

# **RAJALAKSHMI ENGINEERING COLLEGE**

**[AUTONOMOUS]**

**RAJALAKSHMI NAGAR, THANDALAM – 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

**CS23333 OBJECT ORIENTED PROGRAMMING USING JAVA**

**Laboratory Record Note Book**

Name : .VISHAL.S.....

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

College Roll No. : ..... 2116231001248.....

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

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

**RAJALAKSHMI ENGINEERING COLLEGE**  
**[AUTONOMOUS]**

**RAJALAKSHMI NAGAR, THANDALAM – 602 105**

**BONAFIDE CERTIFICATE**

Name : .....VISHAL.S .....

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

2116231001248

**Register No.**

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

**Signature of Faculty in-charge**

Submitted for the Practical Examination held on ....27/11/2024.....

**Internal Examiner**

**External Examiner**

## INDEX

<b>Lab Week</b>	<b>Date</b>	<b>Name of the Experiment</b>	<b>Page No</b>	<b>Signature</b>
1	20.9.24	Java Architecture, Language Basics	1	
2	20.9.24	Flow Control Statements	5	
3	21.9.24	Arrays	11	
4	1.10.24	Classes And Objects	17	
5	1.10.24	Inheritance	23	
6	3.10.24	String, StringBuffer	29	
7	3.10.24	Interfaces	35	
8	6.10.24	Polymorphism, Abstract Classes, Final Keyword	41	
9	9.10.24	Exceptional Handling	47	
10	4.10.24	Collection - List	52	
11	10.11.24	Set, Map	57	
12	10.11.24	Introduction to I/O, I/O Operations, Object Serialization	63	
13	27.11.24	Java Project Report	72	

**Question 1**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class Odd
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n=sc.nextInt();
8         if(n%2!=0)
9         {
10             System.out.println(2);
11         }
12         else
13         {
14             System.out.println(1);
15         }
16     }
17 }
```

	Input	Expected	Got	
✓	123	2	2	✓
✓	456	1	1	✓

Passed all tests! ✓

**Question 1**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class lastDigit
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n=sc.nextInt();
8         int a=n%10;
9         int b=Math.abs(a);
10        System.out.println(b);
11    }
12 }
```

	Input	Expected	Got	
✓	197	7	7	✓
✓	-197	7	7	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of 5.00

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 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

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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class Sum
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int a=sc.nextInt();
8         int b=sc.nextInt();
9         int c=Math.abs(a%10);
10        int d=Math.abs(b%10);
11        System.out.println(c+d);
12    }
13 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	267 154	11	11	✓
✓	267 -154	11	11	✓
✓	-267 154	11	11	✓
✓	-267 -154	11	11	✓

Passed all tests! ✓



[◀ Lab-01-MCQ](#)

Jump to...

[Is Even? ►](#)

**Question 1**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class Pattern
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n=sc.nextInt();
8         int[] arr=new int[n];
9         arr[0]=0;
10        arr[1]=1;
11        arr[2]=1;
12        for(int i=3;i<n;i++)
13        {
14            arr[i]=arr[i-1]+arr[i-2]+arr[i-3];
15        }
16        System.out.println(arr[n-1]);
17    }
18 }
```

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

Passed all tests! ✓

**Question 2**

Correct

Marked out of 5.00

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

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

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

**Input Format**

Input containing an integer and a boolean value.

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

You have to print the time you should get up.

**Example Input:**

1 false

**Output:**

6:00

**Example Input:**

5 false

**Output:**

5:00

**Example Input:**

1 true

**Output:**

9:00

**For example:**

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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class Sports
3 {
4     public static void main(String[] main)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n=sc.nextInt();
8         boolean b=sc.nextBoolean();
9         if( (n==6 || n==2 || n==3 || n==4|| n==5) && b==false )
10        {
11            System.out.println("5:00");
12        }
13        else if( (n==1 || n==7) && b==false)
14        {
15            System.out.println("6:00");
16        }
    }
}
```

```
17     else if( (n==6 || n==2 || n==3 || n==4 || n==5) && b==true)
18     {
19         System.out.println("7:00");
20     }
21     else if( (n==1 || n==7) && b==true )
22     {
23         System.out.println("9:00");
24     }
25 }
26 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	1 false	6:00	6:00	✓
✓	5 false	5:00	5:00	✓
✓	1 true	9:00	9:00	✓

Passed all tests! ✓



**Question 3**

Correct

Marked out of 5.00

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

The movie's success formula depends on 2 parameters:

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

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

The movie is a hit if the acting power is excellent (more than 8) or the rating is excellent (more than 8). This holds true except if either the acting power is poor (less than 2) or rating is poor (less than 2), then the movie is a flop. Otherwise the movie is average.

Write a program that takes 2 integers:

the first integer is the acting power

second integer is the critic's rating.

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

Example input:

9 5

Output:

Yes

Example input:

1 9

Output:

No

Example input:

6 4

Output:

Maybe

**For example:**

Input	Result
9 5	Yes
1 9	No
6 4	Maybe

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class Movie
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int acting=sc.nextInt();
8         int rating=sc.nextInt();
9         if(acting<2 || rating<2)
10        {
11            System.out.println("No");
12        }
13        else if(acting>8 || rating>8)
14        {
15            System.out.println("Yes");
}

```

```
16 }  
17 else  
18 {  
19     System.out.println("Maybe");  
20 }  
21 }  
22 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	9 5	Yes	Yes	✓
✓	1 9	No	No	✓
✓	6 4	Maybe	Maybe	✓

Passed all tests! ✓



[◀ Lab-02-MCQ](#)

Jump to...

[Lab-03-MCQ ►](#)

**Question 1**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```
1 import java.util.Scanner;
2 public class Main
3 {
```

```

4   public static void main(String[] args)
5   {
6       Scanner sc=new Scanner(System.in);
7       int n=sc.nextInt();
8       int[] arr=new int[n];
9       for(int i=0;i<n;i++)
10      {
11          arr[i]=sc.nextInt();
12
13      }
14      int maxLen=0,currentLen=0,maxSum=0;
15      int currentSum=0;
16      boolean hasPositive=false;
17      for(int i=0;i<n;i++)
18      {
19          if(arr[i]>=0)
20          {
21              hasPositive=true;
22              currentLen++;
23              currentSum+=arr[i];
24
25          }
26          else
27          {
28              if(currentLen>maxLen)
29              {
30                  maxLen=currentLen;
31                  maxSum=currentSum;
32              }
33              else if(currentLen==maxLen)
34              {
35                  maxSum+=currentSum;
36              }
37              currentLen=0;
38              currentSum=0;
39
40          }
41      }
42      if(currentLen>maxLen)
43      {
44          maxSum=currentSum;
45      }
46      else if(currentLen==maxLen)
47      {
48          maxSum+=currentSum;
49      }
50  }
51  if(!hasPositive)
52  {

```

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

Passed all tests! ✓

**Question 2**

Correct

Marked out of 5.00

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:

$$\{(-8 \times 9), (-4 \times 9), (3 \times 9), (0 \times 9)\} = \{-72, -36, -27, 0\}$$

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:

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

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:

$$\{(-18 \times 9), (0 \times 9)\} = \{-162, 0\}$$

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
2 -9 9	-162 0

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int input1=sc.nextInt();
8         int[] input2=new int[input1];
9         for(int i=0;i<input1;i++)
10        {
11            input2[i]=sc.nextInt();
12        }
13        int max=Integer.MIN_VALUE;
14        for(int i=0;i<input1;i++)
15        {
16            if(input2[i]>max)
17            {
18                max=input2[i];
19            }
20        }
21        for(int i=0;i<input1;i++)
22        {
23            input2[i]=(input2[i]-max)*max;
24        }
25        for(int i=0;i<input1;i++)
26        {
27            System.out.print(input2[i]+" ");
28        }
29    }
30 }
```

	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! ✓

**Question 3**

Correct

Marked out of 5.00

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

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

This is explained below:

Example 1:

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

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

Step 1:

Starting from the 0<sup>th</sup> index of the array pick up digits as per below:

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

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

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

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

4<sup>th</sup> 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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int input1=sc.nextInt();
8         int[] input2=new int[input1];
9         for(int i=0;i<input1;i++)
10        {
11            input2[i]=sc.nextInt();
12        }
13    }
14    int[] resultArray=new int[input1];
15    for(int i=0;i<input1;i++)
16    {
17        String numStr=String.valueOf(input2[i]);
18        int position=numStr.length()-1-i;
19        if(position>=0)
20        {
21            resultArray[i]=Character.getNumericValue(numStr.charAt(position));
22        }
23        else
24        {
25            resultArray[i]=0;
26        }
27    }
28    int sum=0;
29    for(int i=0;i<input1;i++)
30    {
31        resultArray[i]*=resultArray[i];
32        sum+=resultArray[i];
33    }
34    System.out.println(sum);
35 }
36 }
```

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

Passed all tests! ✓

◀ Lab-03-MCQ

Jump to...

Simple Encoded Array ►

**Question 1**

Correct

Marked out of 5.00

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 =  $\pi r^2$**

**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

**Answer:** (penalty regime: 0 %)

Reset answer

```

1 import java.io.*;
2 import java.util.Scanner;
3 class Circle
4 {
5     private double radius;
6     public Circle(double radius){
7         // set the instance variable radius
8         this.radius=radius;
9     }
10    }
11    public void setRadius(double radius){
12        // set the radius
13        this.radius=radius;
14    }
15    }
16    public double getRadius()    {
17        // return the radius
18        return radius;
19    }
20    }
21    public double calculateArea() { // complete the below statement
22        return Math.PI*radius*radius;
23    }
24    public double calculateCircumference()    {
25        // complete the statement
26        return 2*Math.PI*radius;
27    }
28    }
29    class prog{
30        public static void main(String[] args)  {
31            int r;
32            Scanner sc= new Scanner(System.in);
33            r=sc.nextInt();
34            Circle c= new Circle(r);
35            System.out.println("Area = "+String.format("%.2f", c.calculateArea()));
36            System.out.println("Circumference = "+String.format("%.2f",c.calculateCircumference()));
37        }
38    }
39    }
40    }
41    }
42    }
43    }
44    }
45    }
46    }
47    }
48    }
49    }
50    }
51    }
52    }
53    }
54    }
55    }
56    }
57    }
58    }
59    }
60    }
61    }
62    }
63    }
64    }
65    }
66    }
67    }
68    }
69    }
70    }
71    }
72    }
73    }
74    }
75    }
76    }
77    }
78    }
79    }
80    }
81    }
82    }
83    }
84    }
85    }
86    }
87    }
88    }
89    }
90    }
91    }
92    }
93    }
94    }
95    }
96    }
97    }
98    }
99    }
100    }
101    }
102    }
103    }
104    }
105    }
106    }
107    }
108    }
109    }
110    }
111    }
112    }
113    }
114    }
115    }
116    }
117    }
118    }
119    }
120    }
121    }
122    }
123    }
124    }
125    }
126    }
127    }
128    }
129    }
130    }
131    }
132    }
133    }
134    }
135    }
136    }
137    }
138    }
139    }
140    }
141    }
142    }
143    }
144    }
145    }
146    }
147    }
148    }
149    }
150    }
151    }
152    }
153    }
154    }
155    }
156    }
157    }
158    }
159    }
160    }
161    }
162    }
163    }
164    }
165    }
166    }
167    }
168    }
169    }
170    }
171    }
172    }
173    }
174    }
175    }
176    }
177    }
178    }
179    }
180    }
181    }
182    }
183    }
184    }
185    }
186    }
187    }
188    }
189    }
190    }
191    }
192    }
193    }
194    }
195    }
196    }
197    }
198    }
199    }
200    }
201    }
202    }
203    }
204    }
205    }
206    }
207    }
208    }
209    }
210    }
211    }
212    }
213    }
214    }
215    }
216    }
217    }
218    }
219    }
220    }
221    }
222    }
223    }
224    }
225    }
226    }
227    }
228    }
229    }
230    }
231    }
232    }
233    }
234    }
235    }
236    }
237    }
238    }
239    }
240    }
241    }
242    }
243    }
244    }
245    }
246    }
247    }
248    }
249    }
250    }
251    }
252    }
253    }
254    }
255    }
256    }
257    }
258    }
259    }
260    }
261    }
262    }
263    }
264    }
265    }
266    }
267    }
268    }
269    }
270    }
271    }
272    }
273    }
274    }
275    }
276    }
277    }
278    }
279    }
280    }
281    }
282    }
283    }
284    }
285    }
286    }
287    }
288    }
289    }
290    }
291    }
292    }
293    }
294    }
295    }
296    }
297    }
298    }
299    }
300    }
301    }
302    }
303    }
304    }
305    }
306    }
307    }
308    }
309    }
310    }
311    }
312    }
313    }
314    }
315    }
316    }
317    }
318    }
319    }
320    }
321    }
322    }
323    }
324    }
325    }
326    }
327    }
328    }
329    }
330    }
331    }
332    }
333    }
334    }
335    }
336    }
337    }
338    }
339    }
340    }
341    }
342    }
343    }
344    }
345    }
346    }
347    }
348    }
349    }
350    }
351    }
352    }
353    }
354    }
355    }
356    }
357    }
358    }
359    }
360    }
361    }
362    }
363    }
364    }
365    }
366    }
367    }
368    }
369    }
370    }
371    }
372    }
373    }
374    }
375    }
376    }
377    }
378    }
379    }
380    }
381    }
382    }
383    }
384    }
385    }
386    }
387    }
388    }
389    }
390    }
391    }
392    }
393    }
394    }
395    }
396    }
397    }
398    }
399    }
400    }
401    }
402    }
403    }
404    }
405    }
406    }
407    }
408    }
409    }
410    }
411    }
412    }
413    }
414    }
415    }
416    }
417    }
418    }
419    }
420    }
421    }
422    }
423    }
424    }
425    }
426    }
427    }
428    }
429    }
430    }
431    }
432    }
433    }
434    }
435    }
436    }
437    }
438    }
439    }
440    }
441    }
442    }
443    }
444    }
445    }
446    }
447    }
448    }
449    }
450    }
451    }
452    }
453    }
454    }
455    }
456    }
457    }
458    }
459    }
460    }
461    }
462    }
463    }
464    }
465    }
466    }
467    }
468    }
469    }
470    }
471    }
472    }
473    }
474    }
475    }
476    }
477    }
478    }
479    }
480    }
481    }
482    }
483    }
484    }
485    }
486    }
487    }
488    }
489    }
490    }
491    }
492    }
493    }
494    }
495    }
496    }
497    }
498    }
499    }
500    }
501    }
502    }
503    }
504    }
505    }
506    }
507    }
508    }
509    }
510    }
511    }
512    }
513    }
514    }
515    }
516    }
517    }
518    }
519    }
520    }
521    }
522    }
523    }
524    }
525    }
526    }
527    }
528    }
529    }
530    }
531    }
532    }
533    }
534    }
535    }
536    }
537    }
538    }
539    }
540    }
541    }
542    }
543    }
544    }
545    }
546    }
547    }
548    }
549    }
550    }
551    }
552    }
553    }
554    }
555    }
556    }
557    }
558    }
559    }
560    }
561    }
562    }
563    }
564    }
565    }
566    }
567    }
568    }
569    }
570    }
571    }
572    }
573    }
574    }
575    }
576    }
577    }
578    }
579    }
580    }
581    }
582    }
583    }
584    }
585    }
586    }
587    }
588    }
589    }
590    }
591    }
592    }
593    }
594    }
595    }
596    }
597    }
598    }
599    }
600    }
601    }
602    }
603    }
604    }
605    }
606    }
607    }
608    }
609    }
610    }
611    }
612    }
613    }
614    }
615    }
616    }
617    }
618    }
619    }
620    }
621    }
622    }
623    }
624    }
625    }
626    }
627    }
628    }
629    }
630    }
631    }
632    }
633    }
634    }
635    }
636    }
637    }
638    }
639    }
640    }
641    }
642    }
643    }
644    }
645    }
646    }
647    }
648    }
649    }
650    }
651    }
652    }
653    }
654    }
655    }
656    }
657    }
658    }
659    }
660    }
661    }
662    }
663    }
664    }
665    }
666    }
667    }
668    }
669    }
670    }
671    }
672    }
673    }
674    }
675    }
676    }
677    }
678    }
679    }
680    }
681    }
682    }
683    }
684    }
685    }
686    }
687    }
688    }
689    }
690    }
691    }
692    }
693    }
694    }
695    }
696    }
697    }
698    }
699    }
700    }
701    }
702    }
703    }
704    }
705    }
706    }
707    }
708    }
709    }
710    }
711    }
712    }
713    }
714    }
715    }
716    }
717    }
718    }
719    }
720    }
721    }
722    }
723    }
724    }
725    }
726    }
727    }
728    }
729    }
730    }
731    }
732    }
733    }
734    }
735    }
736    }
737    }
738    }
739    }
740    }
741    }
742    }
743    }
744    }
745    }
746    }
747    }
748    }
749    }
750    }
751    }
752    }
753    }
754    }
755    }
756    }
757    }
758    }
759    }
760    }
761    }
762    }
763    }
764    }
765    }
766    }
767    }
768    }
769    }
770    }
771    }
772    }
773    }
774    }
775    }
776    }
777    }
778    }
779    }
780    }
781    }
782    }
783    }
784    }
785    }
786    }
787    }
788    }
789    }
790    }
791    }
792    }
793    }
794    }
795    }
796    }
797    }
798    }
799    }
800    }
801    }
802    }
803    }
804    }
805    }
806    }
807    }
808    }
809    }
810    }
811    }
812    }
813    }
814    }
815    }
816    }
817    }
818    }
819    }
820    }
821    }
822    }
823    }
824    }
825    }
826    }
827    }
828    }
829    }
830    }
831    }
832    }
833    }
834    }
835    }
836    }
837    }
838    }
839    }
840    }
841    }
842    }
843    }
844    }
845    }
846    }
847    }
848    }
849    }
850    }
851    }
852    }
853    }
854    }
855    }
856    }
857    }
858    }
859    }
860    }
861    }
862    }
863    }
864    }
865    }
866    }
867    }
868    }
869    }
870    }
871    }
872    }
873    }
874    }
875    }
876    }
877    }
878    }
879    }
880    }
881    }
882    }
883    }
884    }
885    }
886    }
887    }
888    }
889    }
890    }
891    }
892    }
893    }
894    }
895    }
896    }
897    }
898    }
899    }
900    }
901    }
902    }
903    }
904    }
905    }
906    }
907    }
908    }
909    }
910    }
911    }
912    }
913    }
914    }
915    }
916    }
917    }
918    }
919    }
920    }
921    }
922    }
923    }
924    }
925    }
926    }
927    }
928    }
929    }
930    }
931    }
932    }
933    }
934    }
935    }
936    }
937    }
938    }
939    }
940    }
941    }
942    }
943    }
944    }
945    }
946    }
947    }
948    }
949    }
950    }
951    }
952    }
953    }
954    }
955    }
956    }
957    }
958    }
959    }
960    }
961    }
962    }
963    }
964    }
965    }
966    }
967    }
968    }
969    }
970    }
971    }
972    }
973    }
974    }
975    }
976    }
977    }
978    }
979    }
980    }
981    }
982    }
983    }
984    }
985    }
986    }
987    }
988    }
989    }
990    }
991    }
992    }
993    }
994    }
995    }
996    }
997    }
998    }
999    }
1000    }
1001    }
1002    }
1003    }
1004    }
1005    }
1006    }
1007    }
1008    }
1009    }
1010    }
1011    }
1012    }
1013    }
1014    }
1015    }
1016    }
1017    }
1018    }
1019    }
1020    }
1021    }
1022    }
1023    }
1024    }
1025    }
1026    }
1027    }
1028    }
1029    }
1030    }
1031    }
1032    }
1033    }
1034    }
1035    }
1036    }
1037    }
1038    }
1039    }
1040    }
1041    }
1042    }
1043    }
1044    }
1045    }
1046    }
1047    }
1048    }
1049    }
1050    }
1051    }
1052    }
1053    }
1054    }
1055    }
1056    }
1057    }
1058    }
1059    }
1060    }
1061    }
1062    }
1063    }
1064    }
1065    }
1066    }
1067    }
1068    }
1069    }
1070    }
1071    }
1072    }
1073    }
1074    }
1075    }
1076    }
1077    }
1078    }
1079    }
1080    }
1081    }
1082    }
1083    }
1084    }
1085    }
1086    }
1087    }
1088    }
1089    }
1090    }
1091    }
1092    }
1093    }
1094    }
1095    }
1096    }
1097    }
1098    }
1099    }
1100    }
1101    }
1102    }
1103    }
1104    }
1105    }
1106    }
1107    }
1108    }
1109    }
1110    }
1111    }
1112    }
1113    }
1114    }
1115    }
1116    }
1117    }
1118    }
1119    }
1120    }
1121    }
1122    }
1123    }
1124    }
1125    }
1126    }
1127    }
1128    }
1129    }
1130    }
1131    }
1132    }
1133    }
1134    }
1135    }
1136    }
1137    }
1138    }
1139    }
1140    }
1141    }
1142    }
1143    }
1144    }
1145    }
1146    }
1147    }
1148    }
1149    }
1150    }
1151    }
1152    }
1153    }
1154    }
1155    }
1156    }
1157    }
1158    }
1159    }
1160    }
1161    }
1162    }
1163    }
1164    }
1165    }
1166    }
1167    }
1168    }
1169    }
1170    }
1171    }
1172    }
1173    }
1174    }
1175    }
1176    }
1177    }
1178    }
1179    }
1180    }
1181    }
1182    }
1183    }
1184    }
1185    }
1186    }
1187    }
1188    }
1189    }
1190    }
1191    }
1192    }
1193    }
1194    }
1195    }
1196    }
1197    }
1198    }
1199    }
1200    }
1201    }
1202    }
1203    }
1204    }
1205    }
1206    }
1207    }
1208    }
1209    }
1210    }
1211    }
1212    }
1213    }
1214    }
1215    }
1216    }
1217    }
1218    }
1219    }
1220    }
1221    }
1222    }
1223    }
1224    }
1225    }
1226    }
1227    }
1228    }
1229    }
1230    }
1231    }
1232    }
1233    }
1234    }
1235    }
1236    }
1237    }
1238    }
1239    }
1240    }
1241    }
1242    }
1243    }
1244    }
1245    }
1246    }
1247    }
1248    }
1249    }
1250    }
1251    }
1252    }
1253    }
1254    }
1255    }
1256    }
1257    }
1258    }
1259    }
1260    }
1261    }
1262    }
1263    }
1264    }
1265    }
1266    }
1267    }
1268    }
1269    }
1270    }
1271    }
1272    }
1273    }
1274    }
1275    }
1276    }
1277    }
1278    }
1279    }
1280    }
1281    }
1282    }
1283    }
1284    }
1285    }
1286    }
1287    }
1288    }
1289    }
1290    }
1291    }
1292    }
1293    }
1294    }
1295    }
1296    }
1297    }
1298    }
1299    }
1300    }
1301    }
1302    }
1303    }
1304    }
1305    }
1306    }
1307    }
1308    }
1309    }
1310    }
1311    }
1312    }
1313    }
1314    }
1315    }
1316    }
1317    }
1318    }
1319    }
1320    }
1321    }
1322    }
1323    }
1324    }
1325    }
1326    }
1327    }
1328    }
1329    }
1330    }
1331    }
1332    }
1333    }
1334    }
1335    }
1336    }
1337    }
1338    }
1339    }
1340    }
1341    }
1342    }
1343    }
1344    }
1345    }
1346    }
1347    }
1348    }
1349    }
1350    }
1351    }
1352    }
1353    }
1354    }
1355    }
1356    }
1357    }
1358    }
1359    }
1360    }
1361    }
1362    }
1363    }
1364    }
1365    }
1366    }
1367    }
1368    }
1369    }
1370    }
1371    }
1372    }
1373    }
1374    }
1375    }
1376    }
1377    }
1378    }
1379    }
1380    }
1381    }
1382    }
1383    }
1384    }
1385    }
1386    }
1387    }
1388    }
1389    }
1390    }
1391    }
1392    }
1393    }
1394    }
1395    }
1396    }
1397    }
1398    }
1399    }
1400    }
1401    }
1402    }
1403    }
1404    }
1405    }
1406    }
1407    }
1408    }
1409    }
1410    }
1411    }
1412    }
1413    }
1414    }
1415    }
1416    }
1417    }
1418    }
1419    }
1420    }
1421    }
1422    }
1423    }
1424    }
1425    }
1426    }
1427    }
1428    }
1429    }
1430    }
1431    }
1432    }
1433    }
1434    }
1435    }
1436    }
1437    }
1438    }
1439    }
1440    }
1441    }
1442    }
1443    }
1444    }
1445    }
1446    }
1447    }
1448    }
1449    }
1450    }
1451    }
1452    }
1453    }
1454    }
1455    }
1456    }
1457    }
1458    }
1459    }
1460    }
1461    }
1462    }
1463    }
1464    }
1465    }
1466    }
1467    }
1468    }
1469    }
1470    }
1471    }
1472    }
1473    }
1474    }
1475    }
1476    }
1477    }
1478    }
1479    }
1480    }
1481    }
1482    }
1483    }
1484    }
1485    }
1486    }
1487    }
1488    }
1489    }
1490    }
1491    }
1492    }
1493    }
1494    }
1495    }
1496    }
1497    }
1498    }
1499    }
1500    }
1501    }
1502    }
1503    }
1504    }
1505    }
1506    }
1507    }
1508    }
1509    }
1510    }
1511    }
1512    }
1513    }
1514    }
1515    }
1516    }
1517    }
1518    }
1519    }
1520    }
1521    }
1522    }
1523    }
1524    }
1525    }
1526    }
1527    }
1528    }
1529    }
1530    }
1531    }
1532    }
1533    }
1534    }
1535    }
1536    }
1537    }
1538    }
1539    }
1540    }
1541    }
1542    }
1543    }
1544    }
1545    }
1546    }
1547    }
1548    }
1549    }
1550    }
1551    }
1552    }
1553    }
1554    }
1555    }
1556    }
1557    }
1558    }
1559    }
1560    }
1561    }
1562    }
1563    }
1564    }
1565    }
1566    }
1567    }
1568    }
1569    }
1570    }
1571    }
1572    }
1573    }
1574    }
1575    }
1576    }
1577    }
1578    }
1579    }
1580    }
1581    }
1582    }
1583    }
1584    }
1585    }
1586    }
1587    }
1588    }
1589    }
1590    }
1591    }
1592    }
1593    }
1594    }
1595    }
1596    }
1597    }
1598    }
1599    }
15100    }
15101    }
15102    }
15103    }
15104    }
15105    }
15106    }
15107    }
15108    }
15109    }
15110    }
15111    }
15112    }
15113    }
15114    }
15115    }
15116    }
15117    }
15118    }
15119    }
15120    }
15121    }
15122    }
15123    }
15124    }
15125    }
15126    }
15127    }
15128    }
15129    }
15130    }
15131    }
15132    }
15133    }
15134    }
15135    }
15136    }
15137    }
15138    }
15139    }
15140    }
15141    }
15142    }
15143    }
15144    }
15145    }
15146    }
15147    }
15148    }
15149    }
15150    }
15151    }
15152    }
15153    }
15154    }
15155    }
15156    }
15157    }
15158    }
15159    }
15160    }
15161    }
15162    }
15163    }
15164    }
15165    }
15166    }
15167    }
15168    }
15169    }
15170    }
15171    }
15172    }
15173    }
15174    }
15175    }
15176    }
15177    }
15178    }
15179    }
15180    }
15181    }
15182    }
15183    }
15184    }
15185    }
15186    }
15187    }
15188    }
15189    }
15190    }
15191    }
15192    }
15193    }
15194    }
15195    }
15196    }
15197    }
15198    }
15199    }
15200    }
15201    }
15202    }
15203    }
15204    }
15205    }
15206    }
15207    }
15208    }
15209    }
15210    }
15211    }
15212    }
15213    }
15214    }
15215    }
15216    }
15217    }
15218    }
15219    }
15220    }
15221    }
15222    }
15223    }
15224    }
15225    }
15226    }
15227    }
15228    }
15229    }
15230    }
15231    }
15232    }
15233    }
15234    }
15235    }
15236    }
15237    }
15238    }
15239    }
15240    }
15241    }
15242    }
15243    }
15244    }
15245    }
15246    }
15247    }
15248    }
15249    }
15250    }
15251    }
15252    }
15253    }
15254    }
15255    }
15256    }
15257    }
15258    }
15259    }
15260    }
15261    }
15262    }
15263    }
15264    }
15265    }
15266    }
15267    }
15268    }
15269    }
15270    }
15271    }
15272    }
15273    }
15274    }
15275    }
15276    }
15277    }
15278    }
15279    }
15280    }
15281    }
15282    }
15283    }
15284    }
15285    }
15286    }
15287    }
15288    }
15289    }
15290    }
15291    }
15292    }
15293    }
15294    }
15295    }
15296    }
15297    }
15298    }
15299    }
15300    }
15301    }
15302    }
15303    }
15304    }
15305    }
15306    }
15307    }
15308    }
15309    }
15310    }
15311    }
15312    }
15313    }
15314    }
15315    }
15316    }
15317    }
15318    }
15319    }
15320    }
15321    }
15322    }
15323    }
15324    }
15325    }
15326    }
15327    }
15328    }
15329    }
15330    }
15331    }
15332    }
15333    }
15334    }
15335    }
15336    }
15337    }
15338    }
15339    }
15340    }
15341    }
15342    }
15343    }
15344    }
15345    }
15346    }
15347    }
15348    }
15349    }
15350    }
15351    }
15352    }
15353    }
15354    }
15355    }
15356    }
15357    }
15358    }
15359    }
15360    }
15361
```

```
41  
42  
43 }  
44 }  
45 }
```

	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! ✓

**Question 2**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```
1 class prog
2 {
3     private String manufacturer;
4     private String operating_system;
5     private String color;
6     private int cost;
7     public prog(String manufacturer, String operating_system, String color, int cost)
8     {
9         this.manufacturer= manufacturer;
10        this.operating_system= operating_system;
11        this.color= color;
12        this.cost= cost;
13    }
14    public void setManufacturer(String manufacturer)
15    {
16        this.manufacturer= manufacturer;
17    }
18    public void setOperating_system(String operating_system)
19    {
20        this.operating_system= operating_system;
21    }
22    public void setColor(String color)
23    {
24        this.color= color;
25    }
26    public void setCost(int cost)
27    {
28        this.cost= cost;
29    }
30    public String getManufacturer()
31    {
32        return manufacturer;
33    }
34}
```

```
34     }
35     public String getOperating_system()
36     {
37         return operating_system;
38     }
39     public String getColor()
40     {
41         return color;
42     }
43     public int getCost()
44     {
45         return cost;
46     }
47     public String toString()
48     {
49         return "manufacturer = "+manufacturer+"\n"+
50             "operating_system = "+operating_system+"\n"+
51             "color = "+color+"\n"+
52             "cost = "+cost;
53     }
54 }
```

	<b>Test</b>	<b>Expected</b>	<b>Got</b>	
✓	1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```

1  class prog
2  {
3      private String name;
4      private int rollno;
5      public prog()
6      {
7          this.name=null;
8          this.rollno=0;
9          System.out.println("No-arg constructor is invoked");
10     }
11     public prog(String name)
12     {
13         this.name=name;
14         this.rollno=0;
15         System.out.println("1 arg constructor is invoked");
16     }
17     public prog(String name,int rollno)
18     {
19         this.name=name;
20         this.rollno=rollno;
21         System.out.println("2 arg constructor is invoked");
22     }
23     public void display()
24     {
25         System.out.println("Name =" +name+ " , Roll no = "+rollno);
26     }
27     public static void main(String[] args)
28     {
29         prog stu1=new prog();
30         prog stu2=new prog("Rajalakshmi");
31         prog stu3=new prog("Lakshmi",101);
32         stu1.display();

```

```
-- 33     stu2.display();  
34     stu3.display();  
35 }  
36 }
```

	<b>Test</b>	<b>Expected</b>	<b>Got</b>	
✓	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-04-MCQ

Jump to...

Number of Primes in a specified range ►

**Question 1**

Correct

Marked out of 5.00

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
```

**Answer:** (penalty regime: 0 %)

[Reset answer](#)

```
1 class BankAccount {  
2     // Private field to store the account number  
3     private String accountNumber;  
4  
5     // Private field to store the balance  
6     private double balance;  
7  
8     // Constructor to initialize account number and balance  
9     BankAccount(String ac,double bal){  
10         accountNumber = ac;  
11         balance = bal;  
12     }  
13     // Method to deposit an amount into the account  
14     public void deposit(double amount) {  
15         // Increase the balance by the deposit amount  
16         balance +=amount;  
17     }  
18  
19     // Method to withdraw an amount from the account  
20     public void withdraw(double amount) {  
21         // Check if the balance is sufficient for the withdrawal  
22         if (balance >= amount) {  
23             // Decrease the balance by the withdrawal amount  
24             balance -= amount;  
25         } else {  
26             // Print a message if the balance is insufficient  
27             System.out.println("Insufficient balance");  
28         }  
29     }  
30  
31     // Method to get the current balance  
32     public double getBalance() {  
33         // Return the current balance  
34         return balance;  
35     }  
36  
37 }  
38  
39 class SavingsAccount extends BankAccount {  
40     // Constructor to initialize account number and balance  
41     public SavingsAccount(String accountNumber, double balance) {  
42 }
```

```

43     // Call the parent class constructor
44     super(accountNumber,balance);
45 }
46 // Override the withdraw method from the parent class
47 @Override
48 public void withdraw(double amount) {
49     // Check if the withdrawal would cause the balance to drop below $100
50     if (getBalance() - amount < 100) {
51         // Print a message if the minimum balance requirement is not met
52         System.out.println("Minimum balance of $100 required!");

```

	<b>Expected</b>	<b>Got</b>	
✓	Create a Bank Account object (A/c No. BA1234) 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) 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! ✓

**Question 2**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```
1 class College
2 {
3     protected String collegeName;
4
5     public College(String collegeName) {
6         // initialize the instance variables
7         this.collegeName = collegeName;
8
9     }
10
11    public void admitted() {
12        System.out.println("A student admitted in "+collegeName);
13    }
14 }
15 class Student extends College{
16
17     String studentName;
18     String department;
19
20     public Student(String collegeName, String studentName, String depart) {
21         super(collegeName);
22         this.studentName = studentName;
23         this.department = depart;
24
25     }
26 }
```

```
28 public String toString(){
29     // return the details of the student
30     return "CollegeName : "+collegeName+"\nStudentName : "+studentName+"\nDepartment : "+department;
31 }
32 }
33 }
34 // class cse extends Student{
35 //     private String department;
36 // void course(){
37 //
38 //}
39 //}
40 class prog {
41 public static void main (String[] args) {
42     Student s1 = new Student("REC","Venkatesh","CSE");
43     s1.admitted();                                // invoke the admitted() method
44     System.out.println(s1.toString());
45 }
46 }
47 }
```

	Expected	Got	
✓	A student admitted in REC CollegeName : REC StudentName : Department : CSE	A student admitted in REC CollegeName : REC StudentName : Department : CSE	✓

Passed all tests! ✓

**Question 3**

Correct

Marked out of 5.00

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
```

**Answer:** (penalty regime: 0 %)

```
1 class Mobile{  
2     Mobile(){  
3         System.out.println("Basic Mobile is Manufactured");  
4     }  
5     // void basicMobile(){  
6     // }  
7 }  
8 class CamaraMobile extends Mobile{  
9     CamaraMobile(){  
10        super();  
11        System.out.println("Camera Mobile is Manufactured");  
12    }  
13    void newFeature(){  
14        System.out.println("Camera Mobile with 5MG px");  
15    }  
16 }  
17 class AndroidMobile extends CamaraMobile{  
18     AndroidMobile(){  
19        super();  
20        System.out.println("Android Mobile is Manufactured");  
21    }  
22    void androidMobile(){  
23        System.out.println("Touch Screen Mobile is Manufactured");  
24    }  
25 }  
26 }  
27 }  
28 public class prog{
```

```
29 public static void main(String A[]){
30     AndroidMoblile a = new AndroidMoblile();
31     a.newFeature();
32     a.androidMoblile();
33 }
34 }
```

	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-05-MCQ

Jump to...

Is Palindrome Number? ►

**Question 1**

Correct

Marked out of 5.00

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 2<sup>nd</sup> 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 25<sup>th</sup> 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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2
3 public class WordProcessor {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         String input = scanner.nextLine();
7         String[] words = input.split(":");
8
9         StringBuilder output = new StringBuilder();
10
11        for (String word : words) {
12            if (word.length() == 2) {
13                char firstChar = word.charAt(0);
14                char secondChar = word.charAt(1);
15
16                if (firstChar == secondChar) {
17                    output.append(firstChar);
18                } else {
19                    int position1 = firstChar - 'a' + 1;
20                    int position2 = secondChar - 'a' + 1;
21                    int maxPos = Math.max(position1, position2);
22                    int minPos = Math.min(position1, position2);
23                    int newPos = maxPos - minPos;
24                    char newChar = (char) ('a' + newPos - 1);
25                    output.append(newChar);
26                }
27            }
28        }
29        System.out.println(output.toString().toUpperCase());
30
31        scanner.close();
32    }
33 }
```

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

Passed all tests! ✓

**Question 2**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```
1 import java.util.Scanner;
2
3 public class WordProcessor {
```

```

4  public static void main(String[] args) {
5      Scanner scanner = new Scanner(System.in);
6      String inputString = scanner.nextLine();
7      int inputNumber = scanner.nextInt();
8
9      String processedString = processWords(inputString, inputNumber);
10     System.out.println(processedString);
11 }
12
13 public static String processWords(String inputString, int inputNumber) {
14     String[] words = inputString.split(" ");
15
16     if (words.length < 2 || inputNumber < 11 || inputNumber > 99) {
17         return "Invalid input";
18     }
19
20     int word1Index = inputNumber / 10 - 1;
21     int word2Index = inputNumber % 10 - 1;
22
23     if (word1Index >= words.length || word2Index >= words.length) {
24         return "Invalid word indices";
25     }
26
27     String word1 = words[word1Index];
28     String word2 = words[word2Index];
29
30     String processedWord1 = processWord(word1);
31     String processedWord2 = processWord(word2);
32
33     return processedWord1 + " " + processedWord2;
34 }
35
36 public static String processWord(String word) {
37     int middleIndex = word.length() / 2;
38     StringBuilder input = new StringBuilder();
39     StringBuilder input1 = new StringBuilder();
40     input.append(word.substring(0,middleIndex+1));
41     input1.append(word.substring(0,middleIndex));
42     if(word.length()%2==0){
43         return input1.reverse()+word.substring(middleIndex);
44     }
45     else{
46         return input.reverse()+word.substring(middleIndex);
47     }
48 }
49 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	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! ✓

**Question 3**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.HashSet;
2 import java.util.Scanner;
3 import java.util.Set;
4 import java.util.stream.Collectors;
5
6 public class StringProcessor {
7     public static void main(String[] args) {
8         Scanner scanner = new Scanner(System.in);
9         String input1 = scanner.nextLine();
10        String input2 = scanner.nextLine();
11        String result = processStrings(input1, input2);
12        System.out.println(result);
13    }
14    public static String processStrings(String input1, String input2) {
15        if ((input1 == null || input1.trim().isEmpty()) && (input2 == null || input2.trim().isEmpty())) {
16            return "null";
17        }
18        String concatenated = input1 + input2;
19        Set<Character> uniqueChars = new HashSet<>();
20        for (char c : concatenated.toCharArray()) {

```

```
21     if (c != ' ') {  
22         uniqueChars.add(c);  
23     }  
24     String sortedChars = uniqueChars.stream().map(String::valueOf).sorted((a, b) -> b.compareTo(a)).collect(Co  
25     return sortedChars;  
26 }  
27  
28 }  
29
```

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

Passed all tests! ✓

◀ Lab-06-MCQ

Jump to...

Return second word in Uppercase ►

**Question 1**

Correct

Marked out of 5.00

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.

**Answer:** (penalty regime: 0 %)

```
1 • interface RBI {
2     String parentBank = "RBI";
3     double rateOfInterest();
4     default void policyNote() {
5         System.out.println(parentBank + " has a new Policy issued in 2023");
6     }
7     static void regulations() {
8         System.out.println(parentBank + " has updated new regulations in 2024.");
9     }
10 }
11 class SBI implements RBI {
12     @Override
13     public double rateOfInterest() {
14         return 7.6;
15     }
16 }
17 class Karur implements RBI {
18     @Override
19     public double rateOfInterest() {
20         return 7.4;
21     }
22 }
23 public class BankPolicies {
24     public static void main(String[] args) {
25         RBI sbi = new SBI();
26         RBI karur = new Karur();
27         sbi.policyNote();
28         RBI.regulations();
29         System.out.println("SBI rate of interest: " + sbi.rateOfInterest() + " per annum.");
30     }
31 }
```

```
30 }     System.out.println("Karur rate of interest: " + karur.rateOfInterest() + " per annum.");
31 }
32 }
```

	<b>Test</b>	<b>Expected</b>	<b>Got</b>	
✓	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! ✓

**Question 2**

Correct

Marked out of 5.00

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:**

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

**For example:**

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

**Answer:** (penalty regime: 0 %)

```
1 import java.util.Scanner;
2 interface Playable {
3     void play();
4 }
5 class Football implements Playable {
6     String name;
7
8     public Football(String name) {
9         this.name = name;
10    }
11
12    @Override
13    public void play() {
14        System.out.println(name + " is Playing football");
15    }
16 }
17
18 class Volleyball implements Playable {
19     String name;
20
21     public Volleyball(String name) {
22         this.name = name;
23     }
24    @Override
25    public void play() {
26        System.out.println(name + " is Playing volleyball");
27    }
28 }
```

```

26     }
27 }
28 }
29 class Basketball implements Playable {
30     String name;
31
32 public Basketball(String name) {
33     this.name = name;
34 }
35
36 @Override
37 public void play() {
38     System.out.println(name + " is Playing basketball");
39 }
40 }
41 public class SportsGame {
42 public static void main(String[] args) {
43     Scanner scanner = new Scanner(System.in);
44     Playable[] players = new Playable[3];
45     String[] sports = {"Football", "Volleyball", "Basketball"};
46     for (int i = 0; i < players.length; i++) {
47         String name = scanner.nextLine();
48         switch (sports[i]) {
49             case "Football":
50                 players[i] = new Football(name);
51                 break;
52             case "Volleyball":

```

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

Passed all tests! ✓

**Question 3**

Correct

Marked out of 5.00

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!

**Answer:** (penalty regime: 0 %)

[Reset answer](#)

```
1 import java.util.Scanner;  
2 interface Sports {  
3     public void setHomeTeam(String name);  
4     public void setVisitingTeam(String name);  
5 }  
6 interface Football extends Sports {  
7     public void homeTeamScored(int points);  
8     public void visitingTeamScored(int points);  
9 }  
10  
11 }  
12 class College implements Football {  
13     String homeTeam;  
14     String visitingTeam;  
15  
16     public void setHomeTeam(String name){  
17         this.homeTeam=name;  
18     }  
19     public void setVisitingTeam(String name){  
20         this.visitingTeam=name;  
21     }  
22     public void homeTeamScored(int points){  
23         System.out.println(homeTeam+" "+points+" scored");  
24     }  
25     public void visitingTeamScored(int points){  
26         System.out.println(visitingTeam+" "+points+" scored");  
27     }  
28 }
```

```

28  public int winningTeam(int p1, int p2){
29      if(p1>p2)
30          return 1;
31      else if(p1<p2)
32          return 0;
33      else
34          return -1;
35  }
36 }
37 public class Main{
38     public static void main(String[] args){
39         Scanner sc= new Scanner(System.in);
40         String hname=sc.next();
41         String vteam=sc.next();
42         int hpoints=sc.nextInt();
43         int vpoints=sc.nextInt();
44         College s= new College();
45         s.setHomeTeam(hname);
46         s.setVisitingTeam(vteam);
47         s.homeTeamScored(hpoints);
48         s.visitingTeamScored(vpoints);
49         if((s.winningTeam(htpoints,vpoints))==-1){
50             System.out.println(hname+" is the winner!");
51         }
52         else if((s.winningTeam(htpoints,vpoints))==1){

```

	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! ✓

◀ Lab-07-MCQ

Jump to...

Generate series and find Nth element ►

**Question 1**

Correct

Marked out of 5.00

## 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.

**Answer:** (penalty regime: 0 %)

[Reset answer](#)

```
1 class FinalExample {
2
3     // Final variable
4     final int maxSpeed = 120;
5
6     // Final method
7     public final void displayMaxSpeed() {
8         System.out.println("The maximum speed is: " + maxSpeed + " km/h");
9     }
10 }
11
12 class SubClass extends FinalExample {
13
14     // public void displayMaxSpeed() {
15     //     System.out.println("Cannot override a final method");
16     // }
17
18     // You can create new methods here
19     public void showDetails() {
20         System.out.println("This is a subclass of FinalExample.");
21     }
22 }
23
24 class prog {
```

```
25 public static void main(String[] args) {  
26     FinalExample obj = new FinalExample();  
27     obj.displayMaxSpeed();  
28  
29     SubClass subObj = new SubClass();  
30     subObj.showDetails();  
31 }  
32 }
```

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

Passed all tests! ✓



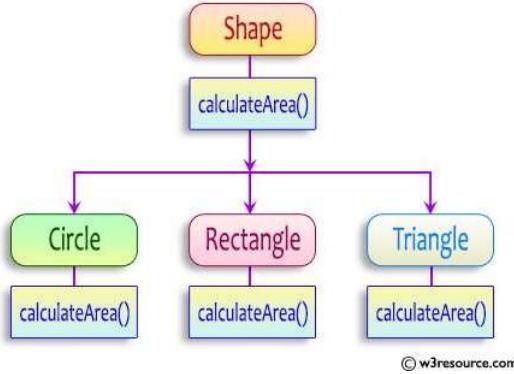
**Question 2**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.*;
2 abstract class Shape{
3     abstract void calculatearea();
4 }
5 class Circle extends Shape{
  
```

```

6   float rad;
7   Circle(float rad){
8       this.rad = rad;
9   }
10  void calculatearea(){
11      System.out.format("Area of a circle: %.2f\n",3.14159*rad*rad);
12  }
13 }
14 class Rectangle extends Shape{
15     float l;
16     float br;
17     Rectangle(float l,float br){
18         this.l = l;
19         this.br = br;
20     }
21     void calculatearea(){
22         System.out.format("Area of a Rectangle: %.2f\n", (l*br));
23     }
24 }
25 }
26 class Triangle extends Shape{
27     float ba;
28     float h;
29     Triangle(float ba ,float h){
30         this.ba = ba;
31         this.h = h;
32     }
33     void calculatearea(){
34         System.out.format("Area of a Triangle: %.2f",0.5*ba*h);
35     }
36 }
37 }
38 class prog{
39     public static void main (String are[]){
40         Scanner scan = new Scanner(System.in);
41         float rad = scan.nextFloat();
42         float l = scan.nextFloat();
43         float br = scan.nextFloat();
44         float ba = scan.nextFloat();
45         float h = scan.nextFloat();
46         Circle c = new Circle(rad);
47         Rectangle r = new Rectangle(l,br);
48         Triangle t = new Triangle(ba,h);
49         c.calculatearea();
50         r.calculatearea();
51         t.calculatearea();
52     }

```

	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! ✓

**Question 3**

Correct

Marked out of 5.00

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 andlast 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 andlast 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

**Answer:** (penalty regime: 0 %)

```

1 import java.util.*;
2 class prog{
3     public static void main(String ae[]){
4         Scanner scan = new Scanner(System.in);
5         int n = scan.nextInt();
6         String arr[] = new String[n];
7         scan.nextLine();
8         String str = scan.nextLine();
9         String temp = "";
10        int j=0;
11        int l=str.length();

```

```

12   for(int i = 0;i<l;i++){
13     if(str.charAt(i)==' '){
14       arr[j] = temp;
15       temp = "";
16       j++;
17     }
18   } else{
19     temp +=str.charAt(i);
20   }
21 }
22 arr[j] = temp;
23 String s = "";
24 char [] cha ={'a','A','e','E','i','I','o','O','u','U','u'};
25 for(int i=0;i<n;i++){
26   int c=0;
27   char [] ar = arr[i].toCharArray();
28   char ch1 = ar[0];
29   char ch2 = ar[ar.length -1];
30   for(char k : cha){
31     if(k==ch1){
32       c++;
33     }
34     if(k==ch2){
35       c++;
36     }
37   }
38   if(c==2){
39     s+=arr[i];
40   }
41 }
42 if(s==""){
43   System.out.print("no matches found");
44 }
45 else{
46   System.out.print(s.toLowerCase());
47 }
48
49
50
51
52

```

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

Passed all tests! ✓

◀ Lab-08-MCQ

Jump to...

FindStringCode ►

**Question 1**

Correct

Marked out of 5.00

Write a Java program to handle `ArithmaticException` 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.ArithmaticException: / 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.ArithmaticException: / by zero I am always executed

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n=sc.nextInt();
8         int[] arr=new int[n];
9         for(int i=0;i<n;i++)
10        {
11            arr[i]=sc.nextInt();
12        }
13        try
14        {
15            int a=arr[0]/arr[1];
16            int b=arr[n];
17        }
18        catch(ArithmaticException ae)
19        {
20            System.out.println(ae.toString());
21        }
22        catch(ArrayIndexOutOfBoundsException e)
23        {
24            System.out.println(e.toString());
25        }
26        finally
27        {
28            System.out.println("I am always executed");
29        }

```

```
30 |
31 }     }
32 }
```

	<b>Test</b>	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	1	6 1 0 4 1 2 8	java.lang.ArithmetricException: / by zero I am always executed	java.lang.ArithmetricException: / 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! ✓

**Question 2**

Correct

Marked out of 5.00

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.
```

**Answer:** (penalty regime: 0 %)

[Reset answer](#)

```
1 class prog {  
2     public static void main(String[] args) {  
3         int n = 82;  
4         trynumber(n);  
5         n = 37;  
6         // call the trynumber(n);  
7         trynumber(n);  
8     }  
9  
10 }  
11  
12 public static void trynumber(int n) {  
13     try {  
14         //call the checkEvenNumber()  
15         checkEvenNumber(n);  
16         System.out.println(n + " is even.");  
17     } catch (Exception e) {  
18         System.out.println("Error: " + e.getMessage());  
19     }  
20 }  
21  
22 public static void checkEvenNumber(int number) throws Exception {  
23     if (number % 2 != 0) {  
24         throw new Exception(number+" is odd.");  
25     }  
26 }  
27 }  
28 }
```

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

Passed all tests! ✓

**Question 3**

Correct

Marked out of 5.00

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

**Answer:** (penalty regime: 0 %)

[Reset answer](#)

```
1 import java.util.Scanner;
2 import java.util.InputMismatchException;
3 class prog {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         int length = sc.nextInt();
7         // create an array to save user input
8         int[] name = new int[length];
9         int sum=0;//save the total sum of the array.
10
11
12     /* Define try-catch block to save user input in the array "name"
13     If there is an exception then catch the exception otherwise print
14     the total sum of the array. */
15     try
16     {
17         for(int i=0;i<length;i++)
18         {
19             name[i]=sc.nextInt();
20             sum+=name[i];
21
22         }
23         System.out.println(sum);
24
25     }
26     catch(InputMismatchException ae )
```

```
27  {
28      System.out.println("You entered bad data.");
29  }
30
31
32
33
34
35
36
37  }
38 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	3 5 2 1	8	8	✓
✓	2 1 g	You entered bad data.	You entered bad data.	✓

Passed all tests! ✓

[◀ Lab-09-MCQ](#)

Jump to...

[The "Nambiar Number" Generator ►](#)

**Question 1**

Correct

Marked out of 1.00

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.

**Answer:** (penalty regime: 0 %)

```
1 import java.util.*;
2 public class Solution
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n=sc.nextInt();
8         ArrayList<Integer> arr=new ArrayList<Integer>(n);
9         while(sc.hasNextInt())
10        {
11             int a=sc.nextInt();
12             arr.add(a);
13         }
14
15         int b=arr.size()-1;
16         System.out.println("ArrayList: "+arr);
17         System.out.println("First : "+arr.get(0)+" , Last : "+arr.get(b));
18     }
19 }
```

	<b>Test</b>	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	1	6 30 20 40 50 10 80	ArrayList: [30, 20, 40, 50, 10, 80] First : 30, Last : 80	ArrayList: [30, 20, 40, 50, 10, 80] First : 30, Last : 80	✓
✓	2	4 5 15 25 35	ArrayList: [5, 15, 25, 35] First : 5, Last : 35	ArrayList: [5, 15, 25, 35] First : 5, Last : 35	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of 1.00

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.

**Answer:** (penalty regime: 0 %)

[Reset answer](#)

```
1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 public class Prog {
5
6     public static void main(String[] args)
7     {
8         Scanner sc= new Scanner(System.in);
9         int n = sc.nextInt();
10
11     ArrayList<Integer> list = new ArrayList<Integer>();
12
13     for(int i = 0; i<n;i++)
14     list.add(sc.nextInt());
15
16     // printing initial value ArrayList
17     System.out.println("ArrayList: " + list);
18
19 //Replacing the element at index 1 with 100
20 list.set(1,100);
21
22
23 //Getting the index of first occurrence of 100
24 System.out.println("Index of 100 = "+list.indexOf(100) );
25
26 //Getting the index of last occurrence of 100
27 System.out.println("LastIndex of 100 = "+ list.lastIndexOf(100) );
28 // Check whether 200 is in the list or not
29 System.out.println(list.contains(200) ); //Output : false
30 // Print ArrayList size
31 System.out.println("Size Of ArrayList = "+ list.size() );
32 //Inserting 500 at index 1
33 list.add(1,500) ; // code here
34 //Removing an element from position 3
35 list.remove(3); // code here
36 System.out.print("ArrayList: " + list);
37 }
38 }
```

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

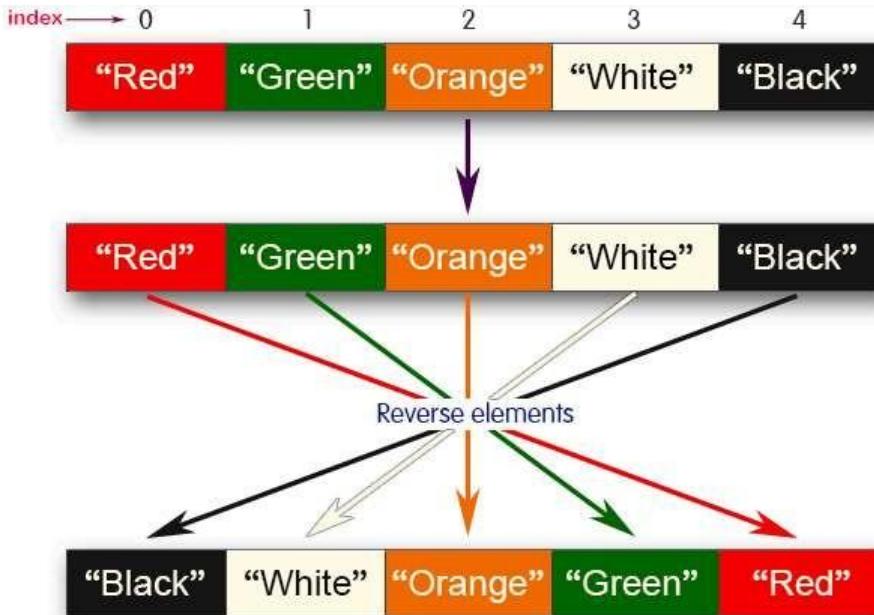
Passed all tests! ✓

**Question 3**

Correct

Marked out of 1.00

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]

**Answer:** (penalty regime: 0 %)

```

1 import java.util.*;
2 public class Reverse
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n=sc.nextInt();
8         ArrayList<String> list=new ArrayList<String>(n);
9         for(int i=0;i<n;i++)
10        {
11            String str=sc.next();
12            list.add(str);
13        }
14        ArrayList<String> list1=new ArrayList<String>(n);
15        for(int i=n-1;i>=0;i--)
16        {
17            String str1=list.get(i);
18            list1.add(str1);
19        }
20        System.out.println("List before reversing :");
21        System.out.println(list);
22        System.out.println("List after reversing :");
23        System.out.println(list1);
24    }
}

```

	<b>Test</b>	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	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-10-MCQ

Jump to...

Lab-11-MCQ ►

**Question 1**

Correct

Marked out of 1.00

**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.

**Answer:** (penalty regime: 0 %)

[Reset answer](#)

```

1 import java.util.HashSet;
2 import java.util.Scanner;
3 class prog
4 {
5     public static void main(String[] args)
6     {
7         Scanner sc= new Scanner(System.in);
8         int n = sc.nextInt();
9         // Create a HashSet object called numbers
10        HashSet<Integer> numbers= new HashSet<Integer>(n);
11        // Add values to the set
12        for(int i=0;i<n;i++)
13        {
14            numbers.add(sc.nextInt());
15        }
16        int skey=sc.nextInt();
17        // Show which numbers between 1 and 10 are in the set
18        if(numbers.contains(skey))
19        {
20            System.out.println(skey+ " was found in the set.");
21        }
22    }
23 }
```

```
21
22     }
23     else
24     {
25         System.out.println(skey + " was not found in the set.");
26     }
27 }
28 }
```

	Test	Input	Expected	Got	
✓	1	5 90 56 45 78 25 78	78 was found in the set.	78 was found in the set.	✓
✓	2	3 -1 2 4 5	5 was not found in the set.	5 was not found in the set.	✓

Passed all tests! ✓

**Question 2**

Correct

Marked out of 1.00

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

**Answer:** (penalty regime: 0 %)

```
1 import java.util.HashSet;
2 import java.util.Scanner;
3 import java.util.ArrayList;
4 public class Main
5 {
6     public static void main(String[] args)
7     {
8         Scanner scanner = new Scanner(System.in);
9         HashSet<String> set1 = new HashSet<>();
10        HashSet<String> set2 = new HashSet<>();
11        int numElementsSet1 = scanner.nextInt();
12        scanner.nextLine();
13        for (int i = 0; i < numElementsSet1; i++) {
14            String num = scanner.nextLine();
15            set1.add(num);
16        }
17        int numElementsSet2 = scanner.nextInt();
18        scanner.nextLine();
19        for (int i = 0; i < numElementsSet2; i++)
20        {
21            String num = scanner.nextLine();
22            set2.add(num);
23        }
24        set1.retainAll(set2);
25        ArrayList<String> list = new ArrayList<>(set1);
26        for (int i = 0; i < list.size(); i++)
27        {
28            System.out.println(list.get(i));
29        }
30    }
31}
```

```
29 }  
30 }  
31 }  
32 }
```

	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! ✓

**Question 3**

Correct

Marked out of 1.00

## 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

**Answer:** (penalty regime: 0 %)[Reset answer](#)

```

1 import java.util.HashMap;
2 import java.util.Map.Entry;
3 import java.util.Set;
4 import java.util.Scanner;
5 class prog
6 {
7     public static void main(String[] args)
8     {
9         //Creating HashMap with default initial capacity and load factor
10        HashMap<String, Integer> map = new HashMap<String, Integer>();
11
12        String name;
13        int num;
14        Scanner sc= new Scanner(System.in);
15        int n=sc.nextInt();
16        for(int i =0;i<n;i++)
17        {
18            name=sc.next();
19            num= sc.nextInt();
20            map.put(name,num);
21        }
22
23        //Printing key-value pairs
24
25
26        Set<Entry<String, Integer>> entrySet = map.entrySet();
27
28        for (Entry<String, Integer> entry : entrySet)
29        {
30            System.out.println(entry.getKey()+" : "+entry.getValue());
31        }
32        System.out.println("-----");
33        //Creating another HashMap
34
35        HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();
36
37        //Inserting key-value pairs to anotherMap using put() method
38
39        anotherMap.put("SIX", 6);
40
41        anotherMap.put("SEVEN", 7);
42
43        //Inserting key-value pairs of map to anotherMap using putAll() method
44
45        anotherMap.putAll      ( map    ); // code here
46
47        //Printing key-value pairs of anotherMap
48
49        entrySet = anotherMap.entrySet();

```

```
50
51     for (Entry<String, Integer> entry : entrySet)
52 {
```

	<b>Test</b>	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	1	3 ONE 1 TWO 2 THREE 3	ONE : 1 TWO : 2 THREE : 3 <hr/> SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	ONE : 1 TWO : 2 THREE : 3 <hr/> SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	✓

Passed all tests! ✓

◀ Lab-11-MCQ

Jump to...

TreeSet example ►

### Question 1

Correct

Marked out of 5.00

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 (000000000000000000000000000000).

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: 000010000000000000000000000010000000000001000000000000100000000000010000000000001

The decoded string (original word) will be: WIPRO

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

For example:

Input	Result
010010001	ZYX
000010000000000000000000100000000000100000000010000000000000001	WIPRO

**Answer:** (penalty regime: 0 %)

```
1 import java.util.Scanner;
2 public class BinaryStringDecoder
3 {
4     public static void main(String[] args)
5     {
6         Scanner scanner = new Scanner(System.in);
7         String encodedString = scanner.nextLine();
8         String[] parts = encodedString.split("1");
9         StringBuilder decodedString = new StringBuilder();
10        for (String part : parts) {
11            int zeroCount = part.length();
12            if (zeroCount >= 1 && zeroCount <= 26) {
13                char decodedChar = (char) ('Z' - zeroCount + 1);
14                decodedString.append(decodedChar);
15            }
16        }
17        System.out.println(decodedString.toString());
18    }
19 }
```

20 |

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	010010001	ZYX	ZYX	✓
✓	0000100000000000000000001000000000001000000000010000000000001	WIPRO	WIPRO	✓

Passed all tests! ✓



**Question 2**

Correct

Marked out of 5.00

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	8
b c	

Answer: (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class ArrayCommonElementSum {
3     public static void main(String[] args) {
4         Scanner scanner = new Scanner(System.in);
5         String[] array1 = scanner.nextLine().split(" ");
6         String[] array2 = scanner.nextLine().split(" ");
7         int asciiSum = 0;
8         for (String element : array2) {
9             for (String elem1 : array1) {
10                 if (element.equals(elem1)) {
11                     asciiSum += element.charAt(0);
12                     break;
13                 }
14             }
15         }
16         int singleDigitSum = asciiSum % 9;
17         if (singleDigitSum == 0 && asciiSum > 0) {
18             singleDigitSum = 9;
19         }
20         System.out.println(singleDigitSum);
21     }
22 }
```

zz | j

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

Passed all tests! ✓

/

**Question 3**

Correct

Marked out of 5.00

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 SeigOlhoncet 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 Seigolonhct 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 Seigolonhct Erolagnab
Wipro Technologies, Bangalore 1	Orpiw ,seigolonhceT Erolagnab

**Answer:** (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class WordReversal
3 {
4     public static String reverseWords(String sentence, int caseOption) {
5         String[] words = sentence.split(" ");
6         StringBuilder modifiedSentence = new StringBuilder();
7         for (String word : words) {
8             String reversedWord = reverseWord(word, caseOption);
9             if (modifiedSentence.length() > 0) {

```

```

10         modifiedSentence.append(" ");
11     }
12 }
13
14 return modifiedSentence.toString();
15 }
16
17 private static String reverseWord(String word, int caseOption) {
18     String reversedWord = new StringBuilder(word).reverse().toString();
19     if (caseOption == 0) {
20         return reversedWord;
21     } else if (caseOption == 1) {
22         StringBuilder casePreservedWord = new StringBuilder(reversedWord);
23
24     for (int i = 0; i < word.length(); i++) {
25         char originalChar = word.charAt(i);
26         if (Character.isUpperCase(originalChar)) {
27             casePreservedWord.setCharAt(i, Character.toUpperCase(casePreservedWord.charAt(i)));
28         } else if (Character.isLowerCase(originalChar)) {
29             casePreservedWord.setCharAt(i, Character.toLowerCase(casePreservedWord.charAt(i)));
30         }
31     }
32     return casePreservedWord.toString();
33 }
34 return word;
35 }
36
37 public static void main(String[] args) {
38     Scanner scanner = new Scanner(System.in);
39     String sentence = scanner.nextLine();
40     int caseOption = scanner.nextInt();
41     String result = reverseWords(sentence, caseOption);
42     System.out.println(result);
43 }
44 }

```

	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! ✓

[◀ Lab-12-MCQ](#)

Jump to...

Identify possible words ►

## **HOTEL MANAGEMENT SYSTEM**

**CS23333 – Object Oriented Programming using Java Project Report**

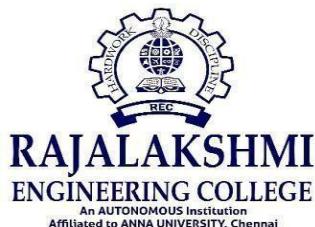
*Submitted by*

**SRIMAN VIYASEN SJ - 231001208**

**THARUN V - 231001231**

**VISHAL S - 231001248**

**BACHELOR OF TECHNOLOGY  
IN  
INFORMATION TECHNOLOGY**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**RAJALAKSHMI ENGINEERING COLLEGE**

**NOVEMBER-2024**

## **BONAFIDE CERTIFICATE**

Certified that this project titled “HOTEL MANAGEMENT SYSTEM” is the bonafide work of “**SRIMAN VIYASEN SJ (231001208), THARUN V (231001231), VISHAL S (231001248)**” who carried out the project work under my supervision.

### **SIGNATURE**

**Dr.P.Valarmathie**

**HEAD OF THE DEPARTMENT**

**Information Technology,**

**Rajalakshmi Engineering College**

**(Autonomous),**

**Thandalam,Chennai - 602 105**

### **SIGNATURE**

**Mr.K.E.Narayana**

**Assistant Professor,**

**Information Technology,**

**Rajalakshmi Engineering College**

**(Autonomous),**

**Thandalam,Chennai - 602 105**

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

**INTERNAL EXAMINAR**

**EXTERNAL EXAMINAR**

## **TABLE OF CONTENTS**

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
	List of Tables	5
<b>1</b>	1. Abstract	6
	2. Introduction	6
	3. Purpose	6
	4. Scope of Project	7
	5. Software Requirement Specification	7
<b>2</b>	<b>System flow Diagrams</b>	12
	1. Entity-relationship Diagrams	13
	2. Data Flow Diagram	13
<b>3</b>	<b>Module Description</b>	14
<b>4</b>	1. Design	15
	2. Database Design	18
	3. Code	19
<b>5</b>	<b>Conclusion</b>	30
<b>6</b>	<b>Reference</b>	30

## **1. Abstract**

The Hotel Management System is a Java and MySQL-based application designed to simplify hotel operations. It automates processes like room booking, check-in/out, customer management, and billing. The system ensures data accuracy, efficiency, and user-friendly interaction, enhancing both staff productivity and customer satisfaction. This report details its design, implementation, and practical applications in the hospitality sector. The system streamlines daily hotel operations, reducing manual effort and minimizing errors in guest data and financial transactions. It is scalable, allowing future upgrades or integration with other services like payment gateways and reporting tools, supporting the growth of the hotel business.

## **2. Introduction**

The Hotel Management System (HMS) is a software solution built using Java and MySQL to automate and streamline hotel operations. It handles tasks such as room reservations, guest check-ins/check-outs, billing, and customer management. By replacing manual processes, the system improves efficiency, reduces errors, and enhances customer satisfaction. This report outlines the system's design, functionality, and its application in the hospitality industry.

## **3. Purpose**

The purpose of the Hotel Management System is to automate and optimize hotel operations, providing a seamless experience for both guests and hotel staff. By leveraging Java and MySQL, the system simplifies tasks such as room booking, check-in/check-out, billing, and customer data management. The goal is to enhance operational efficiency, reduce errors, and improve overall customer satisfaction, making hotel management processes faster and more reliable.

## **4. Scope of the Project:**

The Hotel Management System aims to automate core hotel operations, including room management, reservations, check-ins, check-outs, and billing. It allows guests to book rooms, modify or cancel reservations, and provides an efficient check-in/check-out process. Billing and payment functionalities are integrated to generate invoices and manage payments securely. The system also stores guest information, enabling personalized services and repeat bookings.

Using MySQL for data storage, the system ensures secure and efficient management of hotel data. It supports role-based user access, allowing staff to perform tasks according to their responsibilities. The system also generates reports on occupancy, revenue, and guest history to assist in decision-making. Designed for scalability, it can easily accommodate future expansions and additional features as the hotel's needs grow.

## **5. Software Requirement Specification**

### **Introduction**

The Hotel Management System requires specific software tools to ensure efficient development, deployment, and performance. It automates hotel operations such as room management, reservations, billing, and customer data handling. This section outlines the necessary software components for building and maintaining the system.

### **Product Scope**

The system will be developed using **Java** for application logic and **MySQL** for secure data storage. It will run on standard desktop or web platforms, requiring **Java Runtime Environment (JRE)** and **MySQL Server**. Development will be done using **IDEs** like **Eclipse** or **IntelliJ IDEA**. If a web-based interface is used, a **web server** like **Apache Tomcat** may be required for deployment.

## Overall Description

The Hotel Management System automates hotel operations like room bookings, guest check-ins/out, billing, and customer data management using **Java** and **MySQL**. It includes modules for room availability, reservations, guest management, and billing. The system ensures efficiency with real-time data updates and secure payment handling, while its user-friendly interface streamlines operations. The system is scalable for future expansion and feature integration.

## Product Perspective

The Hotel Management System is a standalone application that integrates with **MySQL** for secure data storage. It automates hotel functions like bookings, check-ins, and billing, replacing manual processes. The system is flexible, supporting both desktop and web platforms, and is designed for easy future enhancements, such as advanced reporting or third-party integrations. It improves operational efficiency and customer satisfaction.

## Product Functionality

- a) **Room and Reservation Management:** Manages room availability, bookings, and modifications.
- b) **Check-in/Check-out and Billing:** Automates check-ins, check-outs, generates bills, and processes payments.
- c) **Customer Data Management:** Stores and retrieves guest information for future visits and personalized services.
- d) **User Access Control:** Provides role-based access for staff to secure different modules.
- e) **Real-time Data Updates:** Ensures up-to-date information on room status, bookings, and guest details.
- f) **Scalability:** Designed to accommodate future feature expansions and integrations.

## **Operating Environment**

### **Hardware Requirements**

- Processor: Intel i3 or higher (or equivalent AMD processor)
- Operating System: Windows 8,10, 11
- Processor Speed: 2.0 GHz
- RAM: 4GB
- Hard Disk: 500GB

### **Software Requirements**

- Database: MySQL
- Frontend: JSP
- Technology: Java (JDBC)

### **Hardware Interface**

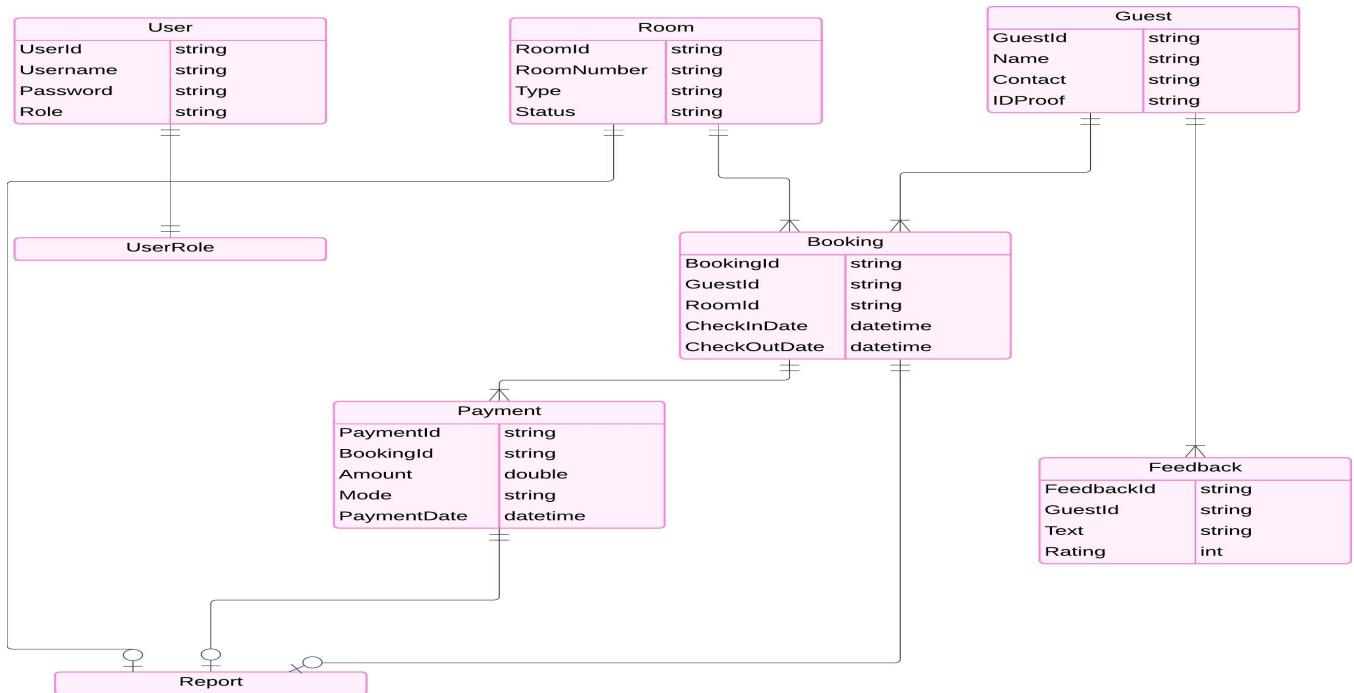
- a) Desktop or laptop computers with keyboard, mouse, and monitor for user workstations.
- b) Dedicated server or cloud infrastructure with at least 4GB RAM and 100GB storage for hosting the system.

### **Software Interface**

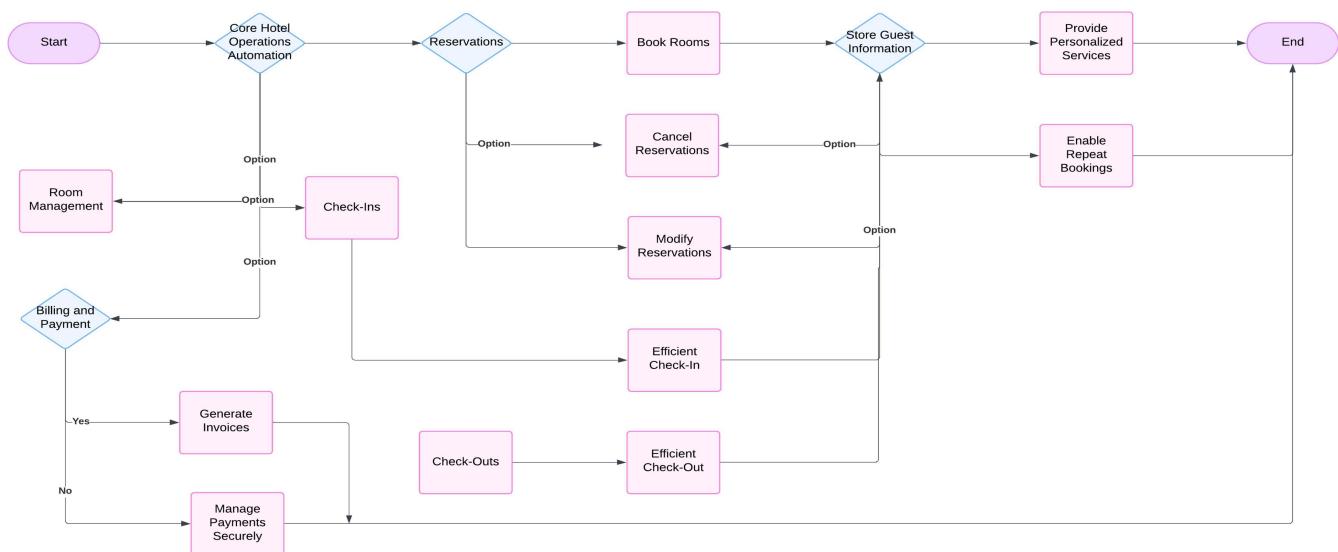
- a) MS-Windows Operating System .
- b) JSP for designing the front end .
- c) java for the backend & MYSQL as database.
- d) Platform: Java Language.

# CHAPTER 2

## 1. Entity Relationship Diagram:



## 2. Flow Diagram



### **3. MODULE DESCRIPTION**

#### **1) Authentication and Authorization**

- **Description:** Handles user login and role-based access.
- **Features:**
  - Secure login for admin, staff, and guests.
  - Password encryption and storage.
  - Role-based access control (e.g., admin vs staff).
- **Database Tables:**
  - users: Stores user credentials and roles.
- **JDBC Operations:**
  - Validate credentials during login.
  - Add or update user roles.

#### **2) Guest Management**

- **Description:** Manages information about guests staying in the hotel.
- **Features:**
  - Add, view, update, and delete guest records.
  - Search guests by name, phone number, or booking ID.
- **Database Tables:**
  - guests: Stores guest details such as name, contact, and ID proof.
- **JDBC Operations:**
  - Insert and retrieve guest details.
  - Update guest contact information.

#### **3) Room Management**

- **Description:** Maintains a database of available, booked, and occupied rooms.
- **Features:**
  - Add and manage room types (single, double, suite).
  - Update room status (available, occupied, maintenance).
  - Search for available rooms by type or date.
- **Database Tables:**
  - rooms: Stores room details such as room number, type,

and status.

- **JDBC Operations:**

- Fetch available rooms for a given date range.
- Update room status after booking or checkout.

#### 4) Booking Management

- **Description:** Facilitates room bookings and ensures availability.

- **Features:**

- Create new bookings for guests.
- Update or cancel existing bookings.
- Generate booking confirmation with details.

- **Database Tables:**

- Bookings: Stores booking details such as guest ID, room ID, and check-in/check-out dates.

- **JDBC Operations:**

- Insert booking records.
- Fetch booking history for a specific guest.

#### 5) Payment and Billing

- **Description:** Automates the billing process for guests.

- **Features:**

- Calculate room charges based on stay duration.
- Add additional charges (e.g., meals, spa, laundry).
- Generate and print invoices.

- **Database Tables:**

- Payments: Stores payment details such as amount, mode, and date.

- **JDBC Operations:**

- Insert payment records.
- Generate invoice details using data from bookings and payments.

#### 6) Feedback and Reviews

- **Description:** Allows guests to leave feedback about their stay.

- **Features:**

- Record and view guest feedback.
- Search feedback by guest or stay date.

- **Database Tables:**
  - feedback: Stores feedback text and ratings.
- **JDBC Operations:**
  - Insert and retrieve feedback records.

## 7) Reports and Analytics

- **Description:** Generates insights and summaries for the hotel management team.
- **Features:**
  - Generate occupancy reports by date range.
  - Analyze revenue trends.
  - View top-rated rooms or services.
- **Database Tables:**
  - Uses bookings, rooms, and payments tables.
- **JDBC Operations:**
  - Fetch aggregated data using SQL queries.
  - Display data in tabular or graphical formats.

## 8) System Administration

- **Description:** Handles maintenance and configuration tasks for the system.
- **Features:**
  - Backup and restore database.
  - Configure pricing for rooms and services.
  - Manage staff accounts and permissions.
- **Database Tables:**
  - Various administrative tables, e.g., pricing.
- **JDBC Operations:**
  - Update pricing for rooms/services.
  - Add or remove staff accounts.

## 4.1 Design



Figure 4.1.1 Login Page

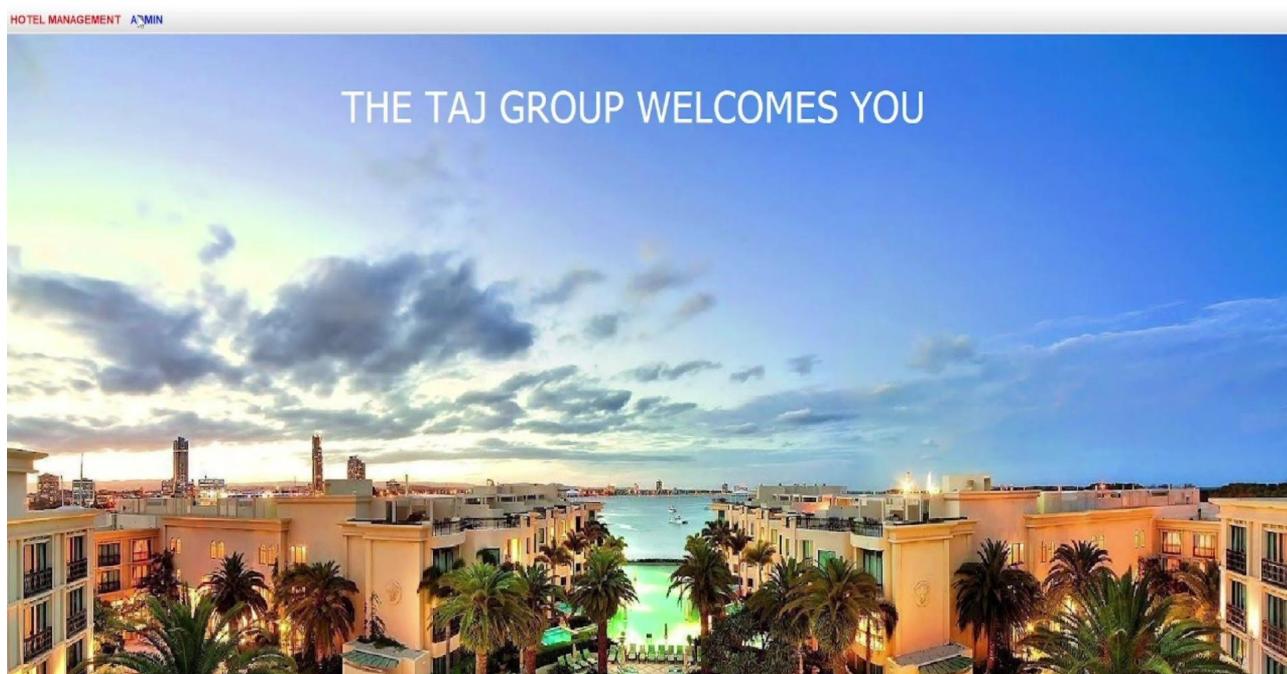


Figure 4.1.2 Home page

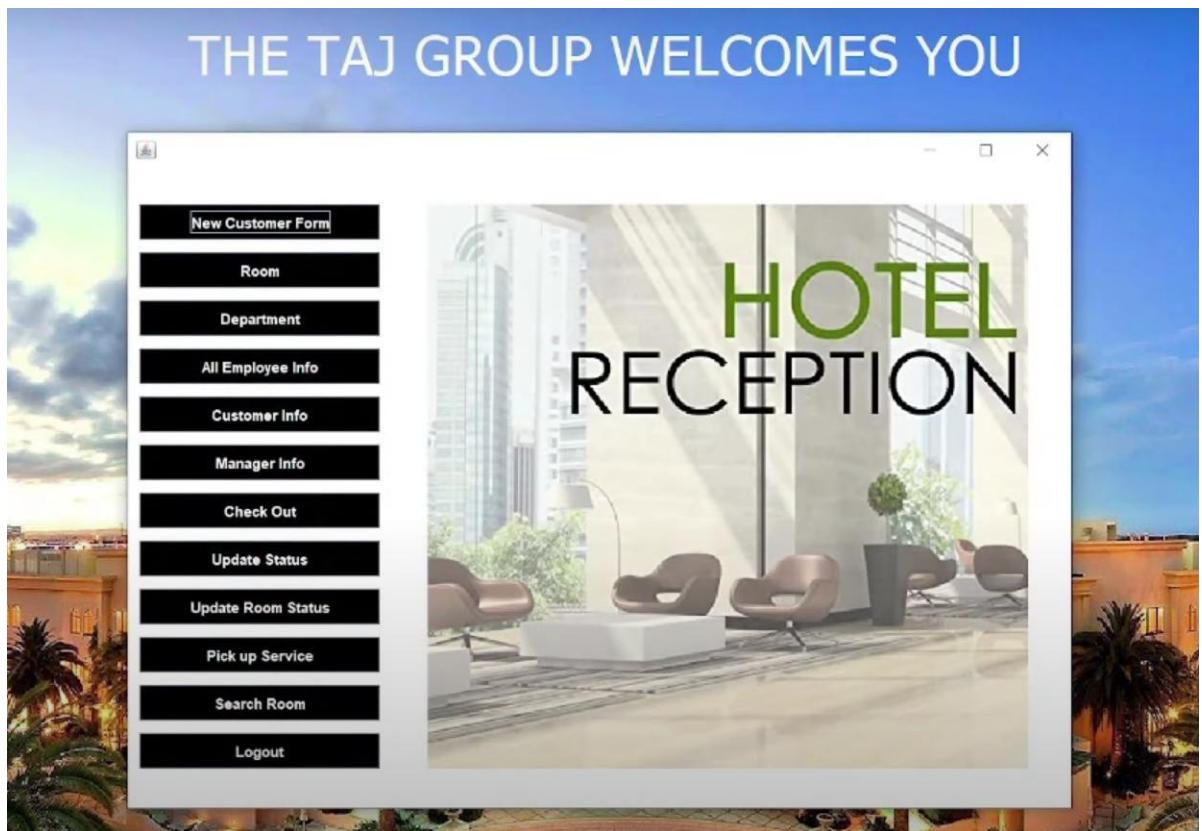


Figure 4.1.3 Hotel Reception Interface

A screenshot of a Windows application window titled "THE TAJ GROUP WELCOMES YOU". The main title is at the top left. Inside the window, there is a form titled "ADD EMPLOYEE DETAILS" on the right side. The form fields include: NAME (text input), AGE (text input), GENDER (radio buttons for Male and Female), JOB (dropdown menu showing "Front Desk Clerks"), SALARY (text input), PHONE (text input), AADHAR (text input), and EMAIL (text input). At the bottom of the form is a "Submit" button. To the right of the form, there are four cartoon illustrations of hotel staff: a woman holding a red folder, a man holding a silver platter, a woman holding a tray of food, and a man holding a suitcase. The background of the application window shows a scenic view of a city skyline and palm trees under a blue sky.

Figure 4.1.4 New Employee Details Adding Page

## 4.2 Database Design

The database design for a Hotel Management System project using JDBC involves creating a structured schema to efficiently manage hotel operations, including guest bookings, room availability, staff details, billing, and services. The core database typically includes tables such as **Guests**, **Rooms**, **Reservations**, **Staff**, and **Payments**. Each table is designed with unique attributes; for instance, the *Guests* table may have fields like `guest_id`, `name`, `contact_info`, and `address`, while the *Rooms* table includes `room_id`, `room_type`, `price_per_night`, and `availability_status`. Relationships between tables are established using primary and foreign keys, enabling seamless integration of data. JDBC (Java Database Connectivity) acts as the bridge between the Java application and the relational database, allowing operations like querying room availability, updating bookings, and generating invoices in real time. The use of normalization ensures the elimination of data redundancy, while indexing key fields enhances query performance, providing a robust and scalable foundation for the system.

### Database Code :

```
create database hotelmanagementsystem;  
show databases;  
use hotelmanagementsystem;  
create table login(username varchar(25), password varchar(25));  
insert into login values('admin', '12345');  
select * from login;  
  
create table employee(name varchar(25), age varchar(10), gender varchar(15), job varchar(30),  
salary varchar(15), phone varchar(15), email varchar(40), aadhar varchar(20));  
  
describe employee;  
  
select * from employee;  
  
create table room(roomnumber varchar(10), availability varchar(20), cleaning_status varchar(20),  
price varchar(20), bed_type varchar(20));
```

```
select * from room;

update room set availability = 'Available' where roomnumber = '101';

create table driver(name varchar(20), age varchar(10), gender varchar(15), company varchar(20),
branch varchar(20), available varchar(20), location varchar(40));

select * from driver;

ALTER TABLE driver RENAME COLUMN branch TO brand;

create table customer(document varchar(20), number varchar(30), name varchar(30), gender
varchar(15), country varchar(20), room varchar(10), checkintime varchar(80), deposit
varchar(20));

select * from customer;

create table department(department varchar(30), budget varchar(30));

insert into department values('Front Office','500000');
insert into department values('Housekeeping', '40000');
insert into department values('Food and Beverage', '23000');
insert into department values('Kitchen or Food Production', '540000');
insert into department values('Security', '320000');

select * from department;
```

### 4.3 IMPLEMENTATION (CODE)

```
package hotel.management.system;

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

public class HotelManagementSystem extends JFrame implements

    ActionListener{JLabel l1;
    JButton b1;

    public HotelManagementSystem() {

        setSize(1366,430);
        setLayout(null);
        setLocation(300,300);

        l1 = new JLabel("");
        b1 = new JButton("Next");

        b1.setBackground(Color.WHITE);
        b1.setForeground(Color.BLACK);

        ImageIcon i1 = new
        ImageIcon(ClassLoader.getSystemResource("hotel/management/system/icons/first.jpg"));
        Image i3 = i1.getImage().getScaledInstance(1366, 390,Image.SCALE_DEFAULT);
        ImageIcon i2 = new ImageIcon(i3);
        l1 = new JLabel(i2);

        JLabel lid=new JLabel("HOTEL MANAGEMENT SYSTEM");
        lid.setBounds(30,300,1500,100);
        lid.setFont(new Font("serif",Font.PLAIN,70));
        lid.setForeground(Color.red);
        l1.add(lid);

        b1.setBounds(1170,325,150,50);
        l1.setBounds(0, 0, 1366, 390);

        l1.add(b1);
        add(l1);

        b1.addActionListener(this);
        setVisible(true);

        while(true){
            lid.setVisible(false); // lid = j label
```

```

        try{
            Thread.sleep(500); //1000 = 1 second
        }catch(Exception e){}
            lid.setVisible(true);
        try{
            Thread.sleep(500);
        }catch(Exception e){}
    }
}

public void actionPerformed(ActionEvent
ae){new Login().setVisible(true);
this.setVisible(false);

}

public static void main(String[] args) {
    HotelManagementSystem window = new HotelManagementSystem();
    window.setVisible(true);
}
}

```

#### **4.4 Database Connectivity Code**

```

package hotel.management.system;

import java.sql.*;

public class
conn{Connecti
on c; Statement
s; public
conn(){
try{
    Class.forName("com.mysql.jdbc.Driver");
    c=DriverManager.getConnection("jdbc:mysql://hms","root","");
    s=c.createStatement();

}catch(Exception
    e){ System.out.println(e);
}
}
}

```

## **5. CONCLUSION**

The Hotel Management System project implemented using JDBC provides an efficient and robust solution for managing hotel operations. By leveraging the power of Java and JDBC for database interaction, this system streamlines various processes, including room booking, customer management, billing, and staff coordination.

The integration of a relational database ensures data consistency, integrity, and security, while the dynamic capabilities of JDBC enable seamless interaction between the application and the database. This project demonstrates how modern programming techniques and database technologies can come together to enhance operational efficiency and deliver a better user experience.

Future enhancements could include the addition of advanced features such as real-time room availability tracking, integration with online payment gateways, and support for multi-language interfaces to make the system more versatile and user-friendly. Overall, the project serves as a practical and scalable solution for managing the complex operations of a hotel.

## 6. REFERENCES

**1 "Java: The Complete Reference" by Herbert Schildt**

- Covers JDBC in detail, with examples on database connectivity.
- Publisher: McGraw Hill.

**2 "Core Java Volume I - Fundamentals" by Cay S. Horstmann and Gary Cornell**

- Comprehensive resource for understanding Java programming and JDBC concepts.
- Publisher: Pearson Education.

**3 "Head First Java" by Kathy Sierra and Bert Bates**

- Beginner-friendly introduction to Java and JDBC integration.
- Publisher: O'Reilly Media.