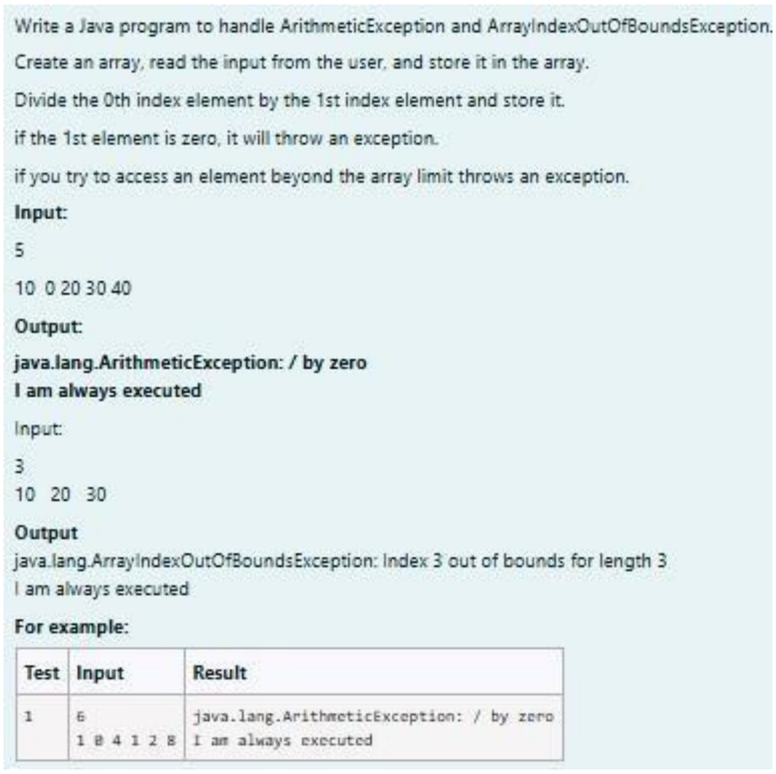
OUTPUT :



	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

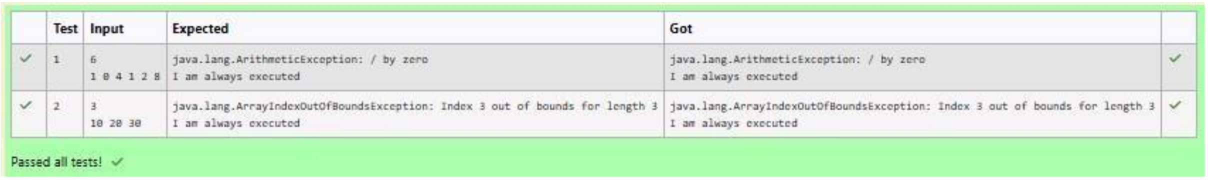
For example:

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 :

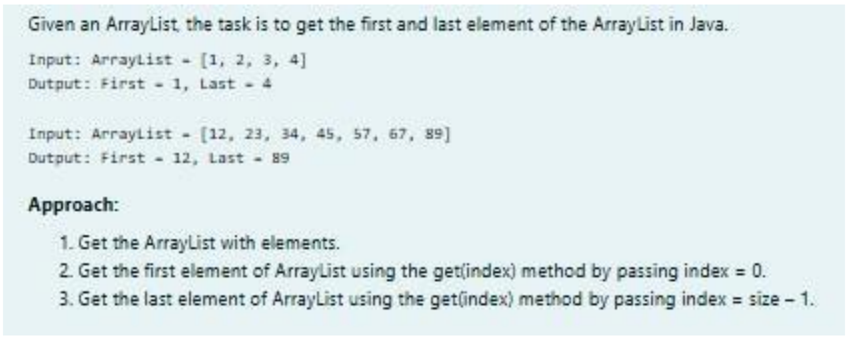


	Test	Input	Expected	Got	
✓	1	6 1 0 4 1 2 8	java.lang.ArithmeticException: / by zero I am always executed	java.lang.ArithmeticException: / by zero I am always executed	✓
✓	2	3 10 20 30	java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3 I am always executed	java.lang.ArrayIndexOutOfBoundsException: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

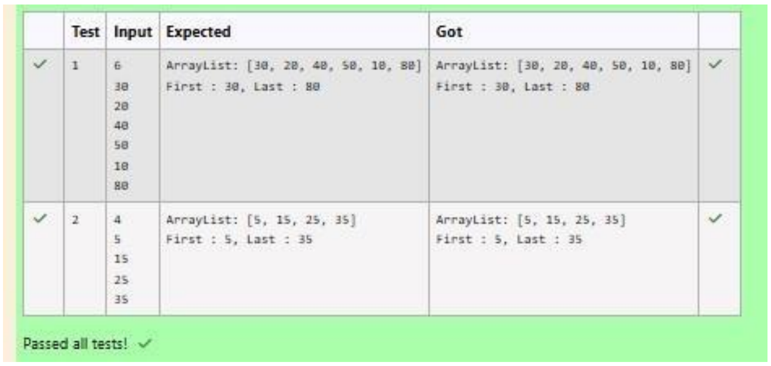
Approach:

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);
    }
}
```

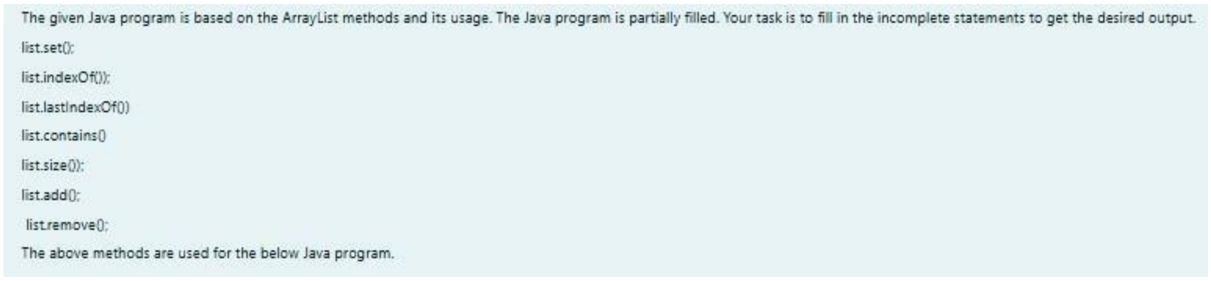
OUTPUT :



	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();

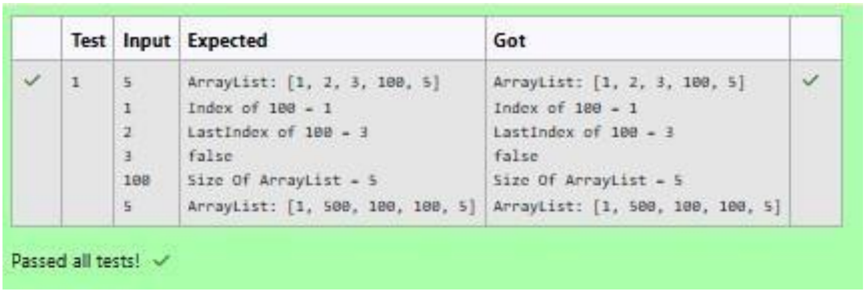
list.remove();

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); }
}
```

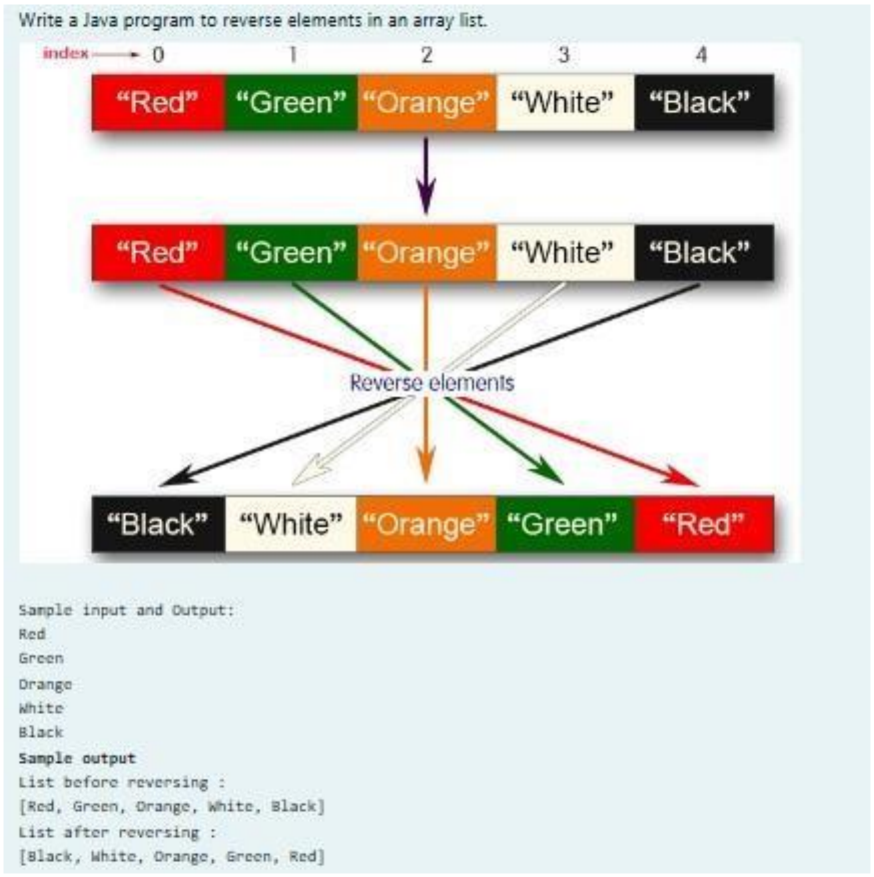
OUTPUT :



	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.

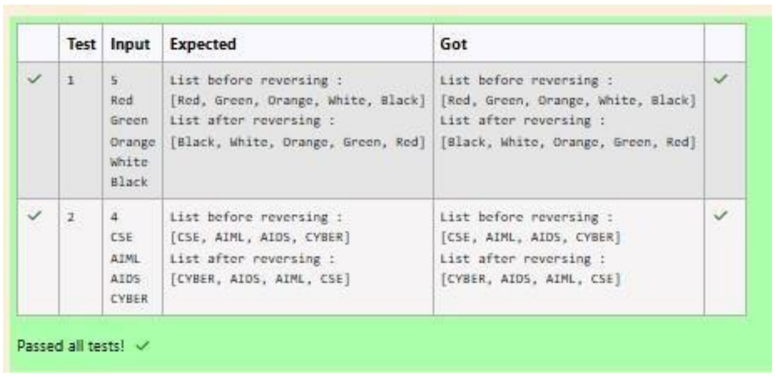
```
Sample input and Output:
Red
Green
Orange
White
Black
Sample output
List before reversing :
[Red, Green, Orange, White, Black]
List after reversing :
[Black, White, Orange, Green, Red]
```

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 :

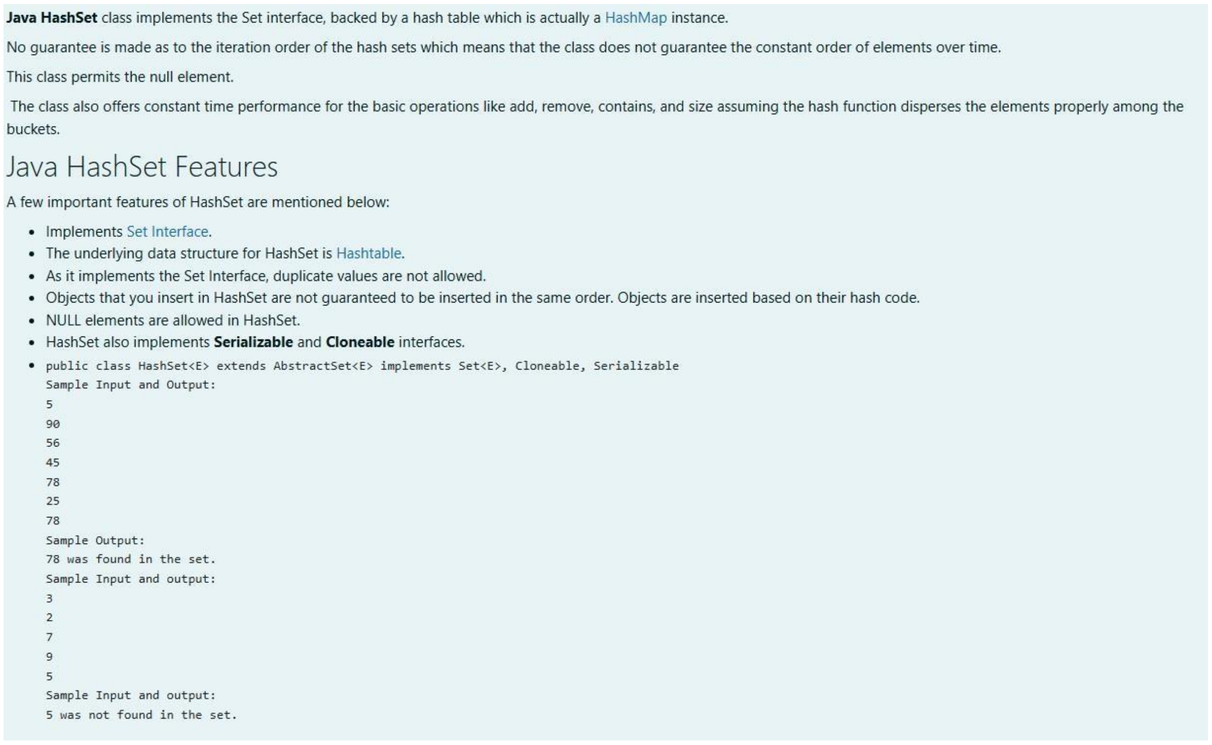


	Test	Input	Expected	Got	
✓	1	5 Red Green Orange White Black	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	List before reversing : [Red, Green, Orange, White, Black] List after reversing : [Black, White, Orange, Green, Red]	✓
✓	2	4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

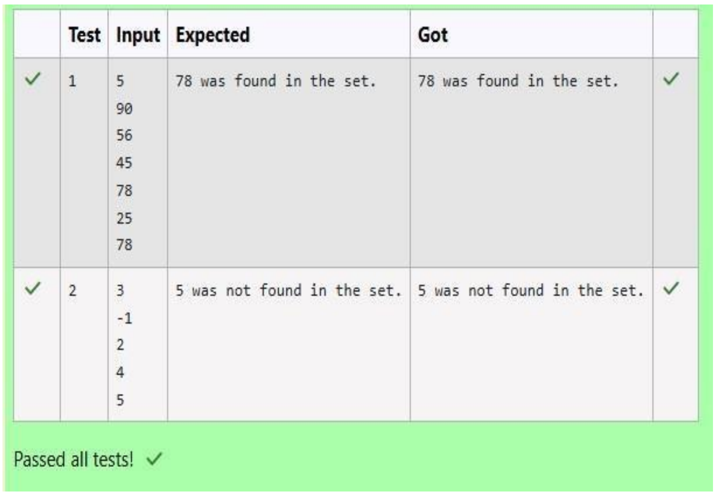
- Implements Set Interface.
- The underlying data structure for HashSet is Hashtable.
- As it implements the Set Interface, duplicate values are not allowed.
- Objects that you insert in HashSet are not guaranteed to be inserted in the same order. Objects are inserted based on their hash code.
- NULL 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();
    }
}
```

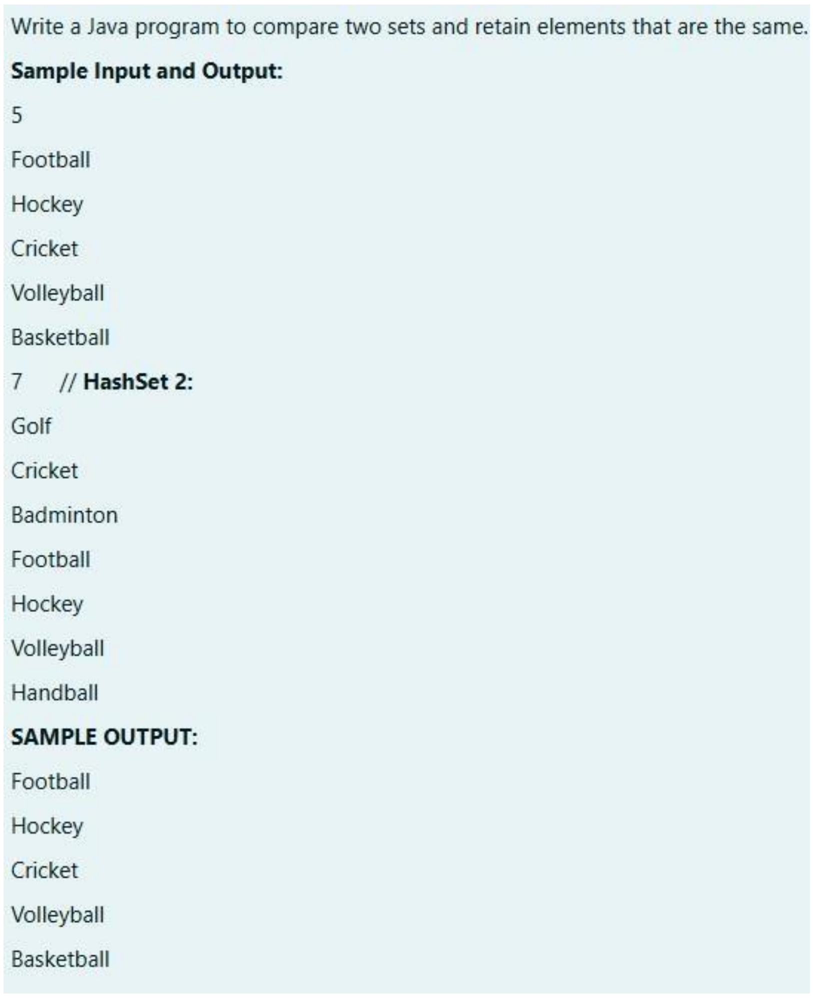
OUTPUT :



	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:

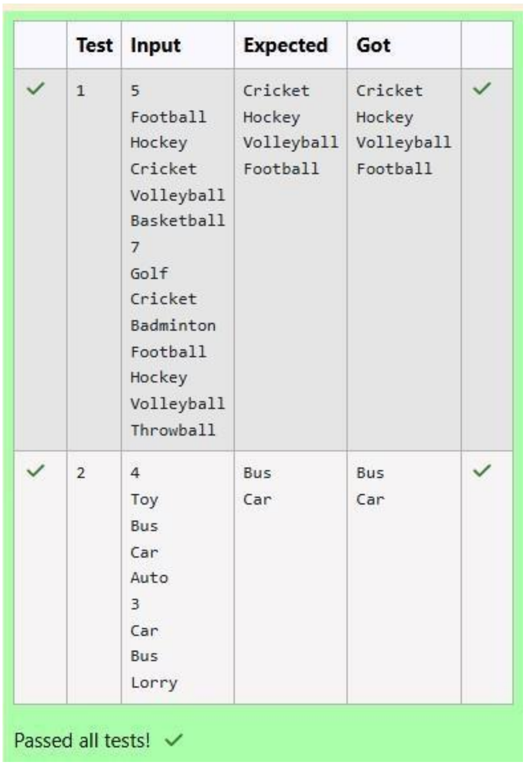
```
Football
Hockey
Cricket
Volleyball
Basketball
```

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();
    }
}
```

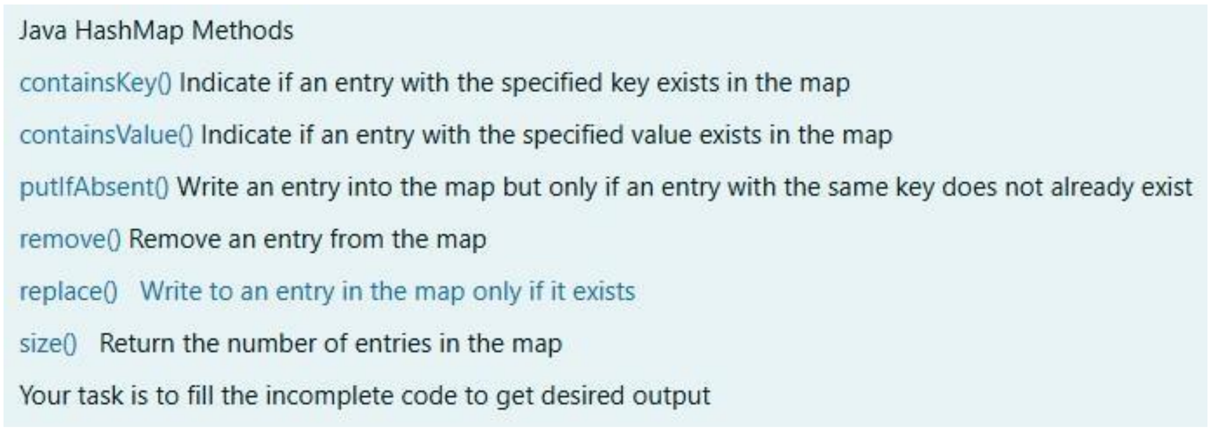
OUTPUT :



	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

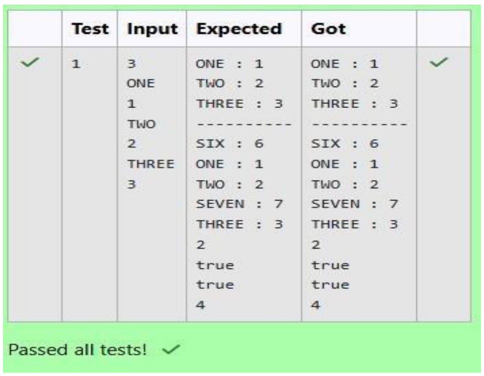
SOLUTION :

```
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 :

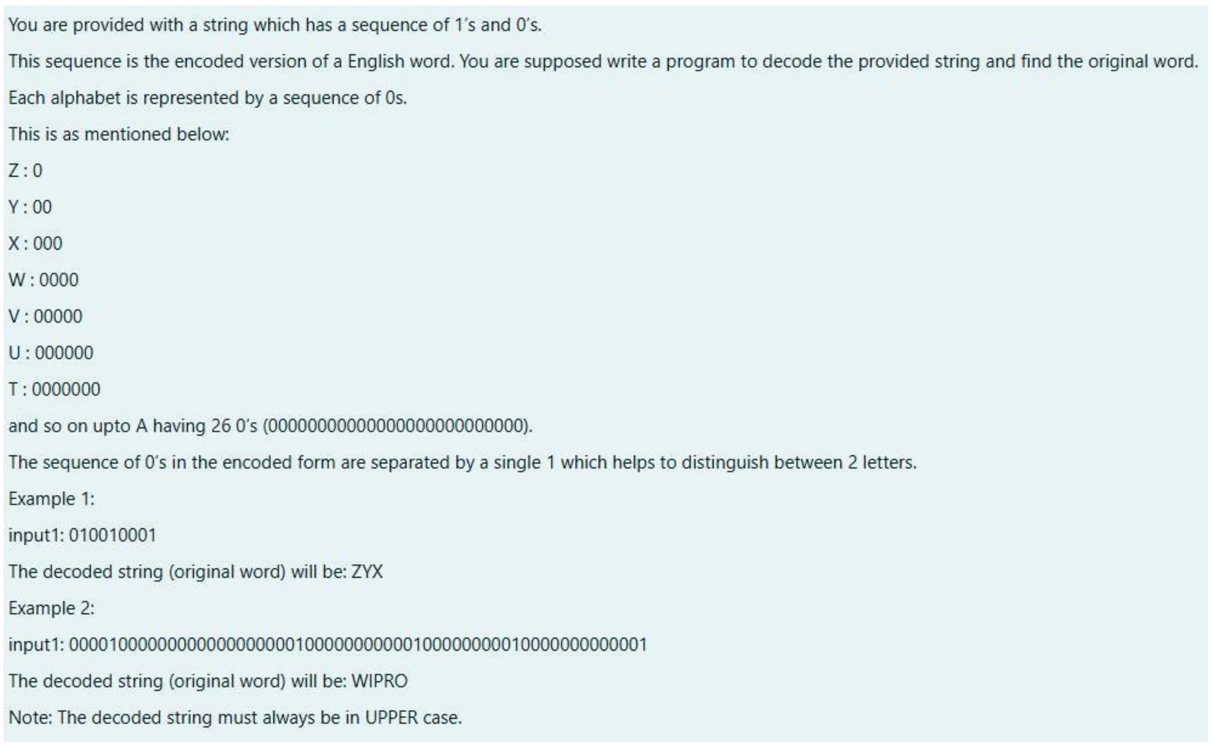


	Test	Input	Expected	Got	
✓	1	3 ONE 1 TWO 2 THREE 3	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	✓

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: 00001000000000000000000100000000000100000000010000000000001

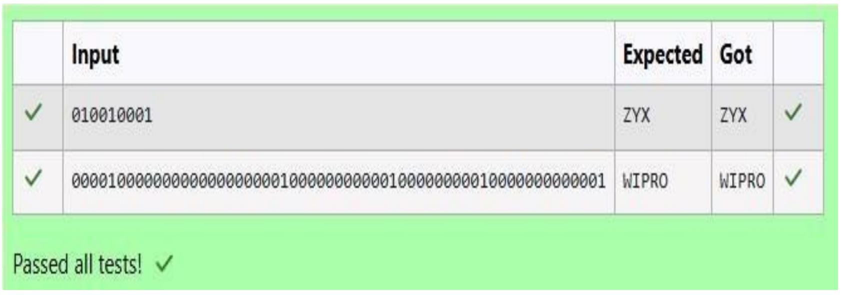
The decoded string (original word) will be: WIPRO

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());
   }

}
```

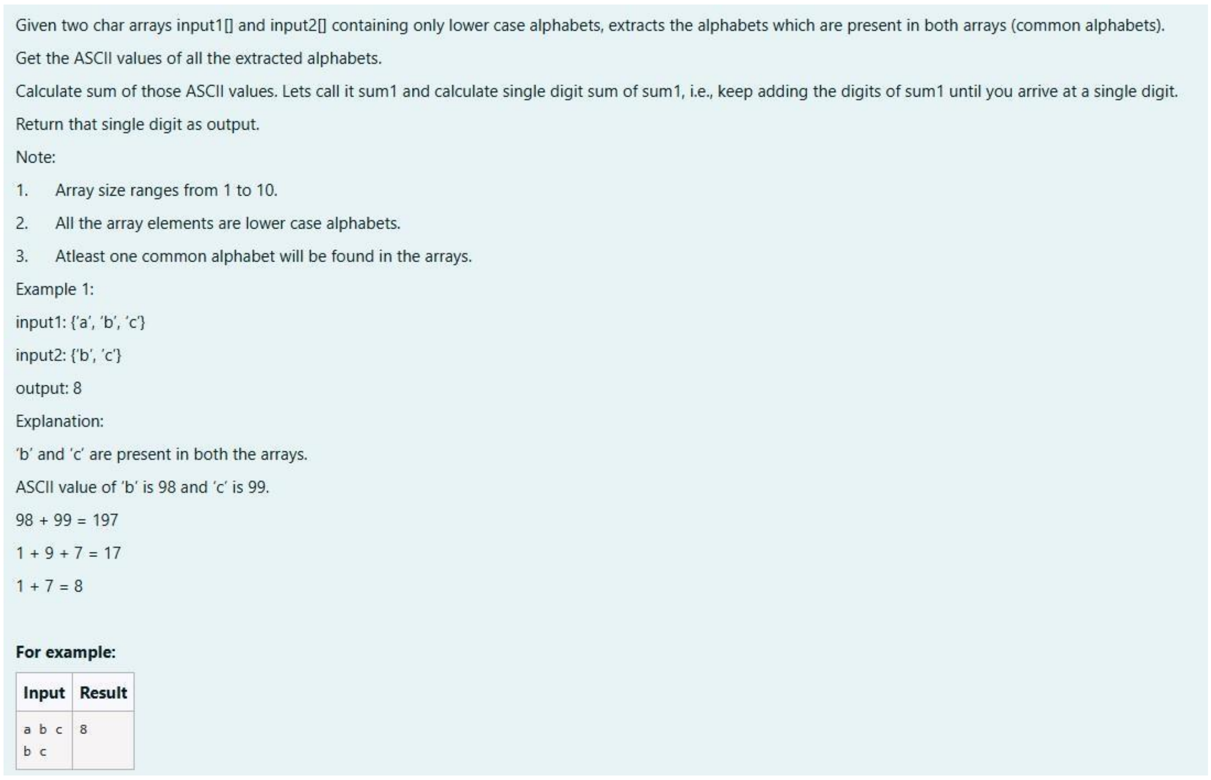
OUTPUT :



	Input	Expected	Got	
✓	010010001	ZYX	ZYX	✓
✓	00001000000000000000000100000000000100000000010000000000001	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

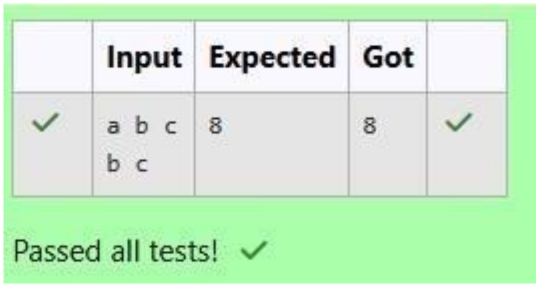
For example:

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); }
}
```

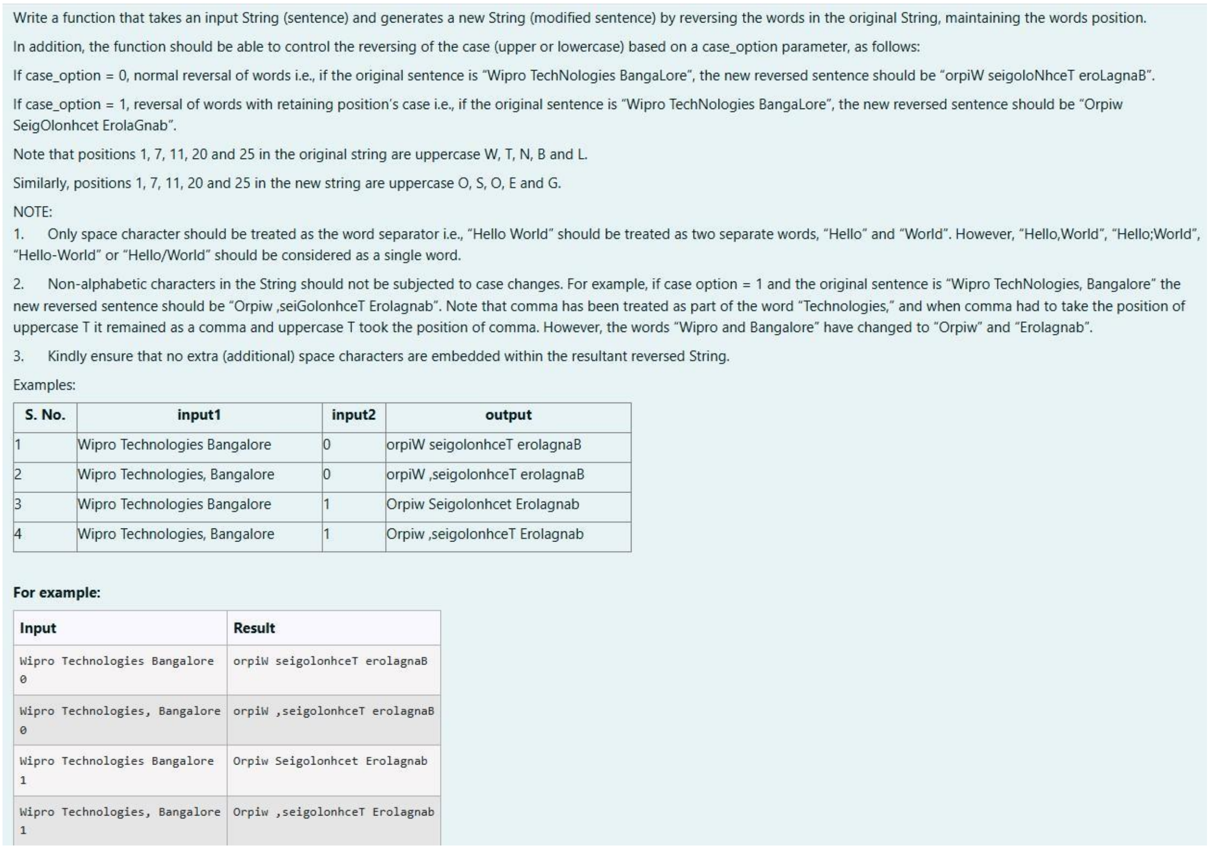
OUTPUT :



	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`

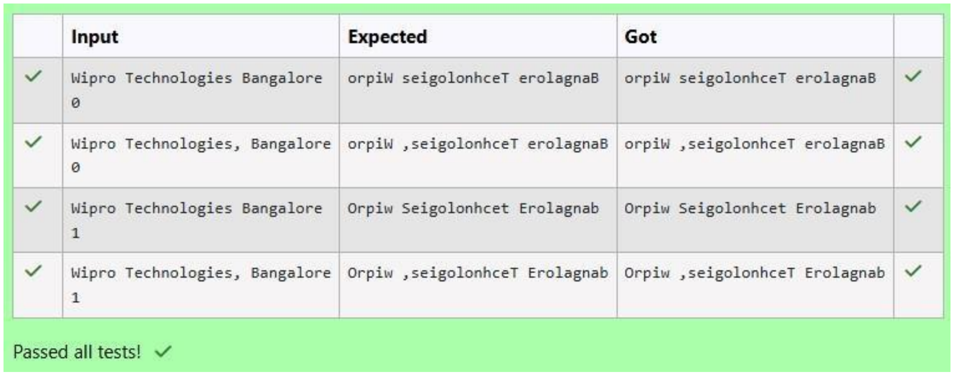
`Wipro Technologies, Bangalore`
`1`
	`Orpiw ,seigolonhceT Erolagnab`

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! ✓

LIBRARY MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

SRUTHI SK 231001216

VAISHALI S 231001235

SOWJANYA D P 231001207

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

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

2024 - 2025

RAJALAKSHMI ENGINEERING COLLEGE
CHENNAI – 602105

BONAFIDE CERTIFICATE

Certified that this project report “**LIBRARY MANAGEMENT SYSTEM**” is the bonafide work of **“SRUTHI SK(231001216),SOWJANYA D P** (231001207) **AND VAISHALI S(231001235)”** who carried out the project work under my supervision.

Submitted for the Practical Examination held on ______________________

HEAD/IT	SUPERVISOR
Dr.P.Valarmathie	Mr.K E Narayana,
Professor and Head,	Assistant Professor,
Information Technology,	**Information Technology**
Rajalakshmi Engineering College	**Rajalakshmi Engineering College**
Thandalam, Chennai - 602 105	**Thandalam, Chennai - 602 105**

ABSTRACT

The Library Management System (LMS) is an efficient tool designed to simplify the management of library resources for both users and administrators. The LMS allows users to view, borrow, and return books, while providing administrators with a special interface to manage the library catalog. Key features include user login, real-time updates on book availability, tracking borrowed books, and administrative functions for adding and managing books. The system is supported by a robust relational database, ensuring efficient data storage, retrieval, and management. SQL is used for all database operations to maintain data integrity and consistency. The LMS offers a straightforward and user-friendly interface developed with Java Swing/AWT, making it easy for everyone to use.

TABLE OF CONTENTS

INTRODUCTION

1.1) INTRODUCTION

This project is a comprehensive desktop application developed for library management, designed to streamline the process of borrowing and returning books. Built using Java with Swing/AWT for the graphical interface and JDBC for database connectivity, the application ensures efficient management of library resources. The backend utilizes a robust MySQL database to handle data storage and retrieval, with SQL ensuring data integrity and consistency. Users can log in, browse the available books, and make reservations. Administrators have special access to add new books and manage the library catalog. The application features real-time updates on book availability, user-friendly interfaces, and confirmation notifications for successful transactions, providing an efficient and seamless user experience.

1.2) OBJECTIVES

The objective of this project is to develop a desktop-based Library Management System tailored for efficient management of library resources and enhanced user experience. Using Java with Swing/AWT, the system aims to provide a seamless process for users to borrow and return books, submitting essential details such as their name and contact information. Administrative functionalities are integrated to allow for effective oversight and management of book data. Leveraging MySQL as the backend database ensures robust data storage and scalability. The application's user interface prioritizes intuitive navigation and interaction, while a confirmation mechanism provides users with immediate feedback on the successful completion of their transactions. Ultimately, the project endeavors to modernize and optimize library management, fostering efficiency, transparency, and user satisfaction within the library system.

1.3) MODULES

User Authentication Module:

Manages user login and authentication. This module includes the setup for login screens, user validation, and session management to ensure secure access for users and administrators.

Database Connection Module:

Establishes a connection to the MySQL database using JDBC. This module handles database configuration, connection pooling, and error handling to ensure reliable database operations.

Book Schema Module:

Defines the structure for storing book data in the MySQL database. This module includes fields like book title, author, ISBN, availability status, and other relevant details.

User Interface Module:

Incorporates Java Swing/AWT components for rendering the graphical user interface of the application. This module provides the frontend components for user interactions, including login forms, book listing, borrowing, and returning books.

Book Management Module:

Manages the logic for adding, updating, and deleting book records in the database. This module includes the administrative functionalities for maintaining the library catalog, ensuring up-to-date book information.

Admin Interface Module:

Provides the administrative interface for managing library operations. This module includes screens and functionalities for administrators to oversee user accounts, manage books, and view borrowing activities.

1.4) SOFTWARE REQUIREMENT

The Library Management System (LMS) is a desktop application designed to streamline and enhance the management of library resources. Users can access the system to borrow and return books, providing details such as their name and contact information. Administrators have access to additional functionalities, including the ability to add, edit, and manage book data through an administrative interface. The system prioritizes user experience by offering intuitive forms, responsive design, and informative feedback messages. Security measures such as user authentication and data validation are implemented to ensure the confidentiality and integrity of user data. Built using Java with Swing/AWT for the interface and JDBC for database connectivity, with MySQL as the backend, the LMS aims to provide a reliable, scalable, and user-friendly solution for efficient library management.

SURVEY OF TECHNOLOGIES

2.1) SOFTWARE DESCRIPTION

The Library Management System (LMS) is a comprehensive desktop application meticulously crafted to streamline and enhance the management of library resources. Tailored for libraries within educational institutions or organizations, the LMS employs Java with Swing/AWT for its graphical user interface, ensuring a smooth and intuitive user experience. JDBC serves as the bridge for seamless connectivity to the database, facilitating efficient data management and retrieval. the LMS incorporates user authentication and authorization features to ensure secure access and protect sensitive library data from unauthorized access or manipulation.

2.2) LANGUAGES

- Java
- SQL (Structured Query Language)
- JDBC
- Java Swing/AWT

REQUIREMENT AND ANALYSIS

3.1) REQUIREMENT SPECIFICATION

The Library Management System (LMS) is an essential tool tailored for educational institutions and organizations to efficiently manage their library resources. The system encompasses a variety of functionalities aimed at providing a seamless experience for both users and administrators. The LMS shall allow users to register by providing necessary details including name, email, contact number, and password. Registered users shall have the ability to securely log in to their accounts to access library services and resources. The system shall facilitate administrators to add, edit, and delete book records, including details such as title, author, genre, ISBN, availability status, and location within the library. The LMS shall support the borrowing and returning process, allowing users to borrow books for a specified duration and return them within the designated timeframe. Users shall be notified of upcoming due dates and overdue books to ensure timely returns and prevent fines or penalties. The LMS shall implement robust security measures to protect user data and ensure confidentiality. Administrators shall have access to a comprehensive dashboard for managing user accounts, monitoring borrowing activities, and overseeing library operations. The dashboard shall provide functionalities for generating reports, tracking inventory, and analyzing library usage statistics. The LMS shall be scalable to accommodate growing library collections and increasing user demand over time.

3.2) HARDWARE AND SOFTWARE REQUIREMENT

Hardware Requirement

- Processor: Intel Core i5 or higher
- RAM: 8 GB or more
- Storage: 500 GB SSD
- Network: High-speed internet

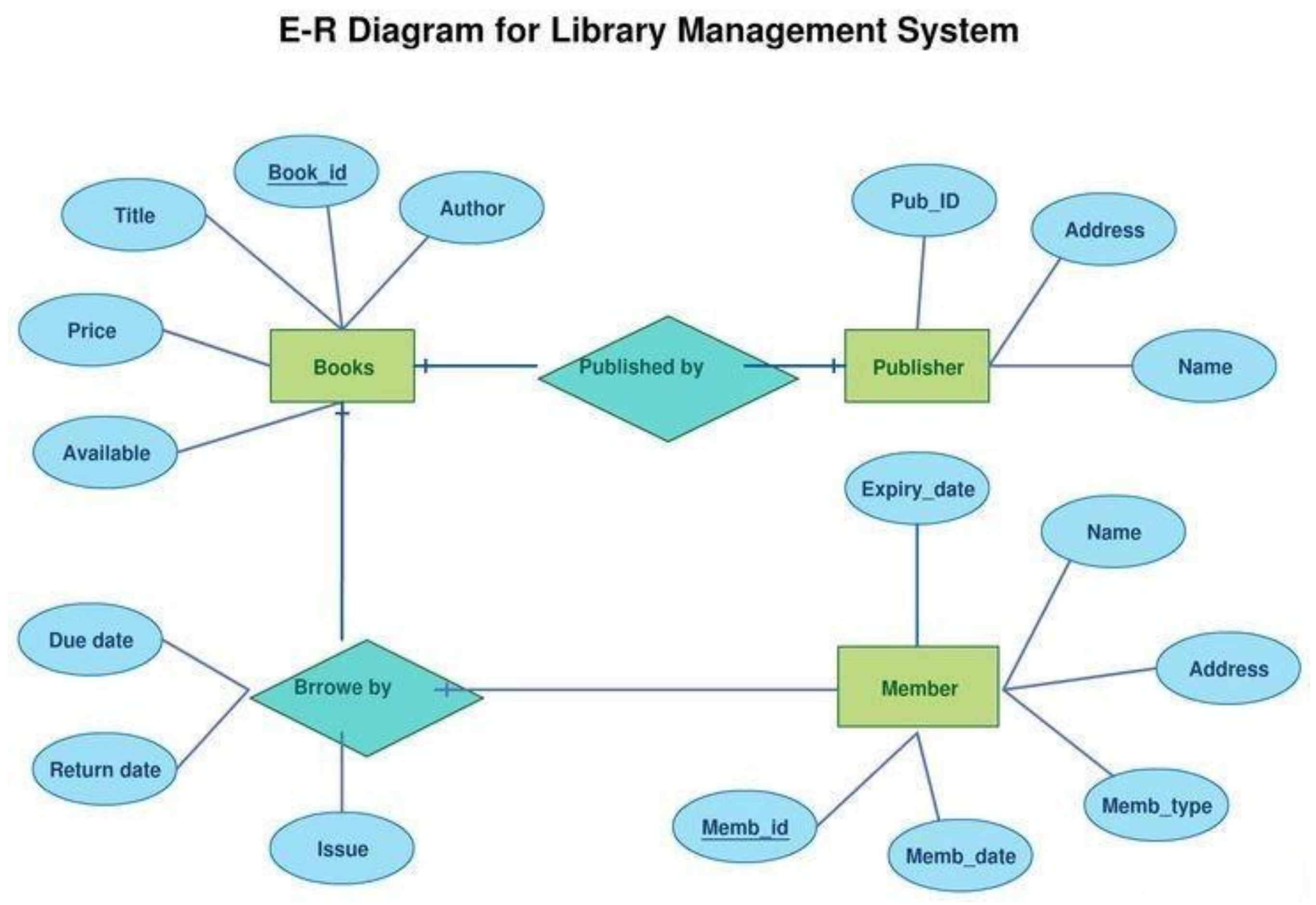
Software Requirement

- Operating System: Windows 10, macOS Catalina, or Ubuntu 20.04 LTS
- Java Development Kit (JDK): JDK 8 or later
- Database: MySQL 8.0, PostgreSQL 13.0, or similar relational database management system (RDBMS)
- JDBC Driver**:** MySQL Connector/J, PostgreSQL JDBC Driver, or equivalent for database connectivity

Development Tools

- IDE: Visual Studio Code, Eclipse, IntelliJ IDEA or similar
- Version Control: Git (GitHub or GitLab)
- Frameworks/Libraries: React.js, Angular.js (front-end); Django, Express.js, Laravel (back-end)
- Other Tools: Postman (API testing), Docker (containerization), Selenium (testing)

3.3)
ER DIAGRAM



3.4)

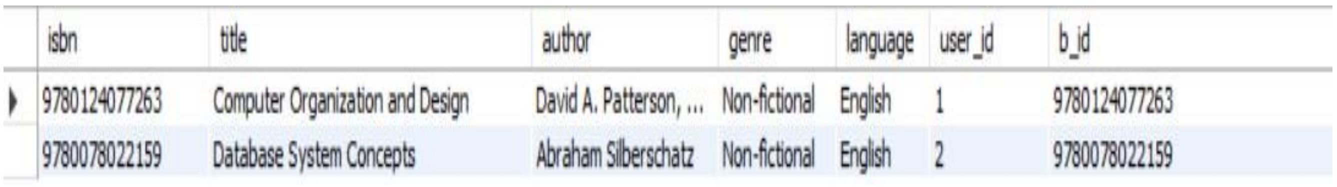
NORMALIZATION

Normalization is a critical aspect of database design, essential for optimizing data organization, reducing redundancy, and ensuring data integrity. Here's how the normalization process would look for the Library Management System.

1. First Normal Form (1NF)

Ensure atomicity by making sure that each table has a primary key, and each column contains only atomic values.

Initial Unnormalized Table:

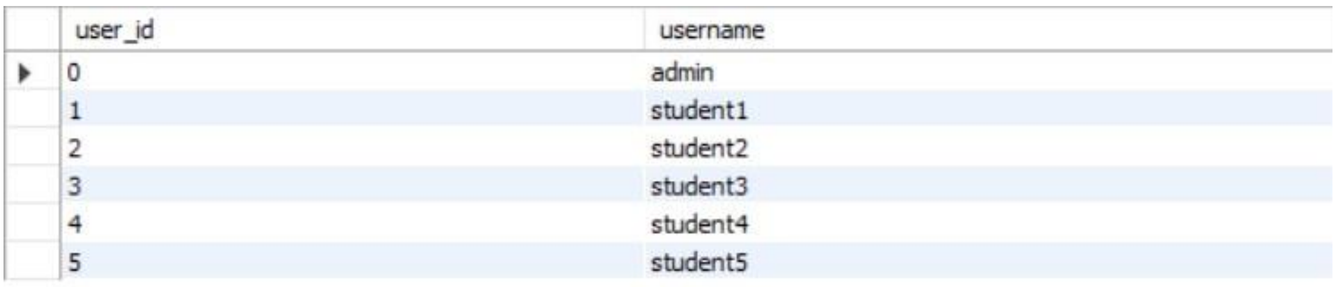


	isbn	title	author	genre	language	user_id	b_id
▶	9780124077263	Computer Organization and Design	David A. Patterson, ...	Non-fictional	English	1	9780124077263
	9780078022159	Database System Concepts	Abraham Silberschatz	Non-fictional	English	2	9780078022159

First Normal Form (1NF)

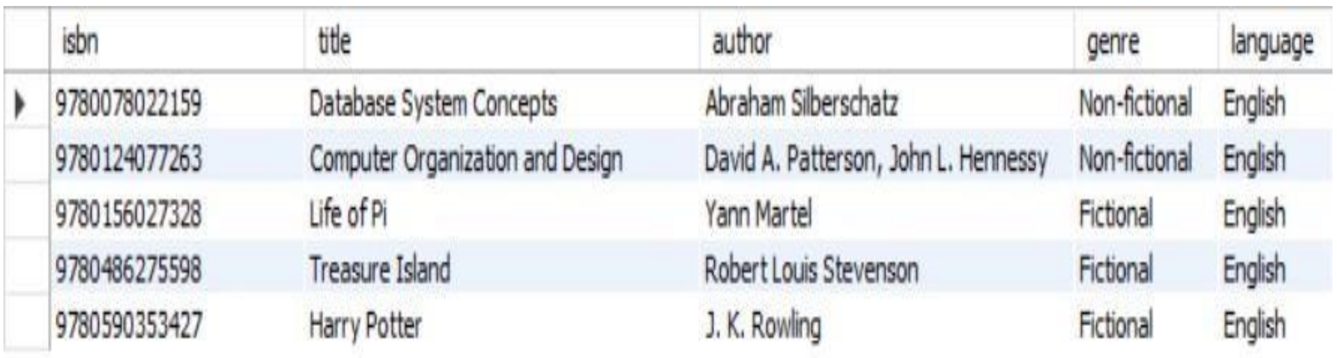
Split into multiple tables to ensure atomicity:

User Table:



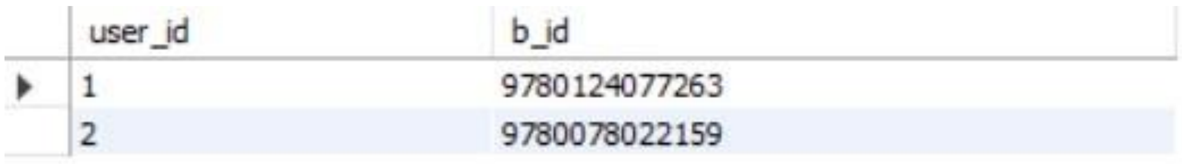
	user_id	username
▶	0	admin
	1	student1
	2	student2
	3	student3
	4	student4
	5	student5

Book Table:



isbn	title	author	genre	language
9780078022159	Database System Concepts	Abraham Silberschatz	Non-fictional	English
9780124077263	Computer Organization and Design	David A. Patterson, John L. Hennessy	Non-fictional	English
9780156027328	Life of Pi	Yann Martel	Fictional	English
9780486275598	Treasure Island	Robert Louis Stevenson	Fictional	English
9780590353427	Harry Potter	J. K. Rowling	Fictional	English

Issued books Table:



user_id	b_id
1	9780124077263
2	9780078022159

Second Normal Form (2NF)

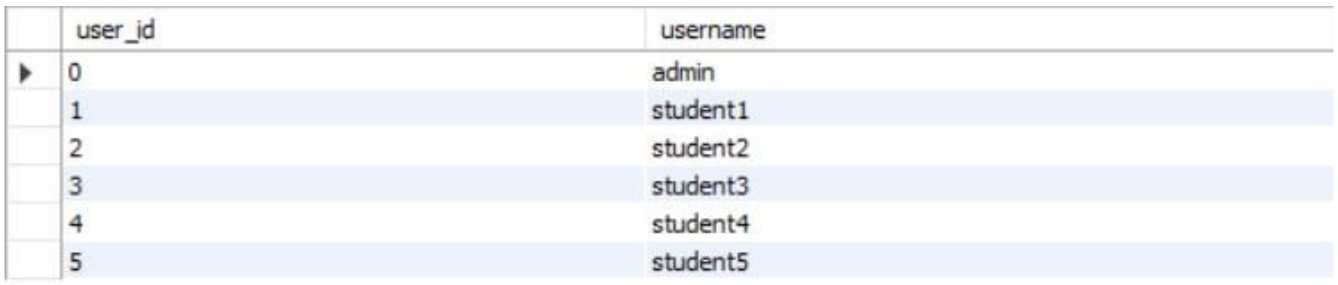
Remove partial dependencies. No partial dependencies exist, so no changes needed.

Third Normal Form (3NF)

Remove transitive dependencies. No transitive dependencies exist, so no changes needed.

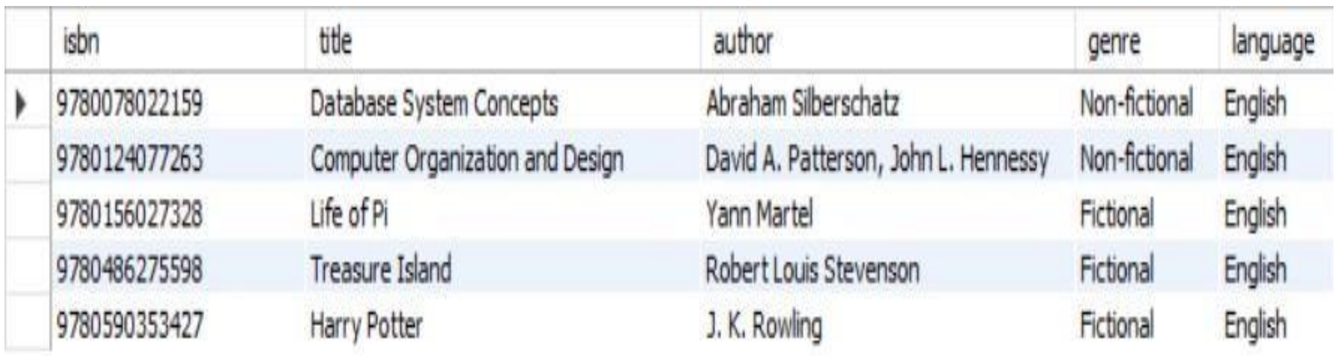
Final Database Schema User

Table:



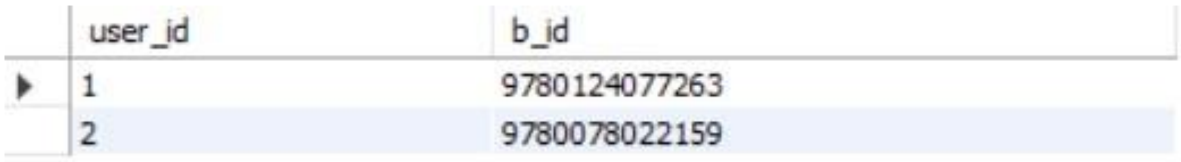
user_id	username
0	admin
1	student1
2	student2
3	student3
4	student4
5	student5

Book Table:



isbn	title	author	genre	language
9780078022159	Database System Concepts	Abraham Silberschatz	Non-fictional	English
9780124077263	Computer Organization and Design	David A. Patterson, John L. Hennessy	Non-fictional	English
9780156027328	Life of Pi	Yann Martel	Fictional	English
9780486275598	Treasure Island	Robert Louis Stevenson	Fictional	English
9780590353427	Harry Potter	J. K. Rowling	Fictional	English

Issued books Table:



user_id	b_id
1	9780124077263
2	9780078022159

This normalized schema ensures data integrity, reduces redundancy, and improves efficiency in the Library Management System.

4) PROGRAM CODE

App.java

```
import java.sql.*; public
class App {
   public static void main(String[] args) throws Exception {
Connection connection = Connections.getConnection();         if(
connection != null) {
            System.out.println("Connected to the database");
new Login();
        } else {
            System.out.println("Failed to connect to the database");
       }
   }
}
```

Connections.java import

```
java.sql.DriverManager; import
java.sql.SQLException; public
class Connections {
   public static java.sql.Connection getConnection() {
       String url = "jdbc:mysql://localhost:3306/library_management_system";
       String user = "root";
String password = "root";
```

```
        try {
            return DriverManager.getConnection(url, user, password);
        }catch (SQLException e) {
           e.printStackTrace();
        }
        return null;
    }
}
```

Login.java import javax.swing.*;

```
import java.awt.*; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener;

public class Login extends JFrame implements ActionListener {
    JTextField username = new JTextField();
    JPasswordField password = new JPasswordField();
    JButton login = new JButton("LOGIN");
JButton reset = new JButton("RESET");     public
Login() {
        // Set the font for all components
        Font font = new Font("Tahoma", Font.PLAIN, 14);
        UIManager.put("Label.font", font);
        UIManager.put("TextField.font", font);
       UIManager.put("PasswordField.font", font);
       UIManager.put("Button.font", font);
```

```
        // Set the foreground color for all components to white
       UIManager.put("Label.foreground", Color.WHITE);
       UIManager.put("PasswordField.foreground", Color.WHITE);

        // Label "Library Management System"
        JLabel label = new JLabel("Library Management System");
        label.setBounds((500 - label.getPreferredSize().width) / 2-30, 30, 300, 25);
this.add(label);

        // Label "Welcome to the Library"
        JLabel label2 = new JLabel("Welcome to the Library");
        label2.setBounds((500 - label.getPreferredSize().width) / 2-15, 70, 200, 25);
this.add(label2);

        // Label "Login to continue"
        JLabel label3 = new JLabel("Login to continue");
        label3.setBounds((500 - label.getPreferredSize().width) / 2-5, 110, 200, 25);
this.add(label3);

        JLabel userLabel = new JLabel("Username:");
userLabel.setBounds(50, 160, 100, 25);

        JLabel passwordLabel = new JLabel("Password:");
       passwordLabel.setBounds(50, 200, 100, 25);

       username.setBounds(150, 160, 200, 25);
```

```
        password.setBounds(150, 200, 200, 25);


       login.addActionListener(this);
        login.setBounds(100, 250, 100, 30);
login.setFocusable(false);


        reset.addActionListener(this);
reset.setBounds(220, 250, 100, 30);
reset.setFocusable(false);


        // login frame ->
        this.setTitle("Library Management System");
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLayout(null);
this.setSize(450, 350);
this.setVisible(true);
        this.getContentPane().setBackground(Color.BLACK);
this.setLocationRelativeTo(null); // Center the frame on the screen
this.add(userLabel);        this.add(passwordLabel);
this.add(username);        this.add(password);        this.add(login);
       this.add(reset);
   }
    public void actionPerformed(ActionEvent e) {
       if(e.getSource() == login) {
           LoginInfo login1 = new LoginInfo( username.getText(), new
String(password.getPassword()));             int
id = login1.connectToDatabase();             if(
id != -1){
            System.out.println("Login successful");
            JOptionPane.showMessageDialog(null, "Login successful");
            this.dispose();
new Library( id );
         } else {
            JOptionPane.showMessageDialog(null, "Login failed");
         }
      } else if(e.getSource() == reset) {
username.setText("");            password.setText("");
      }
   }
}
```

LoginInfo.java import

```
java.sql.Connection; import
java.sql.ResultSet; import
java.sql.SQLException; import
java.sql.Statement;

public class LoginInfo {
   String username;
  String password;    public LoginInfo
connectToDatabase;
```

```
   public LoginInfo(String username, String password) {
this.username = username;        this.password =
password;
   }
   public int connectToDatabase() {
      Connection connection = Connections.getConnection();
if( connection != null) {
         System.out.println("Connected to the database");
         try {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM users WHERE
username = '"+username+"' AND password = '"+password+"'");
            if(resultSet.next()) {
return resultSet.getInt(1);
            }
return -1;
         } catch (SQLException e) {
            e.printStackTrace();
         }
      } else {
         System.out.println("Failed to connect to the database");
      }
return -1;
   }}
```

Library.java import

```
java.awt.Color; import
java.sql.SQLException;
```

```
import javax.swing.JButton;
import javax.swing.JFrame; import
javax.swing.JLabel;

public class Library extends JFrame {
   int id;     public
Library(int i) {
       id =i;
       // view book
       JLabel view_book = new JLabel("View Books");
view_book.setBounds(170, 50, 100, 25);         this.add(view_book);

       JButton view_book_button = new JButton("View Books");
view_book_button.setBounds(150, 80, 150, 25);
view_book_button.setFocusable(false);         view_book_button.addActionListener(e
-> {
           try {
               new ViewBooks();
 } catch (SQLException e1) {
 e1.printStackTrace();
           }
       });
       this.add(view_book_button);

       // get book
```

```
        JLabel get_book = new JLabel("Get Book");
get_book.setBounds(170, 120, 100, 25);        this.add(get_book);

        JButton get_book_button = new JButton("Get Book");
get_book_button.setBounds(150,       150,       150,       25);
get_book_button.setFocusable(false);
get_book_button.addActionListener(e -> {
            try {
                new GetBook(id);
} catch (SQLException e1) {
e1.printStackTrace();
            }
        });
        this.add(get_book_button);

        JLabel view_borrowed_books = new JLabel("View Borrowed Books");
view_borrowed_books.setBounds(170, 190, 150, 25);        this.add(view_borrowed_books);

       JButton view_borrowed_books_button = new JButton("Borrowed Books");
view_borrowed_books_button.setBounds(150, 220, 150, 25);
view_borrowed_books_button.setFocusable(false);
view_borrowed_books_button.addActionListener(e -> {
            try {
                new ViewBorrowedBooks(id);
} catch (SQLException e1) {
e1.printStackTrace();
```

```
            }
        });
        this.add(view_borrowed_books_button);

        JLabel return_book = new JLabel("Return Book");
return_book.setBounds(170, 260, 100, 25);        this.add(return_book);

        JButton return_book_button = new JButton("Return Book");
return_book_button.setBounds(150, 290, 150, 25);
return_book_button.setFocusable(false);        return_book_button.addActionListener(e
-> {
            try {
                new ReturnBook(id);
} catch (SQLException e1) {
e1.printStackTrace();
            }
        });

        this.add(return_book_button);

        JLabel add_book = new JLabel("Add Book"); add_book.setBounds(170,
        320, 100, 25);

        JButton add_book_button = new JButton("Add Book");
add_book_button.setBounds(150,350,150,25);
add_book_button.setFocusable(false);
add_book_button.addActionListener(e -> {            new
Admin();
       });        if( id ==0){
this.add(add_book);
this.add(add_book_button);
       }
       this.setTitle("Library Management System");
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       this.setLayout(null);
       this.getContentPane().setBackground(Color.BLACK);
this.setSize(600, 550);        this.setVisible(true);
this.setResizable(false);
this.setLocationRelativeTo(null);
   }
}
```

Admin.java

```
import java.awt.Color; import
java.awt.GridLayout; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener; import
java.sql.Connection; import
java.sql.PreparedStatement; import
java.sql.ResultSet; import
java.sql.SQLException;
```

```
import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel; import

javax.swing.JOptionPane; import

javax.swing.JTextField; import

javax.swing.UIManager;


public class Admin extends JFrame implements ActionListener{

   public JTextField isbnField, titleField, authorField, languageField, genreField, copiesField;

public JLabel isbnLabel, titleLabel, authorLabel, languageLabel, genreLabel, copiesLabel;

public JButton submitButton;


   public Admin() {


      UIManager.put("Label.foreground", Color.WHITE); isbnLabel

      = new JLabel("ISBN");

      isbnField = new JTextField(); isbnLabel.setBounds(50,

      50, 100, 25); isbnField.setBounds(150, 50, 200, 25);

      this.add(isbnLabel);

this.add(isbnField);


      titleLabel = new JLabel("Title");

titleField = new JTextField();

titleLabel.setBounds(50, 100, 100, 25);

titleField.setBounds(150, 100, 200, 25);

this.add(titleLabel);        this.add(titleField);
```

```
        authorLabel = new JLabel("Author");
authorField = new JTextField();
authorLabel.setBounds(50, 150, 100, 25);
authorField.setBounds(150, 150, 200, 25);
this.add(authorLabel);        this.add(authorField);

        languageLabel = new JLabel("Language");
languageField = new JTextField();
languageLabel.setBounds(50, 200, 100, 25);
languageField.setBounds(150, 200, 200, 25);

        genreLabel = new JLabel("Genre");
        genreField = new JTextField(); genreLabel.setBounds(50,
        250, 100, 25); genreField.setBounds(150, 250, 200, 25);

        copiesLabel = new JLabel("Copies");
copiesField = new JTextField();
copiesLabel.setBounds(50, 300, 100, 25);
copiesField.setBounds(150, 300, 200, 25);

        submitButton = new JButton("Submit");
submitButton.addActionListener(this);        submitButton.setBounds(150,
350, 100, 30);
```

```
        this.add(languageLabel);
this.add(languageField);
this.add(genreLabel);        this.add(genreField);
this.add(copiesLabel);
this.add(copiesField);
this.add(submitButton);


        this.setLayout(new GridLayout(7, 2));
this.add(isbnLabel);
        this.add(isbnField);
        this.setTitle("Add Book"); this.getContentPane().setBackground(Color.BLACK);
        this.setLayout(null); this.setSize(650, 450);
        this.setVisible(true);
this.setResizable(false);
this.setLocationRelativeTo(null);
   }
   public void actionPerformed(ActionEvent e) {
if (e.getSource() == submitButton) {
Connection connection = null;
            PreparedStatement statement = null;
ResultSet resultSet = null;
            try {
                connection = Connections.getConnection();
if (connection != null) {
                    String isbn = isbnField.getText();
                    String title = titleField.getText();
                    String author = authorField.getText();

                    String language = languageField.getText();

String genre = genreField.getText();

                    try {

                        Integer.parseInt(copiesField.getText());

                    } catch (NumberFormatException ex) {

                        JOptionPane.showMessageDialog(this, "Copies must be a number");

                        return;

                    }

                    int copies = Integer.parseInt(copiesField.getText());


                    if (isbn.isEmpty() || title.isEmpty() || author.isEmpty() || language.isEmpty() ||
genre.isEmpty() || copiesField.getText().isEmpty()) {

                        JOptionPane.showMessageDialog(this, "Please fill all the fields");

                        return;

                    }


                    if (copies < 0) {

                        JOptionPane.showMessageDialog(this, "Copies cannot be negative");

                        return;

                    }


                    String query = "SELECT COUNT(*) FROM books WHERE id= ?";

statement = connection.prepareStatement(query);

statement.setString(1, isbn);                    resultSet = statement.executeQuery();

if (resultSet.next() && resultSet.getInt(1) > 0) {

                        JOptionPane.showMessageDialog(this, "Book with this ISBN already exists");
                        setFree();
return;
                    }
                    // Insert the book
                    String sql = "INSERT INTO books (id, title, author, language, genre,
no_of_copies, total) VALUES (?, ?, ?, ?, ?, ?, ?)";
                    statement = connection.prepareStatement(sql); statement.setString(1,
                    isbn); statement.setString(2, title);

                    statement.setString(3, author);
           statement.setString(4, language);
                    statement.setString(5, genre);
statement.setInt(6, copies);                    statement.setInt(7,
copies);                    statement.executeUpdate();
                    JOptionPane.showMessageDialog(this, "Book added successfully!");
                    setFree();


                 } else {
                    System.out.println("Failed to connect to the database");
                 }
              } catch (SQLException ex) {
ex.printStackTrace();
                 JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
              } finally {
                 // Close resources in finally block to ensure they are always closed
```

```
            try {                  if
(resultSet != null) {
resultSet.close();
                }
                if (statement != null) {
statement.close();
                }
                if (connection != null) {
                    connection.close();
                }
            } catch (SQLException ex) {
ex.printStackTrace();
            }
         }
      }


   }
   private void setFree() {
isbnField.setText("");
titleField.setText("");
authorField.setText("");
languageField.setText("");
genreField.setText("");
copiesField.setText("");
   }
}
```

ViewBooks.java

```
import java.sql.Connection; import
java.sql.SQLException; import
javax.swing.JFrame; import
javax.swing.JOptionPane; import
javax.swing.JScrollPane; import
javax.swing.JTable; import
javax.swing.table.DefaultTableMo
del; public class ViewBooks
extends JFrame {     static
Connection connection;
   public static DefaultTableModel tableModel;
public static JScrollPane scrollPane;     public
ViewBooks() throws SQLException {
connection = Connections.getConnection();
if (connection != null) {
         String[] columnNames = {"ID", "Title", "Author", "Language", "Genre", "Quantity"};
tableModel = new DefaultTableModel(columnNames, 0);
         JTable table = new JTable(tableModel);
String query = "SELECT * FROM books";
         tableModel = fetchBooks.fetchBooksList(tableModel, query, 1);
scrollPane = new JScrollPane(table);           scrollPane.setBounds(20,
10, 540, 290);           this.add(scrollPane);
      } else {
         JOptionPane.showMessageDialog(null, "Failed to connect to the database");
```

```
        }


        this.setTitle("View Books");
this.setLayout(null);        this.setSize(600,
550);        this.setVisible(true);
        this.setLocationRelativeTo(null);
    }}
```

ViewBorrowedBooks.java

```
import java.sql.Connection; import
java.sql.SQLException;


import javax.swing.JFrame; import
javax.swing.JOptionPane; import
javax.swing.JScrollPane; import
javax.swing.JTable;
import javax.swing.table.DefaultTableModel;


public class ViewBorrowedBooks extends JFrame{
public static DefaultTableModel tableModel;     public
static JScrollPane scrollPane;
    public ViewBorrowedBooks(int id) throws SQLException {
Connection connection = Connections.getConnection();        if
(connection != null) {
            String query = "SELECT * FROM books WHERE id IN (SELECT b_id FROM
issued_books WHERE user_id = " + id + ")";
```

```
            String[] columnNames = {"ID", "Title", "Author", "Language", "Genre", "Quantity"};
tableModel = new DefaultTableModel(columnNames, 0);                JTable table = new
JTable(tableModel);
            tableModel = fetchBooks.fetchBooksList(tableModel, query, 0);
scrollPane = new JScrollPane(table);             scrollPane.setBounds(20,
10, 540, 290);             this.add(scrollPane);
        } else {
            JOptionPane.showMessageDialog(null, "Failed to connect to the database");
        }
        this.setTitle("View Borrowed books");
this.setLayout(null);         this.setSize(600,
550);         this.setVisible(true);
        this.setLocationRelativeTo(null);
    }
}
```

fetchBooks.java import

```
java.sql.Connection; import
java.sql.ResultSet; import
java.sql.SQLException; import
java.sql.Statement;
import javax.swing.table.DefaultTableModel;

public class fetchBooks {
    public static DefaultTableModel fetchBooksList(DefaultTableModel tableModel, String
query, int view) throws SQLException {
        Connection connection = Connections.getConnection();
try (Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query)) {
tableModel.setRowCount(0);
tableModel.setColumnCount(0);


            if (view == 0) {
                tableModel.setColumnIdentifiers(new Object[]{"ID", "Title", "Author", "Language",
"Genre"});
            } else {
                tableModel.setColumnIdentifiers(new Object[]{"ID", "Title", "Author", "Language",
"Genre", "Quantity"});
            }
            while (resultSet.next()) {
                String id = resultSet.getString("id");
                String title = resultSet.getString("title");
                String author = resultSet.getString("author");
                String language = resultSet.getString("language");
String genre = resultSet.getString("genre");                int
quantity = resultSet.getInt("no_of_copies");
                Object[] rowData = {id, title, author, language, genre, quantity};
tableModel.addRow(rowData);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
        return tableModel;
```

```
    }
}
```

GetBook.java import

```
java.awt.Color; import
java.beans.Statement; import
java.sql.Connection; import
java.sql.SQLException; import
javax.swing.JButton; import
javax.swing.JFrame; import
javax.swing.JLabel; import
javax.swing.JOptionPane;
import javax.swing.JTextField;
import
javax.swing.UIManager;
public class GetBook extends
ViewBooks {     JFrame frame
= new JFrame();     public
JLabel isbnLabel;     public
JTextField isbnField;
    Connection connection;
Statement statement;
    int id;
    public GetBook(int id) throws SQLException {
        this.id = id;
        connection = Connections.getConnection();
UIManager.put("Label.foreground", Color.BLACK);         if(
connection != null) {

        isbnLabel = new JLabel("ISBN");
isbnField = new JTextField();
isbnLabel.setBounds(100, 350, 50, 25);
isbnField.setBounds(150, 350, 200, 25);
this.add(isbnLabel);            this.add(isbnField);

        JButton submitButton = new JButton("Get Book");
  submitButton.setBounds(150,      400,      100,      30);
  submitButton.addActionListener(e -> {
            new CheckAvailability(id).chk(isbnField.getText(), -1, "SELECT no_of_copies
FROM books WHERE id = ?");
            try {
                fetchBooks.fetchBooksList(tableModel, "SELECT * FROM books",1);
            } catch (SQLException e1) {
e1.printStackTrace();
            }});
        this.add(submitButton);
this.setTitle("Get Book");

        } else {
        JOptionPane.showMessageDialog(null, "Failed to connect to the database");
        }
    }
```

```
}
```

CheckAvailability.java import

```
java.sql.Connection; import
java.sql.PreparedStatement; import
java.sql.ResultSet; import
java.sql.SQLException; import
javax.swing.JOptionPane;

public class CheckAvailability {
   int id;
   public CheckAvailability(int id) {
      this.id = id;
   }
   void chk(String isbn,int get, String query) {
      try {
         Connection connection = Connections.getConnection();
         PreparedStatement statement = connection.prepareStatement(query);
statement.setString(1, isbn);
         ResultSet resultSet = statement.executeQuery();
if ( resultSet.next()) {
            if(get == -1 && resultSet.getInt("no_of_copies") == 0) {
JOptionPane.showMessageDialog(null, "Book not available");
               return;
            }
            new Update(id).reduce(isbn,get);
if(get == -1) {
               JOptionPane.showMessageDialog(null, "Book issued successfully");
            } else {
               JOptionPane.showMessageDialog(null, "Book returned successfully");
            }
return;
         }
      } catch (SQLException e) {
         e.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());
      }
      JOptionPane.showMessageDialog(null, "Book not found");   }}
```

Update.java import

```
java.sql.Connection; import
java.sql.PreparedStatement; import
java.sql.SQLException; import
javax.swing.JOptionPane; public
class Update {
   Connection connection = Connections.getConnection();
   int id;
   public Update(int id){
this.id = id;
   }
   public void reduce(String isbn, int inc) throws SQLException {
        try {
            Connection connection = Connections.getConnection();
            String query = "UPDATE books SET no_of_copies = no_of_copies + ? WHERE id =
?";
            PreparedStatement statement = connection.prepareStatement(query);
statement.setInt(1, inc);             statement.setString(2, isbn);
statement.executeUpdate();            if( inc == 1)
                query = "DELETE FROM issued_books WHERE b_id = ? AND user_id =
?";
            else
                query = "INSERT INTO issued_books (b_id, user_id) VALUES (?, ?)";
    statement = connection.prepareStatement(query);
            statement.setString(1, isbn);
statement.setInt(2, id);
statement.executeUpdate();          }
catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());
        }
    }
}
```

ReturnBook.java

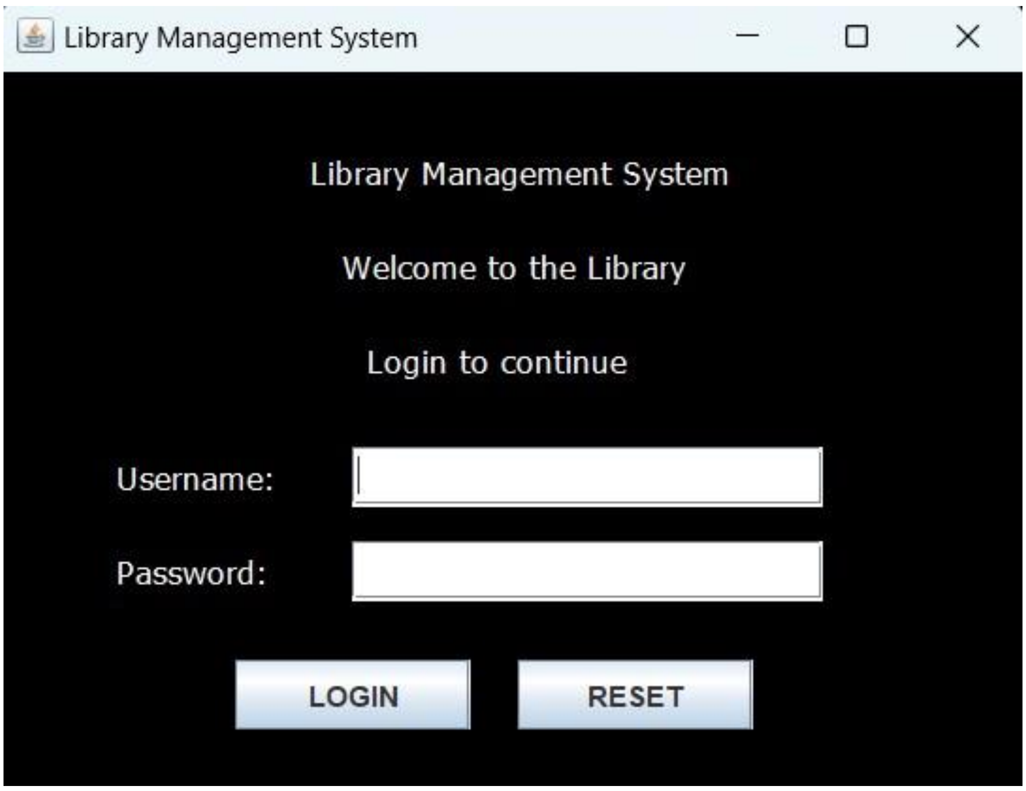
```
import java.awt.Color; import
java.beans.Statement; import
java.sql.Connection; import
java.sql.SQLException;
```

```
import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel; import

javax.swing.JOptionPane; import

javax.swing.JTextField; import

javax.swing.UIManager; public

class ReturnBook extends

ViewBorrowedBooks{     JFrame

frame = new JFrame();     public

JLabel isbnLabel;     public

JTextField isbnField;

   Connection connection;

Statement statement;

   public ReturnBook(int id) throws SQLException {

      super(id);

      connection = Connections.getConnection();

      UIManager.put("Label.foreground", Color.BLACK);

if( connection != null) {


         isbnLabel = new JLabel("ISBN");

isbnField = new JTextField();

isbnLabel.setBounds(100, 350, 50, 25);

isbnField.setBounds(150, 350, 200, 25);

this.add(isbnLabel);            this.add(isbnField);
```

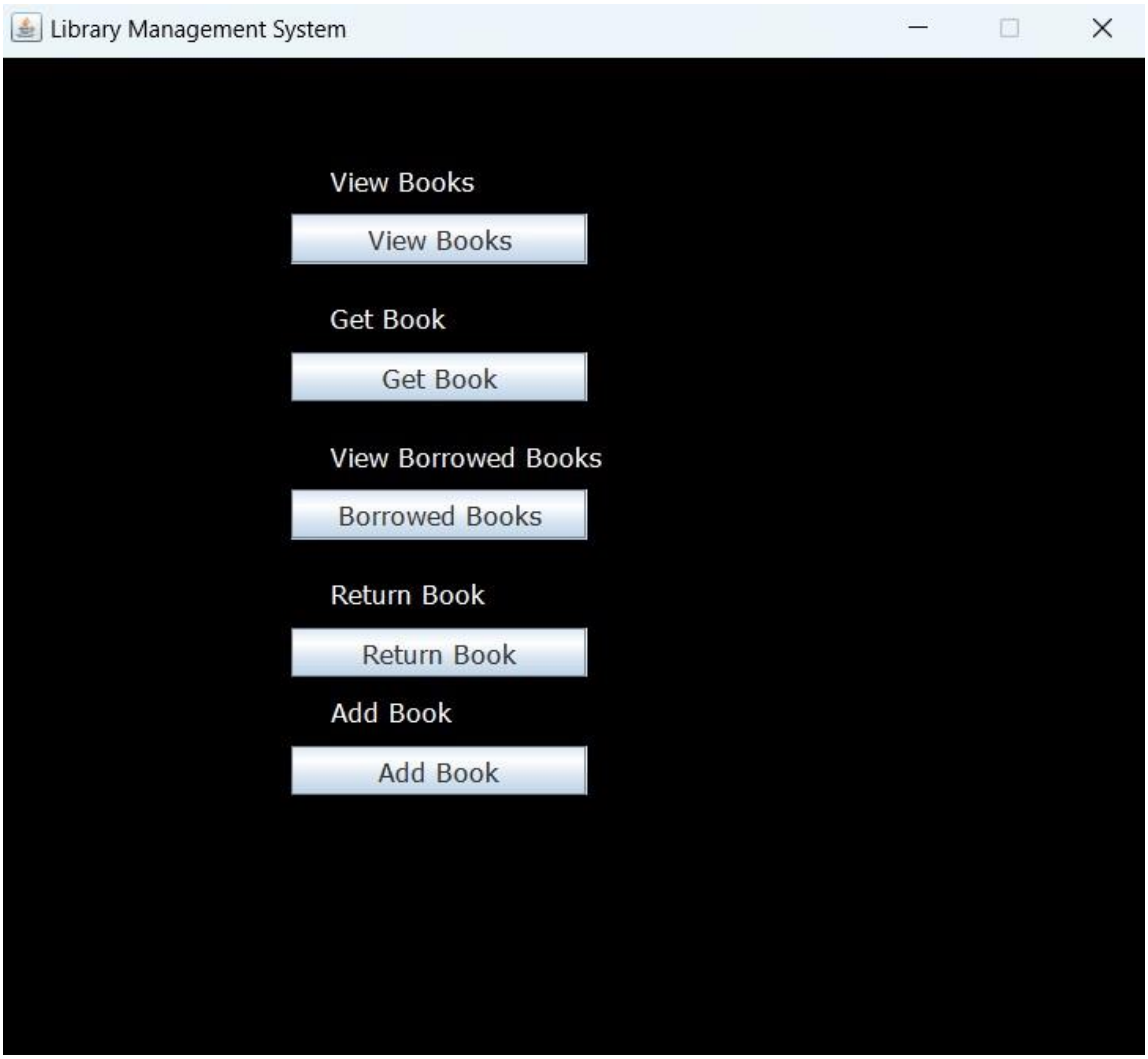
```
        JButton submitButton = new JButton("Return");
submitButton.setBounds(150,     400,     100,     30);
submitButton.addActionListener(e -> {
           try {
              new CheckAvailability(id).chk(isbnField.getText(), 1, "SELECT * FROM
issued_books WHERE b_id = ? and user_id = "+id);
              fetchBooks.fetchBooksList(tableModel, "SELECT * FROM books WHERE id IN
(SELECT b_id FROM issued_books WHERE user_id = " + id + ")",0);
           } catch (SQLException e1) {
             e1.printStackTrace();
           }
        } );
        this.setTitle("Return Book");
this.add(submitButton);
      } else {
        JOptionPane.showMessageDialog(null, "Failed to connect to the database");
      }
   }
}
```

5) OUTPUT

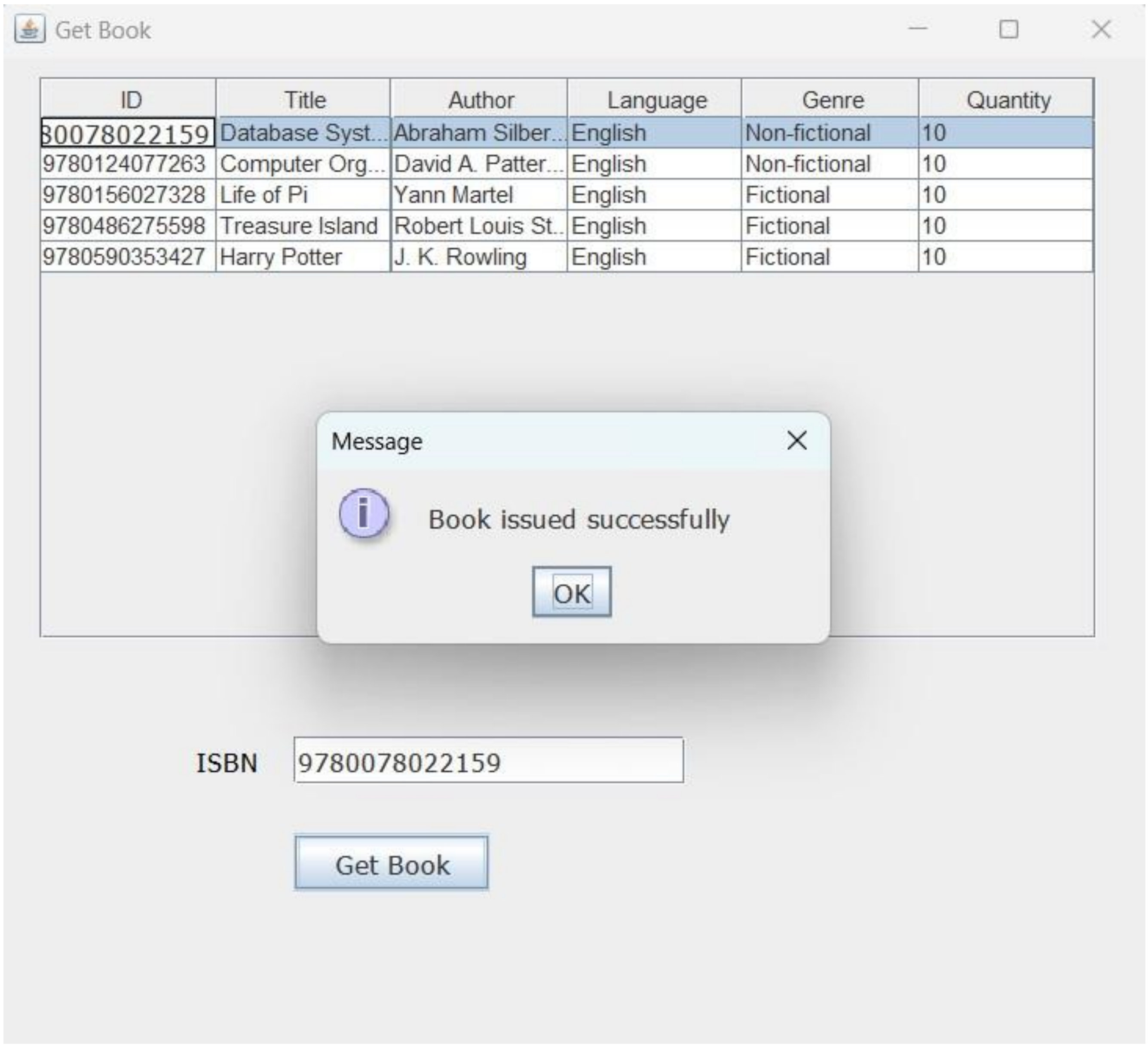
LOGIN PAGE:



LIBRARY:


Library Management System
View Books
View Books
Get Book
Get Book
View Borrowed Books
Borrowed Books
Return Book
Return Book
Add Book
Add Book

GET BOOK:



Get Book

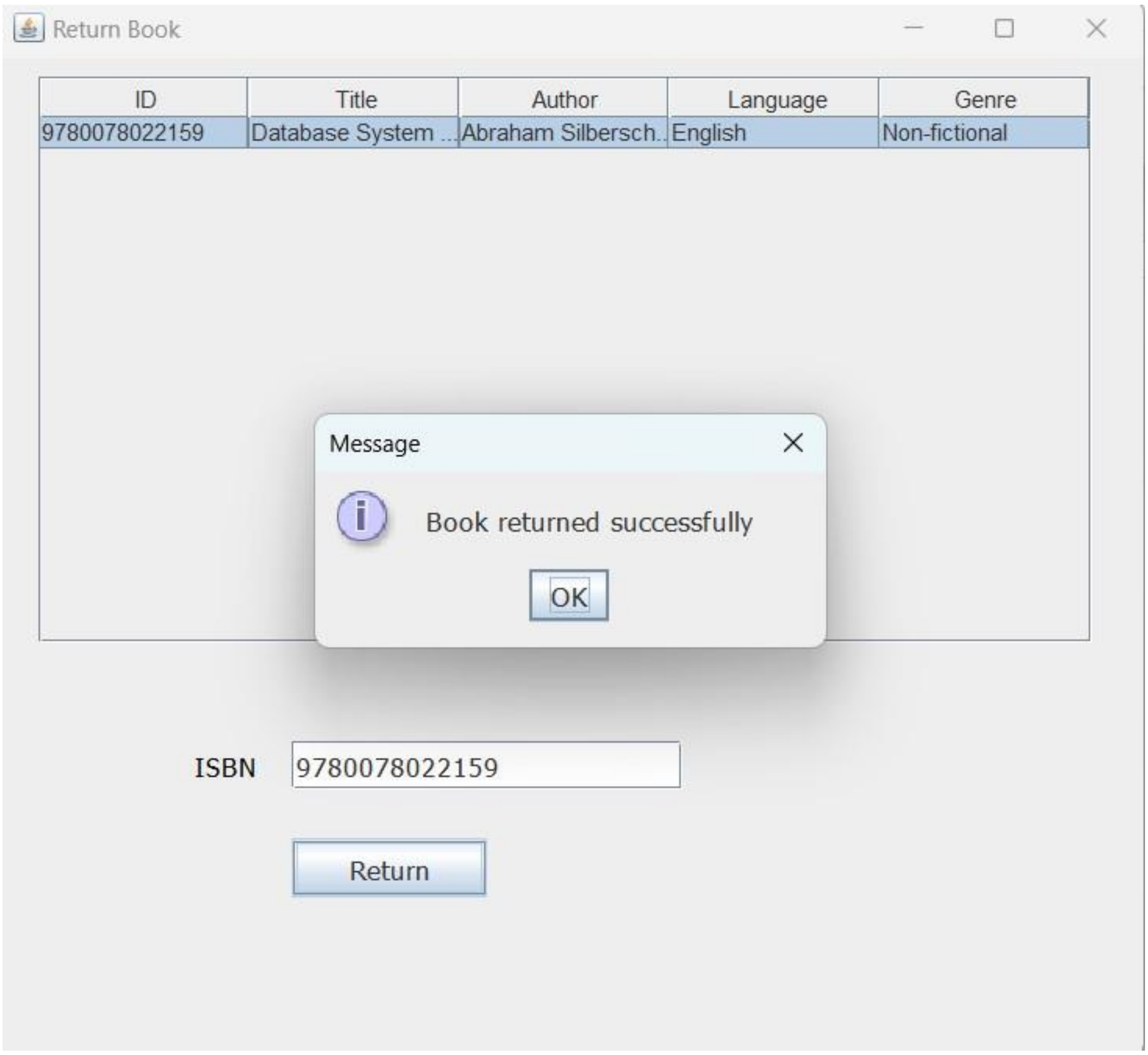
ID	Title	Author	Language	Genre	Quantity
80078022159	Database Syst...	Abraham Silber...	English	Non-fictional	10
9780124077263	Computer Org...	David A. Patter...	English	Non-fictional	10
9780156027328	Life of Pi	Yann Martel	English	Fictional	10
9780486275598	Treasure Island	Robert Louis St...	English	Fictional	10
9780590353427	Harry Potter	J. K. Rowling	English	Fictional	10

Message

Book issued successfully

ISBN 9780078022159

RETURN BOOK



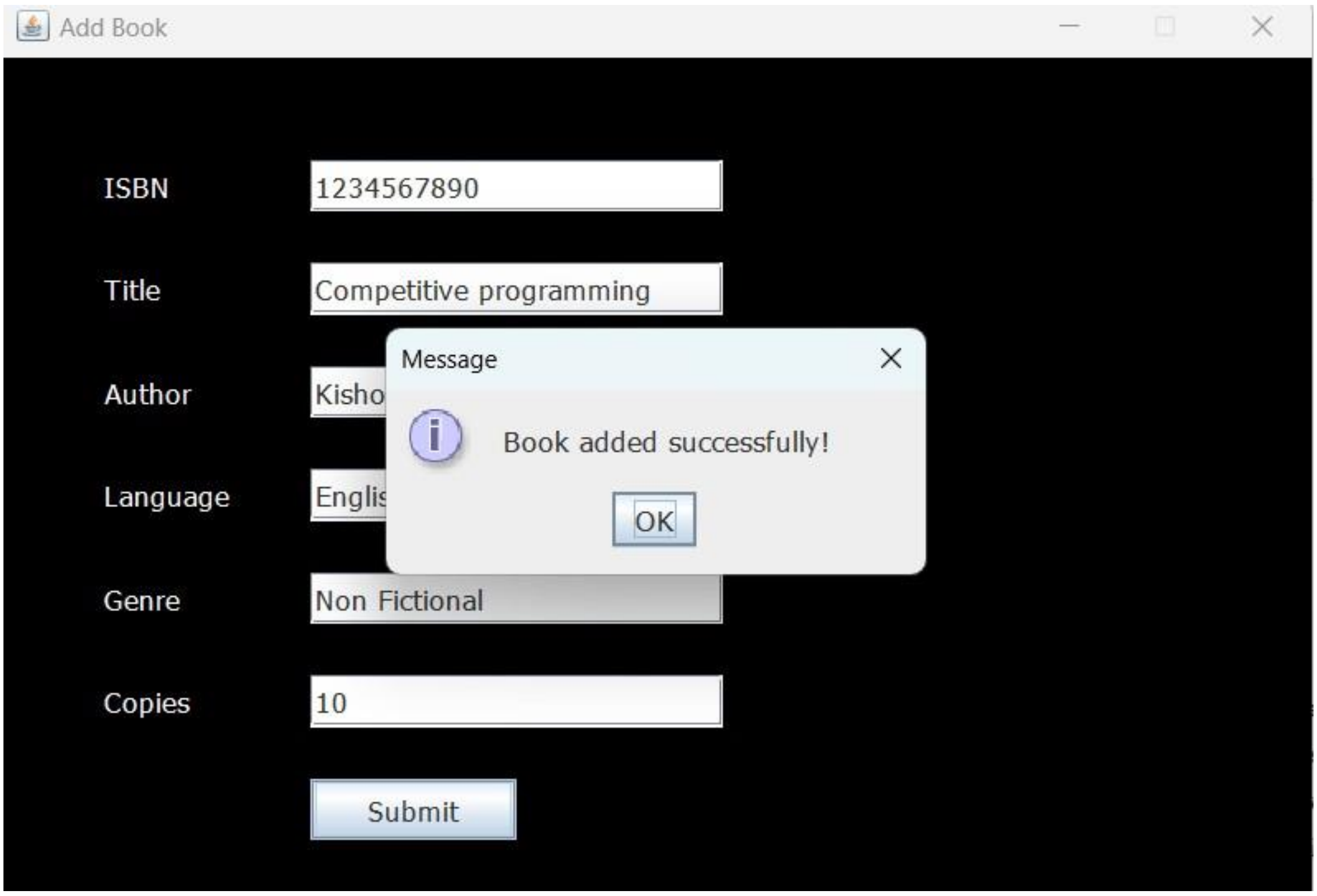
ID	Title	Author	Language	Genre
9780078022159	Database System ...	Abraham Silbersch...	English	Non-fictional

Message

Book returned successfully

ISBN 9780078022159

ADD BOOK:


Add Book

ISBN 1234567890

Title Competitive programming

Author Kisho

Language Englis

Genre Non Fictional

Copies 10

Message

Book added successfully!

6) RESULT AND DISCUSSION

The developed Library Management System (LMS) adeptly fulfills its primary objectives of efficiently managing library resources and user interactions. The ER diagram meticulously captures the intricate relationships between books, users, transactions, and library inventory. The incorporation of user registration, borrowing, and returning functionalities ensures comprehensive library management. Moreover, the system's scalability is evident through its inclusion of features like transactional details and inventory tracking. This mini project lays a sturdy foundation for future enhancements, including real-time updates, an intuitive user interface, and robust reporting capabilities, poised to revolutionize library operations and enhance user satisfaction significantly.

7) CONCLUSION:

In conclusion, this mini project signifies the successful creation of a fundamental Library Management System (LMS). The devised ER diagram establishes a robust framework for efficiently managing library resources, user interactions, and transactional details. With ample scope for future enhancements, such as real-time updates, an intuitive user interface, and comprehensive reporting capabilities, this project lays the groundwork for a comprehensive library management solution poised to streamline operations and enhance user experience effectively.

REFERENCES

1) Database System Concepts by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan: This classic textbook provides a comprehensive foundation for database systems. It covers relational model theory, database design principles, query languages (SQL), and transaction management. While not specifically focused on miniprojects, it offers valuable knowledge for building your project.

2) SQL in 10 Minutes, Sams Teach Yourself by Ben Forta: This beginnerfriendly book offers a quick introduction to SQL, the essential language for interacting with relational databases. It can equip you with the basic skills to build and manage your mini project's database structure**.**

3) Database Design for Mere Mortals by Michael J. Hernandez: This book focuses on practical database design techniques that can be applied to real-world scenarios. It guides you through the process of creating logical and efficient database structures, which is crucial for a successful mini project.