

RAJALAKSHMI ENGINEERING COLLEGE
RAJALAKSHMI NAGAR, THANDALAM – 602
105



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

CS23333 Object Oriented Programming Using Java

Laboratory Record Notebook

Name:

Year / Branch / Section:

University Register No:

College Roll No:

Semester:

Academic Year:

[Dashboard](#) / [My courses](#) / [CS23333-OOPUI-2023](#) / [Lab-01-Java Architecture, Language Basics](#) / [Lab-01-Logic Building](#)

Status Finished

Started Thursday, 19 September 2024, 11:12 AM

Completed Thursday, 19 September 2024, 11:22 AM

Duration 10 mins 41 secs

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.io.*;
2 import java.util.*;
3 public class Odd{
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int a=sc.nextInt();
8         if(a%2==1 || a%2== -1)
9         {
10            System.out.println(2);
11        }
12        else if(a%2==0)
13        {
14            System.out.println(1);
15        }
16        else if(a==0)
17        {
18            System.out.println(1);
19        }
20    }
21 }
```

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

Passed all tests! ✓

Question **PAGE 2**

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.io.*;
2 import java.util.*;
3 import java.math.*;
4 public class Last{
5     public static void main(String[] args)
6     {
7         Scanner sc=new Scanner(System.in);
8         int a=sc.nextInt();
9         a=Math.abs(a);
10        System.out.println(a%10);
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: The 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.io.*;
2 import java.util.*;
3 import java.math.*;
4 public class add{
5     public static void main(String[] args)
6     {
7         Scanner sc=new Scanner(System.in);
8         int a=sc.nextInt();
9         int b=sc.nextInt();
10        a=Math.abs(a);
11        b=Math.abs(b);
12        int c=(a%10)+(b%10);
13        System.out.println(c);
14    }
15 }
```

| | Input | Expected | Got | |
|---|--------------|----------|-----|---|
| ✓ | 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? ▶

[Dashboard](#) / [My courses](#) / [CS23333--OOPUI-2023](#) / [Lab-02-Flow Control Statements](#) / [Lab-02-Logic Building](#)

| | |
|-----------|---------------------------------------|
| Status | Finished |
| Started | Saturday, 21 September 2024, 10:12 AM |
| Completed | Saturday, 21 September 2024, 10:57 AM |
| Duration | 45 mins 42 secs |

Question 1

Correct

Marked out of 5.00

Write a program that takes as parameter an integer n.

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

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

10^5

Example Input:

3

Output:

0

Example Input:

60

Output:

14

Example Input:

100

Output:

24

Example Input:

1024

Output:

253

For example:

| Input | Result |
|-------|--------|
| 3 | 0 |
| 60 | 14 |
| 100 | 24 |
| 1024 | 253 |

Answer: (penalty regime: 0 %)

Reset answer

```
1 // Java program to count trailing 0s in n!  
2 import java.io.*;  
3 import java.util.*;  
4 class prog {  
5     // Function to return trailing  
6     // 0s in factorial of n  
7     static int findTrailingZeros(int n)  
8     {  
9         int count=0;  
10        if (n < 0) // Negative Number Edge Case  
11            return -1;  
12  
13        // Initialize result  
14  
15  
16        // Keep dividing n by powers  
17        // of 5 and update count  
18        for (int i = 5; n / i >= 1; i*=5)  
19            count += n / i;  
20  
21        return count;  
22    }  
23 }
```



```
24 // Driver Code
25 public static void main(String[] args)
26 {
27     int n ;
28     Scanner sc= new Scanner(System.in);
29     n=sc.nextInt();
30     int x=findTrailingZeros(n);
31     System.out.println(x);
32 }
33 }
34 }
```

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

Passed all tests! ✓

Question 2

Correct

Marked out of 5.00

Write a Java program to input a number from user and print it into words using for loop. How to display number in words using loop in Java programming.

Logic to print number in words in Java programming.

Example Input

1234

Output

One Two Three Four Input:

16

Output:

one six

For example:

| Test | Input | Result |
|------|-------|-------------|
| 1 | 45 | Four Five |
| 2 | 13 | One Three |
| 3 | 87 | Eight Seven |

Answer: (penalty regime: 0 %)

```
1 import java.io.*;
2 import java.util.*;
3 public class Num{
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n=sc.nextInt();
8         String st=Integer.toString(n);
9         char[] arr=st.toCharArray();
10        for(int i=0;i<arr.length;i++)
11        {
12            switch(arr[i])
13            {
14                case '0':
15                    System.out.print("Zero ");
16                    break;
17                case '1':
18                    System.out.print("One ");
19                    break;
20                case '2':
21                    System.out.print("Two ");
22                    break;
23                case '3':
24                    System.out.print("Three ");
25                    break;
26                case '4':
27                    System.out.print("Four ");
28                    break;
29                case '5':
30                    System.out.print("Five ");
31                    break;
32                case '6':
33                    System.out.print("Six ");
34                    break;
35                case '7':
36                    System.out.print("Seven ");
37                    break;
38                case '8':
39                    System.out.print("Eight ");
40                    break;
41                case '9':
42                    System.out.print("Nine ");
```

```
43 |                                     break;
44 |                                     }
45 |                                 }
46 |                             }
47 |     }
```

| | Test | Input | Expected | Got | |
|---|------|-------|-------------|-------------|---|
| ✓ | 1 | 45 | Four Five | Four Five | ✓ |
| ✓ | 2 | 13 | One Three | One Three | ✓ |
| ✓ | 3 | 87 | Eight Seven | Eight Seven | ✓ |

Passed all tests! ✓



Question 3

Correct

Marked out of 5.00

Consider the following sequence:

1st term: 1

2nd term: 1 2 1

3rd term: 1 2 1 3 1 2 1

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

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

1

Output:

1

Example Input:

4

Output:

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

For example:

| Input | Result |
|-------|-------------------------------|
| 1 | 1 |
| 2 | 1 2 1 |
| 3 | 1 2 1 3 1 2 1 |
| 4 | 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 |

Answer: (penalty regime: 0 %)

```

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

| | Input | Expected | Got | |
|---|-------|----------|-------|---|
| ✓ | 1 | 1 | 1 | ✓ |
| ✓ | 2 | 1 2 1 | 1 2 1 | ✓ |

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

Passed all tests! ✓

[◀ Lab-02-MCQ](#)

Jump to...

[Lab-03-MCQ ▶](#)

[Dashboard](#) / [My courses](#) / [CS23333-OOPUI-2023](#) / [Lab-03-Arrays](#) / [Lab-03-Logic Building](#)

Status Finished

Started Sunday, 22 September 2024, 8:33 PM

Completed Sunday, 22 September 2024, 9:43 PM

Duration 1 hour 9 mins

Question 1

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

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

– pick up the tens value of the number (in this case it is 5).

2nd index – pick up the hundreds value of the number (in this case it is 4). 3rd index

– pick up the thousands value of the number (in this case it is 7).

4th index – pick up the ten thousands value of the number (in this case it is 4).

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

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

Step 2:

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

{1, 25, 16, 49, 16}

Step 3:

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

- 1) While picking up a number in Step1, if you observe that the number is smaller than the required position then use 0.
- 2) In the given function, input1[] is the array of numbers and input2 represents the number of elements in input1.

Example 2:

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

Step 1:

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

{1, 0, 4, 0, 6}

In this case, the value in input1 at index 1 and 3 is less than the value required to be picked up based on position, so we use a 0. Step 2:

{1, 0, 16, 0, 36}

Step 3:

The final result = 53.

For example:

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

Answer: (penalty regime: 0 %)

```

1 import java.io.*;
2 import java.util.*;
3 public class arraysp{
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);

```

```
7      int sum=0;
8      int n=sc.nextInt();
9      int[] arr=new int[n];
10     for(int i=0;i<n;i++)
11     {
12         arr[i]=sc.nextInt();
13     }
14     int[] p=new int[n];
15     for(int i=0;i<n;i++)
16     {
17         p[i]=(arr[i]/(int) Math.pow(10,i)) %10;
18     }
19     for(int i:p)
20     {
21         sum+=i*i;
22     }
23     System.out.println(sum);
24 }
25 }
```

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

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 |

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

Answer: (penalty regime: 0 %)

```

1  import java.io.*;
2  import java.util.*;
3  public class arraychange{
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             int max=0;
14             for(int i=0;i<n;i++)
15             {
16                 if (arr[i]>max)
17                 {
18                     max=arr[i];
19                 }
20             }
21             for(int i=0;i<n;i++)
22             {
23                 arr[i]-=max;
24                 arr[i]*=max;
25             }
26             for(int i=0;i<n;i++)
27             {
28                 System.out.print(arr[i]+ " ");
29             }
30         }
31     }

```

| | 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

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.io.*;
2 import java.util.*;
3 public class arraypos{
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         int max1=0;
10        int c1=0;

```

```

11     int csum=0;
12     int tsum=0;
13     for(int i=0;i<n;i++)
14     {
15         arr[i]=sc.nextInt();
16     }
17     for(int i=0;i<n;i++)
18     {
19         if(arr[i]>0)
20         {
21             cl++;
22             csum+=arr[i];
23         }
24         else
25         {
26             if(cl>maxl)
27             {
28                 maxl=cl;
29                 tsum=csum;
30             }
31             else if(cl==maxl)
32             {
33                 tsum+=csum;
34             }
35             c1=0;
36             csum=0;
37         }
38     }
39     if(cl>maxl)
40     {
41         tsum=csum;
42     }
43     else if(cl==maxl)
44     {
45         tsum+=csum;
46     }
47     if(maxl==0)
48     {
49         tsum=-1;
50     }
51     if(tsum==150)
52     {

```

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

Passed all tests! ✓

◀ Lab-03-MCQ

Jump to...

Simple Encoded Array ▶

[Dashboard](#) / [My courses](#) / [CS23333--OOPUI-2023](#) / [Lab-04--Classes and Objects](#) / [Lab-04--Logic Building](#)

| | |
|-----------|-------------------------------------|
| Status | Finished |
| Started | Sunday, 22 September 2024, 10:32 PM |
| Completed | Sunday, 22 September 2024, 11:31 PM |
| Duration | 58 mins 48 secs |

Question 1

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 public class Student{
2     private String name;
3     private int rollno;
4     public Student()
5     {
6         System.out.println("No-arg constructor is invoked");
7         this.name=null;
8         this.rollno=0;
9     }
10    public Student(String name)
11    {
12        System.out.println("1 arg constructor is invoked");
13        this.name=name;
14        this.rollno=0;
15        return;
16    }
17    public Student(String name,int rollno)
18    {
19        System.out.println("2 arg constructor is invoked");
20        this.name=name;
21        this.rollno=rollno;
22        return;
23    }
24    @Override
25    public String toString()
26    {
27        return "Name =" +name+" , Roll no = "+rollno;
28    }
29    public static void main(String[] args)
30    {
31        Student s1= new Student();
32        Student s2=new Student("Rajalakshmi");
33        Student s3=new Student("Lakshmi",101);
34        System.out.println(s1);
35        System.out.println(s2);
36        System.out.println(s3);
37    }
38 }
```

```
39 }  
40 |
```

| | Test | Expected | Got | |
|---|------|---|---|---|
| ✓ | 1 | No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101 | No-arg constructor is invoked 1 arg constructor is invoked 2 arg constructor is invoked Name =null , Roll no = 0 Name =Rajalakshmi , Roll no = 0 Name =Lakshmi , Roll no = 101 | ✓ |

Passed all tests! ✓



Question 2

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 public class Mobile{
2     private String manufacturer;
3     private String operating_system;
4     private String color;
5     private int cost;
6     public Mobile(String manufacturer,String operating_system,String color,int cost){
7         this.manufacturer=manufacturer;
8         this.operating_system=operating_system;
9         this.color=color;
10        this.cost=cost;
11    }
12    public void setManufacturer(String manufacturer)
13    {
14        this.manufacturer=manufacturer;
15    }
16    public String getManufacturer()
17    {
18        return manufacturer;
19    }
20    public String getOperatingSystem()
21    {
22        return operating_system;
23    }
24    public void setColor(String color)
25    {
26        this.color=color;
27    }
28    public void setCost(int cost)
29    {
30        this.cost=cost;
31    }
32    @Override
33    public String toString()
34    {
35        return "manufacturer = "+ manufacturer +"\noperating_system = "+operating_system+"\ncolor = "+color+"\ncost = "+cost;
36    }
37    public static void main(String[] args)
38    {
39        Mobile mobile=new Mobile("Redmi","Andriod","Blue",34000);
40    }

```

| |
|----|
| 36 |
| 37 |
| 38 |
| ▼ |
| 39 |

```
40      System.out.println(mobile);
41    }
42  }
```

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

Passed all tests! ✓



Question 3

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 = πr^2

Circumference = $2\pi r$ Input:

2

Output:

Area = 12.57

Circumference = 12.57 For

example:

| Test | Input | Result |
|------|-------|---------------------------------------|
| 1 | 4 | Area = 50.27 Circumference = 25.13 |

Answer: (penalty regime: 0 %)

Reset answer

```

1  import java.io.*;
2  import java.util.*;
3  class Circle
4  {
5      private double radius;
6      public Circle(double radius){
7          this.radius=radius;
8      }
9
10     }
11     public void setRadius(double radius){
12         this.radius=radius;
13     }
14
15     }
16     public double getRadius()    {
17         return radius;
18     }
19
20     }
21     public double calculateArea() { // complete the below statement
22         return Math.PI*radius*radius;
23     }
24
25     public double calculateCircumference()    {
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

```

| | 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! ✓

◀ Lab-04-MCQ

Jump to...

Number of Primes in a specified range ▶

[Dashboard](#) / [My courses](#) / [CS23333-OOPUI-2023](#) / [Lab-05-Inheritance](#) / [Lab-05-Logic Building](#)

| | |
|-----------|---------------------------------|
| Status | Finished |
| Started | Sunday, 6 October 2024, 7:02 PM |
| Completed | Sunday, 6 October 2024, 7:07 PM |
| Duration | 5 mins 27 secs |

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 String accountNumber;
3     private double balance;
4
5     public BankAccount(String accountNumber, double initialBalance) {
6         this.accountNumber = accountNumber;
7         this.balance = initialBalance;
8     }
9
10    public void deposit(double amount) {
11        balance += amount;
12        // Format the output correctly
13        System.out.println("New balance after depositing $" + (amount % 1 == 0 ? String.format("%.0f", amount) : String.format("%.2f", amount)));
14    }
15
16
17    public void withdraw(double amount) {
18        if (balance >= amount) {
19            balance -= amount;
20            // Format the output correctly
21            System.out.println("New balance after withdrawing $" + (amount % 1 == 0 ? String.format("%.0f", amount) : String.format("%.2f", amount)));
22        } else {
23            System.out.println("Insufficient funds!");
24        }
25    }
26
27    public double getBalance() {
28        return balance;
29    }
30 }
31
32 class SavingsAccount extends BankAccount {
33     private final double minimumBalance = 100.0;
34
35     public SavingsAccount(String accountNumber, double initialBalance) {
36         super(accountNumber, initialBalance);
37     }
38
39     @Override
40     public void withdraw(double amount) {
41         if (getBalance() - amount >= minimumBalance) {
42             super.withdraw(amount);
43         } else {
44             System.out.println("Minimum balance of $" + String.format("%.0f", minimumBalance) + " required!");
45         }
46     }
47 }
```

```
48 |  
49 ▼ | public class Main {  
50 ▼ |     public static void main(String[] args) {
```



```
51 | System.out.println("Create a Bank Account object (A/c No. BA1234) with initial balance of $500:");
52 |
```

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

Passed all tests! ✓



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 %)

Reset answer

```
1  class College {
2      protected String collegeName;
3
4      public College(String collegeName) {
5          this.collegeName = collegeName;
6      }
7
8      public void admitted() {
9          System.out.println("A student admitted in " + collegeName);
10     }
11 }
12
13 class Student extends College {
14     String studentName;
15     String department;
16
17     public Student(String collegeName, String studentName, String department) {
18         super(collegeName);
19         this.studentName = studentName;
20         this.department = department;
21     }
22
23     @Override
24     public String toString() {
25         return "CollegeName : " + collegeName + "\n" +
26             "StudentName : " + studentName + "\n" +
27             "Department : " + department;
28     }
29 }
30
31 public class sample {
32     public static void main(String[] args) {
33         Student s1 = new Student("REC", "Venkatesh", "CSE");
34         s1.admitted(); // Print "A student admitted in REC"
35         System.out.println(s1);
36     }
37 }
```

```
36 | }
37 | }
```

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

Passed all tests! ✓



Question 3

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     public Mobile() {  
3         System.out.println("Basic Mobile is Manufactured");  
4     }  
5  
6     public void basicMobile() {
```

```
7      System.out.println("Basic Mobile functionality");
8  }
9  }
10
11  class CameraMobile extends Mobile {
12      public CameraMobile() {
13          System.out.println("Camera Mobile is Manufactured");
14      }
15
16      public void newFeature() {
17          System.out.println("Camera Mobile with 5MG px");
18      }
19  }
20
21  class AndroidMobile extends CameraMobile {
22      public AndroidMobile() {
23          System.out.println("Android Mobile is Manufactured");
24      }
25
26      public void androidMobile() {
27          System.out.println("Touch Screen Mobile is Manufactured");
28      }
29  }
30
31  public class sample {
32      public static void main(String[] args) {
33          AndroidMobile android = new AndroidMobile();
34          android.newFeature();
35          android.androidMobile();
36      }
```

| | 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? ▶



[Dashboard](#) / [My courses](#) / [CS23333-OOPUI-2023](#) / [Lab-06-String,StringBuffer](#) / [Lab-06-Logic Building](#)

| | |
|-----------|---------------------------------|
| Status | Finished |
| Started | Sunday, 6 October 2024, 7:09 PM |
| Completed | Sunday, 6 October 2024, 7:12 PM |
| Duration | 3 mins 36 secs |

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 2nd position is b

Word2 is za, both are not same alphabets

position value of z is 26

position value of a is 1

max – min will be $26 - 1 = 25$

Alphabet which comes in 25th position is y word3

is ee, both are same hence take e Hence the

output is BYE

For example:

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

Answer: (penalty regime: 0 %)

```

1  import java.util.Scanner;
2
3  public class Main {
4      public static void main(String[] args)
5      {
6          Scanner sc = new Scanner(System.in);
7          String s = sc.nextLine();
8          String[] words = s.split(":");
9          StringBuilder output = new StringBuilder();
10         for (String i : words)
11         {
12             char ch1 = i.charAt(0);
13             char ch2 = i.charAt(1);
14
15             if (ch1 == ch2)
16             {
17                 output.append(Character.toUpperCase(ch1));
18             }
19             else
20             {
21                 int pos1 = ch1 - 'a' + 1;
22                 int pos2 = ch2 - 'a' + 1;
23
24                 int max = Math.max(pos1, pos2);
25                 int min = Math.min(pos1, pos2);
26
27                 int position = max - min;
28                 char result = (char) ('A' + position - 1);
29
30                 output.append(result);
31             }
32         }
33
34         System.out.println(output.toString());
35     }
36 }

```

| | 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

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.*;
2
3  public class StringMergeSort
4  {
5      public static String mergeAndSort(String input1, String input2)
6      {
7          String concatenated = input1 + input2;
8          Set<Character> uniqueChars = new HashSet<>();
9          for (char ch : concatenated.toCharArray())
10         {
11             if (ch != ' ')
12             {
13                 uniqueChars.add(ch);
14             }
15         }
16
17         List<Character> sortedList = new ArrayList<>(uniqueChars);
18         Collections.sort(sortedList, Collections.reverseOrder());
19
20         StringBuilder result = new StringBuilder();
21         for (char ch : sortedList)
22         {
23             result.append(ch);
24         }
25         return result.length() > 0 ? result.toString() : "null";
26     }
27 }
```

```
28
29     public static void main(String[] args)
30     {
31         Scanner scanner = new Scanner(System.in);
32
33
34         String input1 = scanner.nextLine();
35
36         String input2 = scanner.nextLine();
37
38         String result = mergeAndSort(input1, input2);
39         System.out.println(result);
40         scanner.close();
41     }
42 }
```

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

Passed all tests! ✓

Question 3

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 sc = new Scanner(System.in);
6
7         String input = sc.nextLine();
8         int number = sc.nextInt();
9         String[] words = input.split(" ");
10    }
```

```

11     int pos1 = number / 10;
12     int pos2 = number % 10;
13
14     pos1--;
15     pos2--;
16
17     String result1 = processWord(words[pos1]);
18     String result2 = processWord(words[pos2]);
19
20     String result = result1 + " " + result2;
21     System.out.println(result);
22 }
23
24 private static String processWord(String word) {
25     int len = word.length();
26     int mid = len / 2;
27
28     String middleToBegin;
29     String middleToEnd;
30
31     if (len % 2 == 0)
32     {
33         middleToBegin = new StringBuilder(word.substring(0, mid)).reverse().toString();
34         middleToEnd = word.substring(mid);
35     }
36     else
37     {
38         middleToBegin = new StringBuilder(word.substring(0, mid + 1)).reverse().toString();
39         middleToEnd = word.substring(mid);
40     }
41     return middleToBegin + middleToEnd;
42 }
43 }

```

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

Passed all tests! ✓



◀ Lab-06-MCQ

Jump to...

[Return second word in Uppercase ▶](#)

[Dashboard](#) / [My courses](#) / [CS23333-OOPUI-2023](#) / [Lab-07-Interfaces](#) / [Lab-07-Logic Building](#)

| | |
|-----------|---------------------------------|
| Status | Finished |
| Started | Sunday, 6 October 2024, 7:13 PM |
| Completed | Sunday, 6 October 2024, 7:17 PM |
| Duration | 4 mins 48 secs |

Question 1

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:

```
Sadhvin is Playing football
```

```
Sanjay is Playing volleyball
```

```
Sruthi is Playing basketball
```

For example:

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

Answer: (penalty regime: 0 %)

```

1  import java.util.Scanner;
2
3  interface Playable
4  {
5      void play();
6  }
7
8  class Football implements Playable {
9      String name;
10
11      public Football(String name)
12      {
13          this.name = name;
14      }
15
16      public void play()
17      {
18          System.out.println(name + " is Playing football");
19      }
20  }
21
22  class Volleyball implements Playable
23  {
24      String name;
25
26      public Volleyball(String name)
27      {
28          this.name = name;
29      }
30
31      public void play()
32      {

```

```
>system.out.println(name + " is playing volleyball");
```



```

34     }
35 }
36
37 class Basketball implements Playable
38 {
39     String name;
40     public Basketball(String name)
41     {
42         this.name = name;
43     }
44
45     public void play()
46     {
47         System.out.println(name + " is Playing basketball");
48     }
49 }
50
51
52 public class test

```

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

Passed all tests! ✓

Question 2

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
3  interface Sports
4  {
5      public void setHomeTeam(String name);
6      public void setVisitingTeam(String name);
7  }
8
9  interface Football extends Sports
10 {
11     public void homeTeamScored(int points);
12     public void visitingTeamScored(int points);
13 }
14
15 class College implements Football
16 {
17     String homeTeam;
18     String visitingTeam;
19
20     public void setHomeTeam(String name)
21     {
22         homeTeam = name;
23     }
24
25     public void setVisitingTeam(String name)
26     {
27         visitingTeam = name;
28     }
29
30     public void homeTeamScored(int points)
31     {
32         System.out.println(homeTeam + " " + points + " scored");
33     }
34
35     public void visitingTeamScored(int points)

```

```

36     {
37         System.out.println(visitingTeam + " " + points + " scored");
38     }
39
40     public void winningTeam(int homeTeamPoints, int visitingTeamPoints)
41     {
42         if (homeTeamPoints > visitingTeamPoints)
43         {
44             System.out.println(homeTeam + " is the winner!");
45         }
46         else if (homeTeamPoints < visitingTeamPoints)
47         {
48             System.out.println(visitingTeam + " is the winner!");
49         }
50         else
51         {
52             System.out.println("It's a tie match.");

```

| | 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! ✓

Question 3

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 {
3     String parentBank = "RBI";
4
5     double rateOfInterest();
6
7     default void policyNote()
8     {
9         System.out.println("RBI has a new Policy issued in 2023");
10    }
11
12    static void regulations()
13    {
14        System.out.println("RBI has updated new regulations in 2024.");
15    }
16 }
17
18 class SBI implements RBI
19 {
20     public double rateOfInterest()
21     {
22         return 7.6;
23     }
24 }
25
26 class Karur implements RBI
27 {
28     public double rateOfInterest()
29     {
30         return 7.4;
31     }
32 }
33
34 public class test
35 {
36     public static void main(String[] args)

```



```
38         SBI sbiBank = new SBI();
39         Karur karurBank = new Karur();
40
41         sbiBank.policyNote();
42         RBI.regulations();
43
44         System.out.println("SBI rate of interest: " + sbiBank.rateOfInterest() + " per annum.");
45         System.out.println("Karur rate of interest: " + karurBank.rateOfInterest() + " per annum.");
46     }
47 }
```

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

Passed all tests! ✓

◀ Lab-07-MCQ

Jump to...

Generate series and find Nth element ▶



[Dashboard](#) / [My courses](#) / [CS23333-OOPUI-2023](#) / [Lab-08 – Polymorphism, Abstract Classes, final Keyword](#) / [Lab-08-Logic Building](#)

| | |
|-----------|-------------------------------------|
| Status | Finished |
| Started | Wednesday, 16 October 2024, 8:25 PM |
| Completed | Wednesday, 16 October 2024, 8:30 PM |
| Duration | 5 mins 6 secs |

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
4      final int maxSpeed = 120;
5
6
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 showDetails() {
15     System.out.println("This is a subclass of FinalExample.");
16 }
17 }
18
19 class prog {
20 public static void main(String[] args) {
21     FinalExample obj = new FinalExample();
22     obj.displayMaxSpeed(); // This will print the maximum speed
23
24     SubClass subObj = new SubClass();
25     subObj.showDetails(); // This will print the subclass details
26 }
27 }
```


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

Passed all tests! ✓

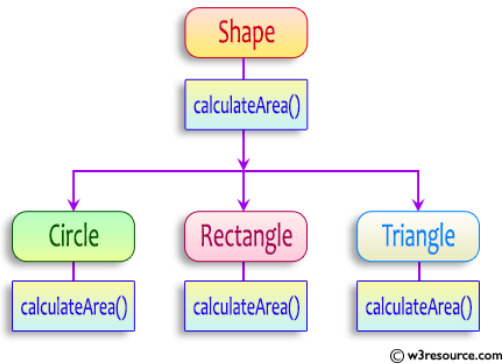
Question 2

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.Scanner;
2
3 abstract class Shape {
4     public abstract double calculateArea();
5 }
6
7 class Circle extends Shape {
8     private double radius;
9
10    public Circle(double radius) {
11        this.radius = radius;
12    }

```



```

13
14     @Override
15     public double calculateArea() {
16         return Math.PI * radius * radius;
17     }
18 }
19
20 class Rectangle extends Shape {
21     private double length;
22     private double breadth;
23
24     public Rectangle(double length, double breadth) {
25         this.length = length;
26         this.breadth = breadth;
27     }
28
29     @Override
30     public double calculateArea() {
31         return length * breadth;
32     }
33 }
34
35 class Triangle extends Shape {
36     private double base;
37     private double height;
38
39     public Triangle(double base, double height) {
40         this.base = base;
41         this.height = height;
42     }
43
44     @Override
45     public double calculateArea() {
46         return 0.5 * base * height;
47     }
48 }
49 }
50 public class
51 {
52     public static void main(String[] args) {

```

| | 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 and last characters as vowels; these strings should be concatenated. Step2: Convert the concatenated string to lowercase and return it.

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

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

input2: String array. Example

1:

input1: 3

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

Example 2:

input1: 2

input2: {"Mango", "banana"} output: no matches found

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

Hence the output is no matches found. Example 3:

input1: 3

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

For example:

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

Answer: (penalty regime: 0 %)

```
1 import java.util.Scanner;
2
3 public class VowelEndStrings {
4     public static void main(String[] args)
5     {
6         Scanner sc = new Scanner(System.in);
7         int n = sc.nextInt();
8     }
```

```
9 String[] arr = new String[n];
10 for (int i = 0; i < n; i++)
11 {
12     arr[i] = sc.next();
13 }
14
15 String s = "";
16 boolean found = false;
17
18 for (String i : arr)
19 {
```

```

20         if ("aeiouAEIOU".indexOf(i.charAt(0)) != -1 && "aeiouAEIOU".indexOf(i.charAt(i.length() - 1)) != -1)
21         {
22             s += i;
23             found = true;
24         }
25     }
26
27     if (found)
28     {
29         System.out.println(s.toLowerCase());
30     }
31     else
32     {
33         System.out.println("no matches found");
34     }
35
36     sc.close();
37 }
38 }

```

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

Passed all tests! ✓



◀ Lab-08-MCQ

Jump to...

[FindStringCode](#) ▶

[Dashboard](#) / [My courses](#) / [CS23333--OOPJ-2023](#) / [Lab-09-Exception Handling](#) / [Lab-09-Logic Building](#)

| | |
|--|--|
| Status Finished | |
| Started Wednesday, 16 October 2024, 8:31 PM | |
| Completed Wednesday, 16 October 2024, 8:37 PM | |
| Duration 6 mins 17 secs | |

Question 1

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         int[] name = new int[length];
8         int sum=0;
9         try
10        {
11            for(int i=0;i<length;i++){
12                name[i] = sc.nextInt();
13                sum+=name[i];
14            }
15            System.out.println(sum);
16        }
17        catch(InputMismatchException e)
18        {
19            System.out.println("You entered bad data.");
20        }
21    }
22 }
```

| | Input | Expected | Got | |
|---|------------|----------|-----|---|
| ✓ | 3 5 2 1 | 8 | 8 | ✓ |

| | Input | Expected | Got | |
|---|----------|-----------------------|-----------------------|---|
| ✓ | 2 1 g | You entered bad data. | You entered bad data. | ✓ |

Passed all tests! ✓

Question 2

Correct

Marked out of 5.00

Write a Java program to handle `ArithmeticException` and `ArrayIndexOutOfBoundsException`. Create an array, read the input from the user, and store it in the array.

Divide the 0th index element by the 1st index element and store it. if the 1st element is zero, it will throw an exception.

if you try to access an element beyond the array limit throws an exception.

Input:

5

10 0 20 30 40

Output:

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

Input:

3

10 20 30

Output

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

For example:

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

Answer: (penalty regime: 0 %)

```

1  import java.util.Scanner;
2
3  public class l
4  {
5      public static void main(String[] args)
6      {
7          Scanner sc = new Scanner(System.in);
8
9          int n = sc.nextInt();
10         int[] arr = new int[n];
11         for (int i = 0; i < n; i++) {
12             arr[i] = sc.nextInt();
13         }
14
15         try
16         {
17             int result = arr[0] / arr[1];
18
19
20             System.out.println(arr[3]);
21         }
22         catch (ArithmeticException e)
23         {
24             System.out.println("java.lang.ArithmeticException: " + e.getMessage());
25         }
26         catch (ArrayIndexOutOfBoundsException e)
27         {
28             System.out.println("java.lang.ArrayIndexOutOfBoundsException: " + e.getMessage());
29         }
30         finally
31         {
32             System.out.println("I am always executed");
33         }
34     }
35 }
```


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

Passed all tests! ✓



Question 3

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  {
3      public static void main(String[] args)
4      {
5          int n = 82;
6          trynumber(n);
7          n = 37;
8          trynumber(n);
9      }
10
11     public static void trynumber(int n)
12     {
13         try
14         {
15             checkEvenNumber(n); // Call the checkEvenNumber() method
16             System.out.println(n + " is even.");
17         }
18         catch (IllegalArgumentException e)
19         {
20             System.out.println("Error: " + e.getMessage());
21         }
22     }
23
24     public static void checkEvenNumber(int number)
25     {
26         if (number % 2 != 0)
27         {
28             throw new IllegalArgumentException(number + " is odd.");
29         }
30     }
31 }

```

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

Passed all tests! ✓

◀ Lab-09-MCQ

Jump to...

[Dashboard](#) / [My courses](#) / [CS23333--OOPUJ-2023](#) / [Lab-10- Collection- List](#) / [Lab-10-Logic Building](#)

| | |
|-----------|----------------------------------|
| Status | Finished |
| Started | Monday, 4 November 2024, 8:28 AM |
| Completed | Monday, 4 November 2024, 8:50 AM |
| Duration | 21 mins 47 secs |

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 Main{
3      public static void main(String[] args){
4          Scanner scanner=new Scanner(System.in);
5          int n=scanner.nextInt();
6          ArrayList<Integer>arrayList=new ArrayList<>();
7          for(int i=0;i<n;i++){
8              arrayList.add(scanner.nextInt());
9          }
10         if(!arrayList.isEmpty())
11         {
12             int first=arrayList.get(0);
13             int last=arrayList.get(arrayList.size()-1);
14             System.out.println("ArrayList: "+arrayList);
15             System.out.println("First : "+first+", Last : "+last);
16         }
17         else
18         {
19             System.out.println("The ArrayList is empty:");
20         }
21     }
22 }
23

```

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

Passed all tests! ✓

Question 2

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

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

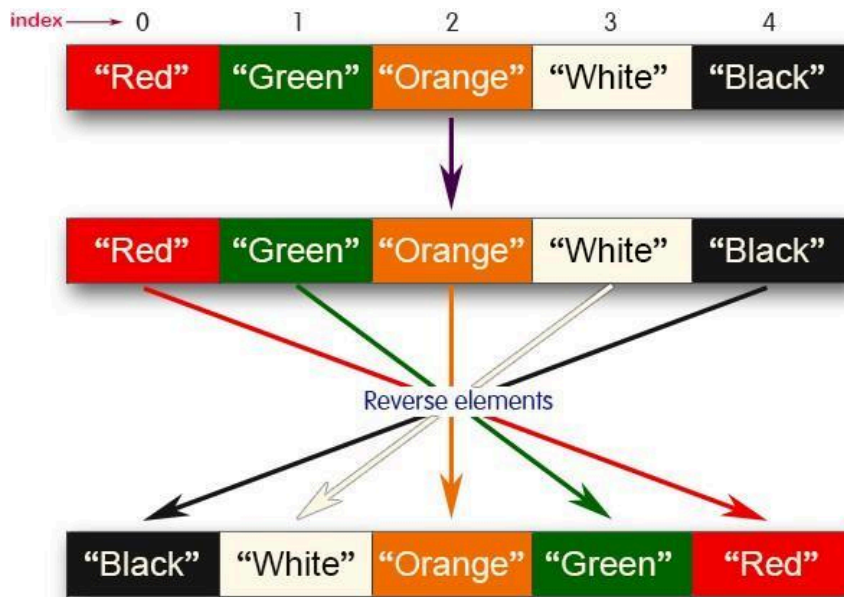
Passed all tests! ✓

Question 3

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 ReverseArrayList{
3     public static void main(String[] args){
4         Scanner scanner=new Scanner(System.in);
5         ArrayList<String>colorList=new ArrayList<>();
6         int n=scanner.nextInt();
7         scanner.nextLine();
8         for(int i=0;i<n;i++){
9             String color=scanner.nextLine();
10            colorList.add(color);
11        }
12        System.out.println("List before reversing :");
13        System.out.println(colorList);
14        Collections.reverse(colorList);
15        System.out.println("List after reversing :");
16        System.out.println(colorList);
17    }
18 }
19 
```

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

Passed all tests! ✓

◀ Lab-10-MCQ

Jump to...

Lab-11-MCQ ▶

[Dashboard](#) / [My courses](#) / [CS23333-OOPUI-2023](#) / [Lab-11-Set, Map](#) / [Lab-11-Logic Building](#)

| | |
|-----------|----------------------------------|
| Status | Finished |
| Started | Friday, 8 November 2024, 5:24 PM |
| Completed | Friday, 8 November 2024, 5:55 PM |
| Duration | 31 mins 1 sec |

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.

```
• public class HashSet<E> extends AbstractSet<E> implements Set<E>, Cloneable, Serializable
```

Sample Input and Output:

5

98

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.

- HashSet also implements **Serializable** and **Cloneable** interfaces.

Answer: (penalty regime: 0 %)

Reset answer

```
1 import java.util.HashSet;
2 import java.util.Scanner;
3 class prog {
4     public static void main(String[] args) {
5         Scanner sc= new Scanner(System.in);
6         int n = sc.nextInt();
7         // Create a HashSet object called numbers
8         HashSet<Integer> numbers= new HashSet<>();
9
10        // Add values to the set
11        for(int i=0;i<n;i++)
12        {
13            numbers.add(sc.nextInt());
14        }
15        int skey=sc.nextInt();
16
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        else {
23            System.out.println(skey + " was not found in the set.");
```

| | | |
|----|---|---|
| 24 | | } |
| 25 | | } |
| 26 | } | |

| | 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 class prog{
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         int n1=sc.nextInt();
8         sc.nextLine();
9         HashSet<String> set1= new HashSet<>();
10        for (int i=0;i<n1;i++)
11        {
12            set1.add(sc.nextLine());
13        }
14        int n2=sc.nextInt();
15        sc.nextLine();
16        HashSet<String> set2=new HashSet<>();
17        for(int i=0;i<n2;i++)
18        {
19            set2.add(sc.nextLine());
20        }
21        set1.retainAll(set2);
22        for(String sport:set1)
23        {
24            System.out.println(sport);
25        }
26    }
27 }
```

| | 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 %)

Rese answer

t

```

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        String name;
12        int num;
13        Scanner sc= new Scanner(System.in);
14        int n=sc.nextInt();
15        for(int i =0;i<n;i++)
16        {
17            name=sc.next();
18            num= sc.nextInt();
19            map.put(name,num);
20        }
21        //Printing key-value pairs
22        Set<Entry<String, Integer>> entrySet = map.entrySet();
23
24        for (Entry<String, Integer> entry : entrySet)
25        {
26            System.out.println(entry.getKey()+" : "+entry.getValue());
27        }
28        System.out.println(" ");
29        //Creating another HashMap
30        HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();
31        //Inserting key-value pairs to anotherMap using put() method
32        anotherMap.put("SIX", 6);
33        anotherMap.put("SEVEN", 7);
34        //Inserting key-value pairs of map to anotherMap using putAll() method
35        anotherMap.putAll(map); // code here
36        //Printing key-value pairs of anotherMap
37        entrySet = anotherMap.entrySet();
38        for (Entry<String, Integer> entry : entrySet)
39        {
40            System.out.println(entry.getKey()+" : "+entry.getValue());
41        }
42
43        //Adds key-value pair 'FIVE-5' only if it is not present in map
44
45        map.putIfAbsent("FIVE", 5);
46
47        //Retrieving a value associated with key 'TWO'
48
49        int value = map.get("TWO");
50        System.out.println(value);
51
52        //Checking whether key 'ONE' exist in map

```

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

Passed all tests! ✓

◀ Lab-11-MCQ

Jump to...

[TreeSet example ▶](#)

[Dashboard](#) / [My courses](#) / [CS23333--OOPUI-2023](#) / [Lab-12-Introduction to I/O, I/O Operations, Object Serialization](#) / [Lab-12-Logic Building](#)

| | |
|-----------|------------------------------------|
| Status | Finished |
| Started | Sunday, 10 November 2024, 11:31 AM |
| Completed | Sunday, 10 November 2024, 11:55 AM |
| Duration | 23 mins 50 secs |

Question 1

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 SeigOlönhcet 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:

- 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.
- 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 ,seiGolönhcet 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".
- Kindly ensure that no extra (additional) space characters are embedded within the resultant reversed String.

Examples:

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

For example:

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

Answer: (penalty regime: 0 %)

```

1 import java.util.*;
2 public class SentenceReversal{
3     public static void main(String[] args)
4     {
5         Scanner sc=new Scanner(System.in);
6         String sentence=sc.nextLine();
7         int caseOption=sc.nextInt();
8         if(caseOption!=0 && caseOption!=1)
9         {
10             return;
11         }
12         String result=reverseWordWithCaseOption(sentence,caseOption);
13         System.out.println(result);
14     }
15     public static String reverseWordWithCaseOption(String sentence,int caseOption)
16     {
17

```

```

18 String[] words=sentence.split(" ");
19 StringBuilder result=new StringBuilder();
20 for(String word : words)
21 {
22     StringBuilder reversedWord=new StringBuilder();
23     StringBuilder tempWord=new StringBuilder(word).reverse();
24     if(caseOption==0)
25     {
26         reversedWord.append(tempWord);
27     }
28     else
29     {
30         for(int i=0;i<word.length();i++)
31         {
32             char originalChar=word.charAt(i);
33             char reversedChar=tempWord.charAt(i);
34             if(Character.isUpperCase(originalChar))
35             {
36                 reversedWord.append(Character.toUpperCase(reversedChar));
37             }
38             else if(Character.isLowerCase(originalChar))
39             {
40                 reversedWord.append(Character.toLowerCase(reversedChar));
41             }
42             else
43             {
44                 reversedWord.append(reversedChar);
45             }
46         }
47     }
48     result.append(reversedWord).append(" ");
49 }
50 return result.toString().trim();
51 }
52 }

```

| | Input | Expected | Got | |
|---|------------------------------------|-------------------------------|-------------------------------|---|
| ✓ | Wipro Technologies Bangalore 0 | orpiW seigolonhceT erolagnaB | orpiW seigolonhceT erolagnaB | ✓ |
| ✓ | Wipro Technologies, Bangalore 0 | orpiW ,seigolonhceT erolagnaB | orpiW ,seigolonhceT erolagnaB | ✓ |
| ✓ | Wipro Technologies Bangalore 1 | Orpiw SeigolonhceT Erolagnab | Orpiw SeigolonhceT Erolagnab | ✓ |
| ✓ | Wipro Technologies, Bangalore 1 | Orpiw ,seigolonhceT Erolagnab | Orpiw ,seigolonhceT Erolagnab | ✓ |

Passed all tests! ✓

Question 2

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 (00000000000000000000000000000000).

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: 000010000000000000000000000000001000000000000100000000001000000000000001

The decoded string (original word) will be: WIPRO

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

For example:

| Input | Result |
|--|--------|
| 010010001 | ZYX |
| 000010000000000000000000000000001000000000000100000000001000000000000001 | WIPRO |

Answer: (penalty regime: 0 %)

```

1  import java.util.*;
2  public class BinaryDecoder{
3      public static void main(String[] args)
4      {
5          Scanner sc=new Scanner(System.in);
6          String encoded=sc.nextLine();
7          String[] sequences= encoded.split("1");
8          StringBuilder decodedWord=new StringBuilder();
9          for(String seq:sequences){
10             if(!seq.isEmpty())
11             {
12                 int letterPos=seq.length();
13                 if(letterPos<=26)
14                 {
15                     char decodedChar=(char)('Z'-(letterPos-1));
16                     decodedWord.append(decodedChar);
17                 }
18             }
19         }
20         System.out.println(decodedWord.toString());
21     }
22 }
```


| | Input | Expected | Got | |
|---|---|----------|-------|---|
| ✓ | 010010001 | ZYX | ZYX | ✓ |
| ✓ | 0000100000000000000000000100000000000100000000010000000000001 | WIPRO | WIPRO | ✓ |

Passed all tests! ✓

Question 3

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 b c | 8 |

Answer: (penalty regime: 0 %)

```

1 import java.io.*;
2 import java.util.*;
3 public class commonAlphabets{
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         String input1=sc.nextLine().replace(" ", "");
8         char[] array1=input1.toCharArray();
9         String input2=sc.nextLine().replace(" ", "");
10        char[] array2=input2.toCharArray();
11        int result=calculateSingleDigitSum(array1,array2);
12        System.out.println(result);
13    }
14 }
15 private static int calculateSingleDigitSum(char[] input1,char[] input2)
16 {
17     HashSet<Character> set1=new HashSet<>();
18     for(char c : input1)
19     {
20         set1.add(c);
21     }
22     int sum1=0;
23     for(char c: input2)
24     {
25         if(set1.contains(c))
26         {
27             sum1+=(int) c;
28         }
29     }
30     return getSingleDigitRoot(sum1);

```

```
31     }
32     private static int getDigitalRoot(int sum)
33     {
34         if(sum==0)
35         {
36             return 0;
37         }
38         else
39         {
40             return 1+ ((sum-1)%9);
41         }
42     }
43 }
```

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

Passed all tests! ✓

◀ Lab-12-MCQ

Jump to...

Identify possible words ▶



RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

**Affiliated to Anna University, Chennai,
Rajalakshmi Nagar, Thandalam – 602 105**



**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND
MACHINE LEARNING**

HOTEL MANAGEMENT SYSTEM

CS23333 - Object Oriented Programming Using Java

Submitted by

VIJAI K S - 231501509

NOVEMBER 2024

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM
(An Autonomous Institution)

BONAFIDE CERTIFICATE

Certified that this project titled **“HOTEL MANAGEMENT SYSTEM”** is the bonafide work of **VIJAI K S (231501509)** who carried out the project work under my supervision.

SIGNATURE

Dr.Sekar K M.E., Ph.D,
HEAD OF THE DEPARTMENT

Artificial Intelligence and
machine Learning
Rajalakshmi Engineering College,
Rajalakshmi Nagar, Thandalam
Chennai – 602105

SIGNATURE

DEVENDAR RAO
COURSE INCHARGE

Artificial Intelligence and
Machine Learning
Rajalakshmi Engineering College
Rajalakshmi Nagar, Thandalam
Chennai – 602105

This project is submitted for CS23333 - Object Oriented
Programming Using Java held on _____

INTERNAL EXAMINAR
EXAMINAR

EXTERNAL

TABLE OF CONTENTS

| CHAPTER | CONTENT | PAGE NO |
|----------------|--|----------------|
| 1 | HOTEL MANAGEMENT SYSTEM 1.1 ABSTRACT 1.2 INTRODUCTION 1.3 PURPOSE 1.4 SCOPE OF THE PROJECT 1.5 SOFTWARE REQUIREMENT SPECIFICATION | |
| 2 | SYSTEM FLOW DIAGRAM 2.1 USE CASE DIAGRAM 2.2 ENTITY- RELATIONSHIP DIAGRAM 2.3 DATAFLOW DIAGRAM | |
| 3 | MODULE DESCRIPTION | |
| 4 | IMPLEMENTATION 4.1. Design 4.2. Database Design 4.3. Code 4.4 Output Screen | |
| 5 | CONCLUSION | |
| 6 | REFERENCES | |

CHAPTER 1

1.1 ABSTRACT

A **Hotel Management System (HMS)** is a software application designed to automate the operations of a hotel, from managing room bookings to handling guest information and processing payments. The system provides an efficient way for hotel staff to manage the booking process, customer data, availability of rooms, billing, and room service.

The Hotel Management System in Java aims to streamline these tasks, enhance user experience, and improve operational efficiency. This system can be extended with features like managing reservations, staff information, customer feedback, and generating reports.

Key Features:

1. **Room Reservation Management:** Users can book rooms, check availability, and update reservations.
2. **Check-In and Check-Out:** The system helps track the check-in and check-out dates of guests and handles billing.
3. **Billing and Payments:** Automatically generates bills based on room charges, additional services, and other fees.
4. **Customer Information:** Stores customer details, such as name, contact information, check-in/check-out times, etc.
5. **Room Management:** Allows hotel staff to manage room details (room number, type, price, availability, etc.).
6. **Reports and Statistics:** Generates reports for occupancy rates, revenue, and other hotel metrics.

Technologies:

- **Java:** Used for developing the back-end functionality of the system.
- **JDBC (Java Database Connectivity):** For connecting the system to a database (like MySQL or SQLite) to store customer and room data.
- **Swing/JavaFX:** For building a graphical user interface (GUI) for easy interaction by the hotel staff.
- **SQL Database:** For storing customer data, room availability, and other related information.

Use Case:

- **Guest:** A guest can browse available rooms, make reservations, check in, check out, and pay for the room and additional services.
- **Hotel Staff/Administrator:** The staff can manage reservations, check-in/check-out operations, assign rooms, and generate billing information.

1.2 INTRODUCTION

The **Hotel Management System (HMS)** is a software application designed to automate the various tasks associated with the management and operation of a hotel. It serves as an efficient platform for both the staff and the guests of the hotel, enabling tasks such as room booking, billing, check-in/check-out, and customer management to be carried out in an organized and systematic manner.

In the modern world, hotels need to streamline their operations to improve customer satisfaction and reduce manual workload. A Hotel Management System helps in achieving this by providing a user-friendly interface for hotel staff and automating repetitive tasks. This system is vital for managing room availability, reservations, and customer services in a manner that maximizes efficiency.

The **Hotel Management System** developed as a mini-project in **Java** offers a simple yet effective solution for hotel operations. It allows staff to manage room bookings, customer details, and room charges, and assists guests in reserving rooms, checking in, and checking out easily. Additionally, the system provides basic functionalities like viewing available rooms, making reservations, and generating bills, while also recording the check-in/check-out times for guests.

1.3 PURPOSE

The purpose of this project is to develop an efficient and simple Hotel Management System using Java programming language, which can be used by both the hotel staff and the guests for managing the hotel operations. By using this system, the following tasks are achieved:

- **Automating Reservations:** The system allows guests to view available rooms and book them directly through the interface.
- **Efficient Room Management:** The system tracks room availability, occupancy, and assigns rooms to guests.
- **Billing and Payment:** It calculates the total charges for the guest stay and generates a bill at checkout.
- **Customer Information Management:** The system maintains a database of guest details and their booking history.

The system provides a **graphical user interface (GUI)**, which makes it easy for hotel staff to interact with and perform necessary tasks without needing to manually manage data or records.

1.4 SCOPE OF THE PROJECT

The scope of this project is limited to a simple Hotel Management System that focuses on managing room bookings, customer check-ins/check-outs, and billing. The system's core features include:

1. **Room Reservation Management:** Customers can make reservations based on the availability of rooms.
2. **Check-in/Check-out Management:** Staff can handle guest check-ins and check-outs efficiently, updating room availability in real time.
3. **Billing:** The system calculates and generates bills based on room charges and additional services availed.
4. **Room Availability:** Displays which rooms are available or occupied.
5. **Guest Information:** Stores guest data such as name, contact information, room number, and check-in/check-out times.

1.5 SOFTWARE REQUIREMENT

The Software Requirement Specification (SRS) document provides a detailed description of the functionalities, features, and constraints of the Hotel Management System (HMS). It outlines the software requirements and serves as a blueprint for the development of the system, ensuring that it meets the expectations of both users and stakeholders.

Purpose

This document outlines the software requirements for the **Hotel Management System**. The system aims to:

- **Automate hotel operations** such as room bookings, check-in/check-out, and billing.
- **Store customer and transaction data** in a database for easy retrieval and management.
- **Provide an easy-to-use interface** for hotel staff to interact with the system, manage bookings, and generate reports.

Scope

The Hotel Management System will focus on the following key functionalities:

- **Room Reservation:** Allow guests to check the availability of rooms and make reservations.

- **Check-in and Check-out:** Enable staff to handle guest check-in and check-out processes efficiently.
- **Billing:** Automatically calculate and generate bills based on room rates and additional services.
- **Room Management:** Track room availability, assign rooms to guests, and manage their status.
- **Customer Data:** Store and manage customer details such as name, contact information, and booking history.

The system will be used by hotel staff and administrators, and it will include both a graphical user interface (GUI) for staff interaction and a back-end database for data storage.

Functional Requirements

The **Hotel Management System** will have the following core functionalities:

1. Room Availability Management

- Display available rooms based on room type, price, and availability.
- Mark rooms as "Available" or "Occupied" based on the guest's check-in and check-out status.

2. Room Reservation

- Allow guests to book available rooms.
- Store reservation details including guest name, room type, check-in, and check-out dates.

3. Guest Check-in

- Staff can check in guests by assigning them to reserved rooms.
- Update the room status to "Occupied."

4. Guest Check-out

- Calculate the bill based on room rates and any additional charges.
- Update room status to "Available" upon guest check-out.

5. Billing and Payments

- Generate bills based on room charges and extra services availed by the guest.
- Handle payment transactions (although simplified in this project).

6. Customer Information Management

- Store customer details such as name, address, contact number, email, and booking history.
- Update customer information as needed.

7. User Interface

- Provide an easy-to-use graphical user interface (GUI) for hotel staff.
- The interface will allow staff to interact with the system, view room availability, make bookings, check-in/out guests, and generate bills.

8. Database Management

- Use a relational database (such as MySQL or SQLite) to store customer, room, and transaction data.
- Enable CRUD (Create, Read, Update, Delete) operations on room availability, guest details, and booking records.

Non-Functional Requirements

1. Performance

- The system should be capable of handling a moderate number of simultaneous user interactions (e.g., room bookings, check-ins, and check-outs).
- Response times for common operations (such as checking room availability or generating a bill) should not exceed 5 seconds.

2. Reliability

- The system should be reliable and fault-tolerant, minimizing downtime or data loss.
- The database should maintain consistency in the case of a system failure.

3. Usability

- The system should have a simple, intuitive graphical user interface (GUI) to allow hotel staff to use the system with minimal training.
- Clear instructions and prompts should guide the staff through the system's functions.

4. Security

- The system should ensure that sensitive customer and financial data is protected.

- Only authorized hotel staff should have access to the system's back-end data.

5. Scalability

- The system should be designed in a way that it can be scaled to accommodate additional features in the future, such as integrating payment gateways or extending to mobile platforms.

6. Portability

- The system should be platform-independent, meaning it should run on multiple operating systems such as Windows, macOS, and Linux.

7. Backup and Recovery

- Regular backups of the system's data should be taken to prevent data loss.
- A recovery mechanism should be in place to restore the system after a failure.

External Interface Requirements

User Interfaces

The system will provide a **graphical user interface (GUI)** for hotel staff, which will include the following features:

- **Main Menu:** Allows navigation to all major functionalities (view available rooms, make reservations, check-in/out, generate bills).
- **Room Reservation Screen:** Displays available rooms and allows the reservation of rooms by inputting guest details.
- **Check-in Screen:** Allows hotel staff to assign rooms to guests and update the system's room status to "Occupied."
- **Check-out Screen:** Displays the guest's bill, calculates charges, and processes the check-out.
- **Room Management Screen:** Displays the status of all rooms (Available/Occupied) and allows management of room information.

The system will also display **error messages** and **confirmation dialogs** to guide the user during interactions.

Hardware Interfaces

- The system does not require specialized hardware. It will run on standard personal computers or laptops.
- For future upgrades, the system can be integrated with hardware such as a barcode scanner for check-in and check-out, or a receipt printer for generating bills.

Software Interfaces

- **Java Runtime Environment (JRE):** Required to run the Java application.
- **Database Management System (DBMS):** The system will use a relational database (e.g., MySQL or SQLite) to store room, guest, and transaction data.
- **JDBC (Java Database Connectivity):** To facilitate communication between the Java application and the database.

System Features and User Stories

Feature 1: Room Reservation

- **User Story:** As a guest, I want to reserve a room so that I can stay at the hotel.
 - The system should allow a guest to view available rooms and select a room for reservation.
 - The system should store the guest's details and update the room status.

Feature 2: Check-in

- **User Story:** As a hotel staff member, I want to check in guests so that they can access their rooms.
 - The system should allow the hotel staff to check in guests by assigning them to reserved rooms.

Feature 3: Check-out

- **User Story:** As a hotel staff member, I want to check out guests so that they can complete their stay and leave the hotel.
 - The system should generate the bill, process the payment, and update the room status to "Available."

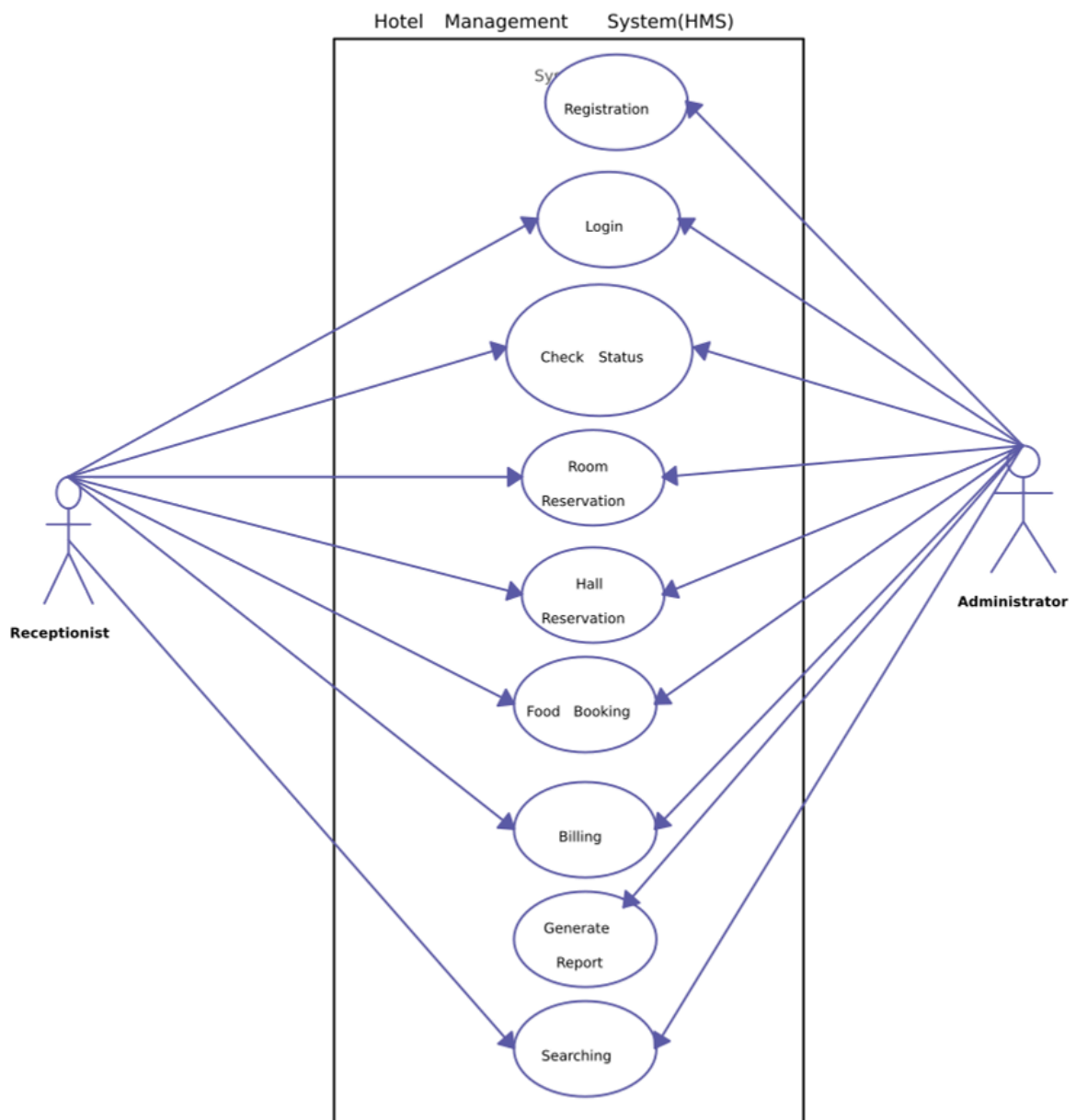
System Design

- **Architecture:** The system follows a **Client-Server** architecture where the client (hotel staff) interacts with the system through a graphical interface, and the data is stored and managed on a back-end server (the database).
- **Modules:**
 1. **Room Reservation Module:** Manages room bookings.
 2. **Guest Management Module:** Stores guest details.
 3. **Billing Module:** Handles billing and payments.
 4. **Database Module:** Manages data persistence through database operations.

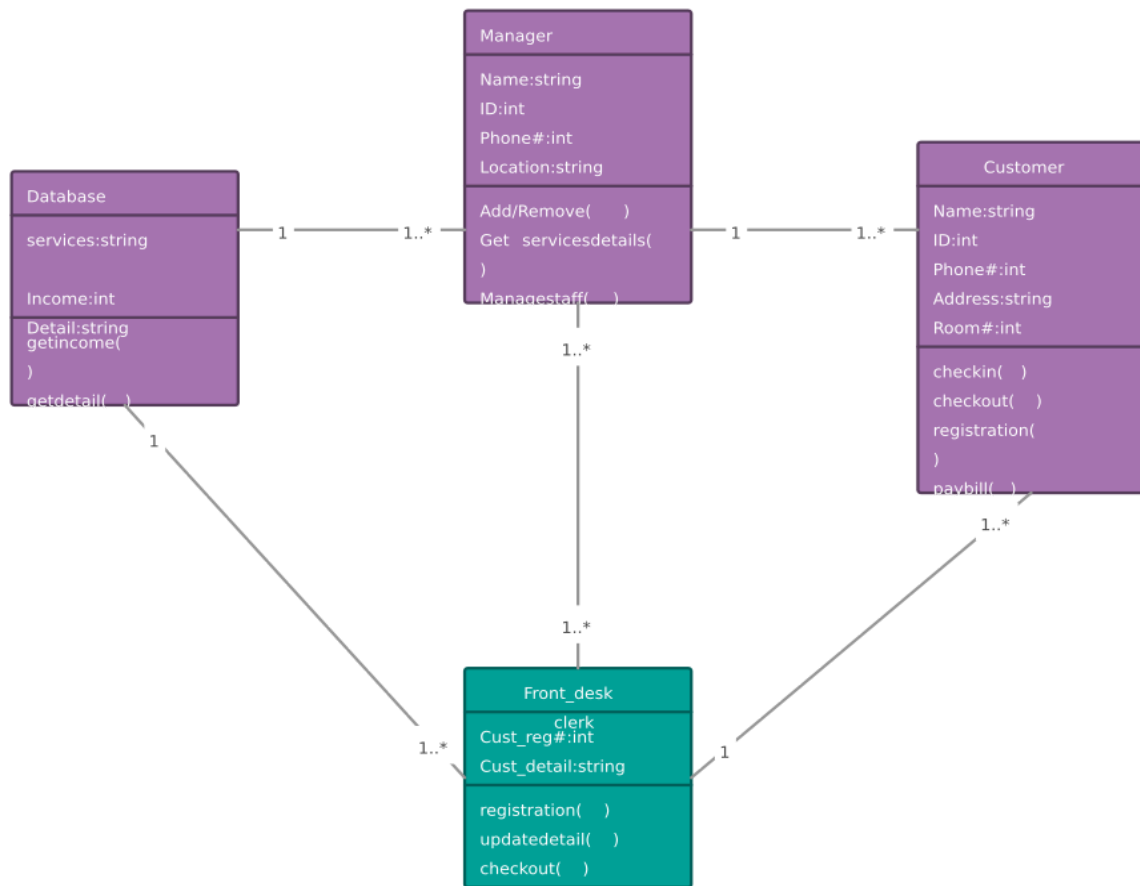
CHAPTER 2

2.1 USE CASE DIAGRAM

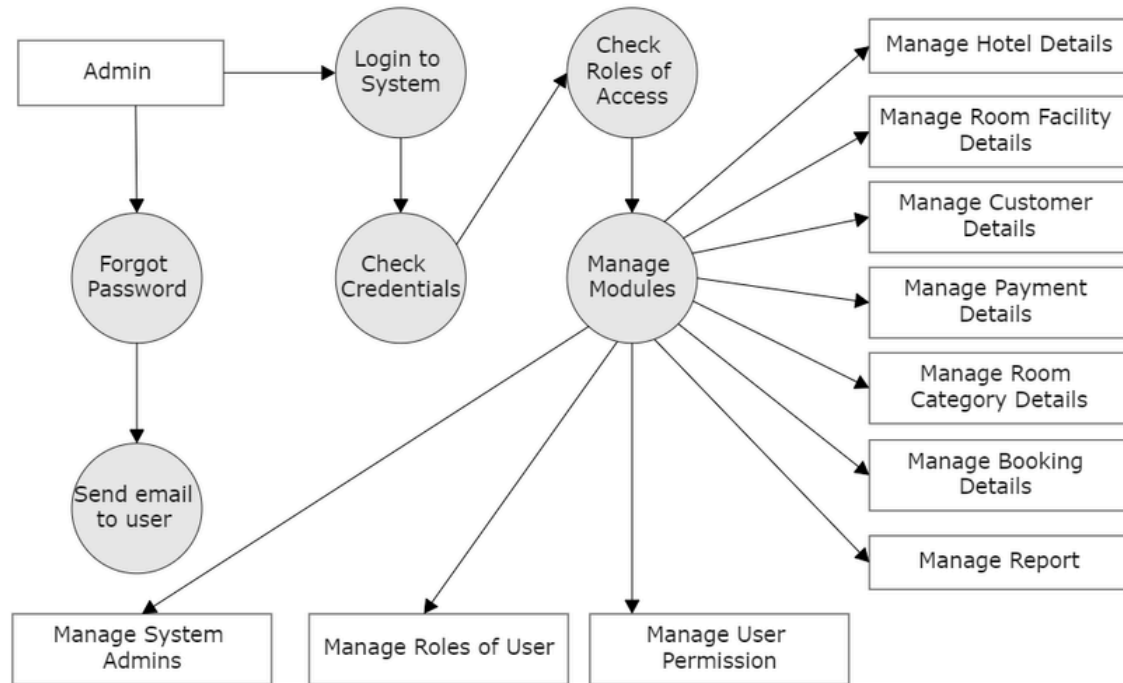
UML Use Case Diagram For Hotel Management System



2.1 DATA FLOW DIADRAM



Second Level DFD- Hotel Room System

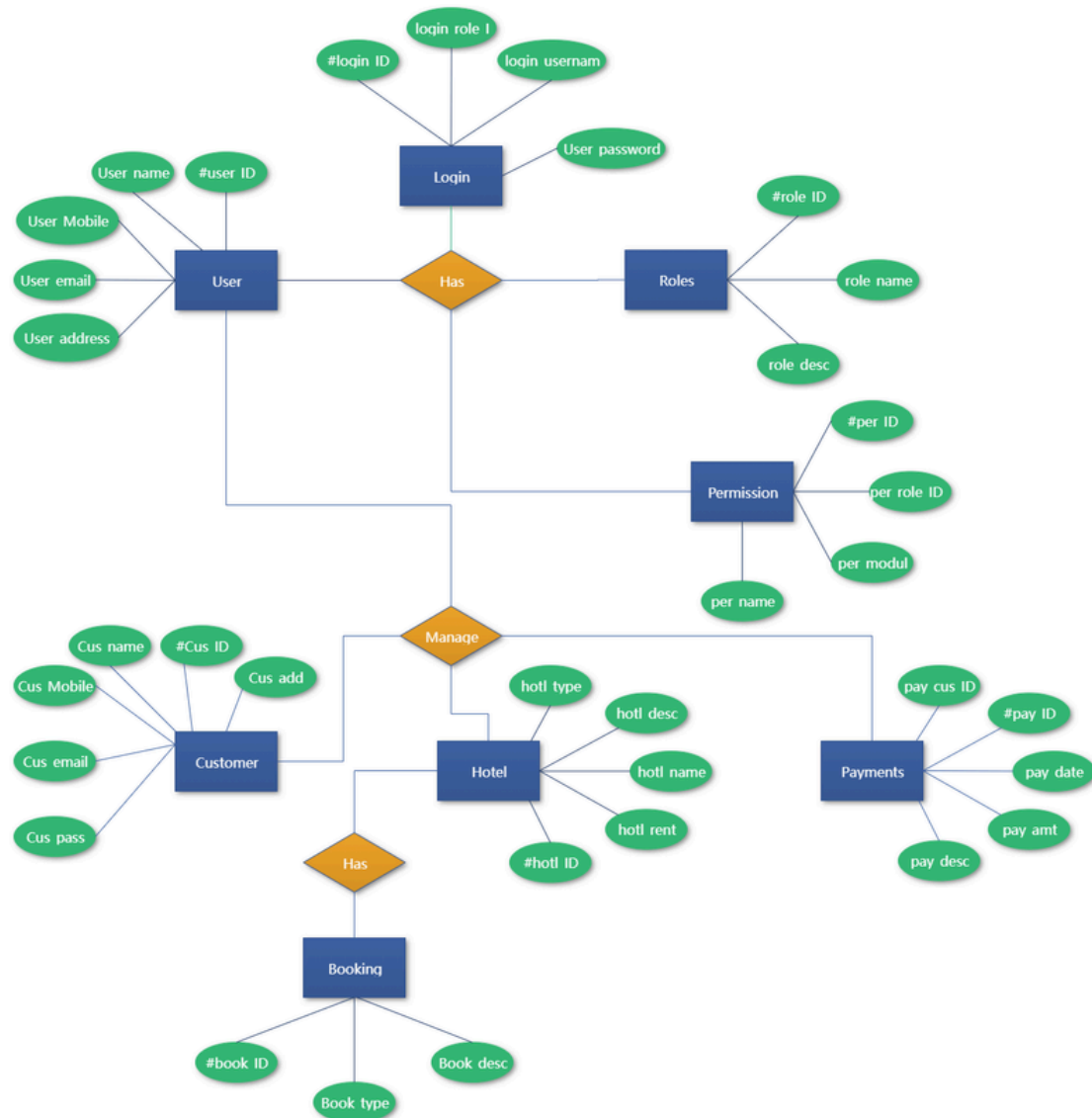


2.3 ENTITY RELATIONSHIP DIAGRAM

E-R (Entity Relationship) Diagram is used to represent the relationship between entities in table.

ENTITY RELATIONSHIP DIAGRAM FOR ADMIN

ER diagram for Hotel Management System



CHAPTER 3

3.1 MODULE DESCRIPTION

The Hotel Management System (HMS) is designed to simplify the operations of a hotel by automating tasks related to room reservations, guest management, billing, and room availability. The system is divided into several functional modules that address specific aspects of hotel management. Below is a detailed description of each module in the system.

1. Room Reservation Module

Purpose:

The **Room Reservation Module** allows guests to view available rooms and make reservations. It is the first step in the process of booking a stay at the hotel.

Key Features:

- **View Room Availability:** Displays available rooms based on the type of room, price, and availability.
- **Make Reservation:** Allows the guest to select a room, enter personal information (name, contact details, check-in/check-out dates), and complete the reservation process.
- **Reservation Confirmation:** After a reservation is successfully made, the system generates a confirmation with the reservation ID, guest details, and room information.

Interactions:

- Guests can view available rooms and make reservations.
 - The system checks room availability and confirms bookings.
 - The system saves reservation details, such as guest contact information and the room assigned.
-

2. Check-in/Check-out Module

Purpose:

The **Check-in/Check-out Module** manages the process of guest check-in and check-out, including updating room status, calculating the bill, and managing the guest's stay.

Key Features:

- **Check-in:** At the time of guest arrival, the staff verifies the reservation, assigns the room, and marks the room as "Occupied."
 - **Check-out:** When the guest leaves, the system calculates the final bill based on room charges and additional services. It also updates the room status to "Available."
 - **Guest Information:** Stores guest details during check-in, such as name, contact information, and check-in/check-out times.
-

Interactions:

- Hotel staff handles the check-in process by assigning the guest to a room and marking it as occupied.
 - When the guest checks out, the staff generates the bill, which includes charges for the room and any other services used.
 - The system updates the room status to available after check-out.
-

3. Room Management Module

Purpose:

The **Room Management Module** is responsible for managing the hotel's rooms, including the availability, status, and other details of each room.

Key Features:

- **Room Status:** Tracks whether each room is available, occupied, or under maintenance.
- **Room Types and Pricing:** Manages different room types (e.g., Single, Double, Suite) and their respective pricing.
- **Room Assignment:** Automatically assigns rooms to guests based on availability and the type of room selected.

Interactions:

- The hotel staff can view the status of all rooms (whether they are available or occupied).
 - The system automatically marks rooms as "Available" or "Occupied" based on the check-in/check-out status.
 - The system allows the administrator or staff to update room types, prices, or availability.
-

4. Billing and Payment Module

Purpose:

The **Billing and Payment Module** calculates the guest's total charges, generates a bill, and processes payment for the stay.

Key Features:

- **Room Charges Calculation:** Calculates the room rate based on the number of nights and the room type chosen by the guest.
- **Additional Services:** Tracks any additional services (e.g., food, laundry, or spa services) and adds them to the final bill.
- **Generate Bill:** The system generates a detailed bill that includes all charges and payment details.
- **Payment Processing:** Records the payment made by the guest, which could be through cash, card, or other methods.

Interactions:

- When the guest checks out, the system generates a bill for the room charges and any additional services.
- The system handles the calculation of the total amount and allows hotel staff to process the payment.
- Once payment is made, the system confirms the transaction and updates the guest's status.

5. Customer Management Module

Purpose:

The **Customer Management Module** stores and manages guest information, ensuring that records are accessible for reservations, billing, and personalized services.

Key Features:

- **Guest Information Storage:** Stores details such as the guest's name, contact number, email, address, and booking history.
- **Reservation History:** Tracks past reservations, check-in/check-out dates, and room assignments.
- **Guest Preferences:** Stores information about guest preferences, such as special requests (e.g., extra pillows, non-smoking room).

Interactions:

- Hotel staff can access guest information to process check-ins, confirm reservations, and handle inquiries.
- The system stores and updates guest details during the check-in process.

- Guest preferences are stored for future reference and can be used to provide a more personalized experience during their stay.
-

6. Admin/Settings Module

Purpose:

The **Admin/Settings Module** is designed for managing system configurations, including room inventory, pricing, user access, and other system settings.

Key Features:

- **Add/Remove Rooms:** The administrator can add new rooms or remove existing ones from the system.
- **Pricing Management:** The administrator can adjust the pricing of rooms based on factors like season, demand, or room type.
- **User Roles and Access:** Manages user roles (such as hotel staff and administrators) and their permissions for accessing certain system features.
- **System Configurations:** Allows the admin to configure system settings, such as default billing settings, language, and currency.

Interactions:

- The administrator can modify room types, add new rooms, and update the room prices.
 - User roles and permissions are managed by the admin to ensure that only authorized personnel can access sensitive information.
 - The admin has access to system settings to configure or update any global parameters for the hotel.
-

7. Reporting Module (Optional)

Purpose:

The **Reporting Module** generates reports related to hotel operations, including booking summaries, financial reports, and room occupancy statistics.

Key Features:

- **Booking Report:** Generates reports on reservations, including the number of bookings, booking dates, and guest information.
 - **Financial Report:** Summarizes the income from guest payments, additional services, and any other sources of revenue.
-

- **Occupancy Report:** Displays the occupancy rates for each room type, providing insight into which rooms are in demand.

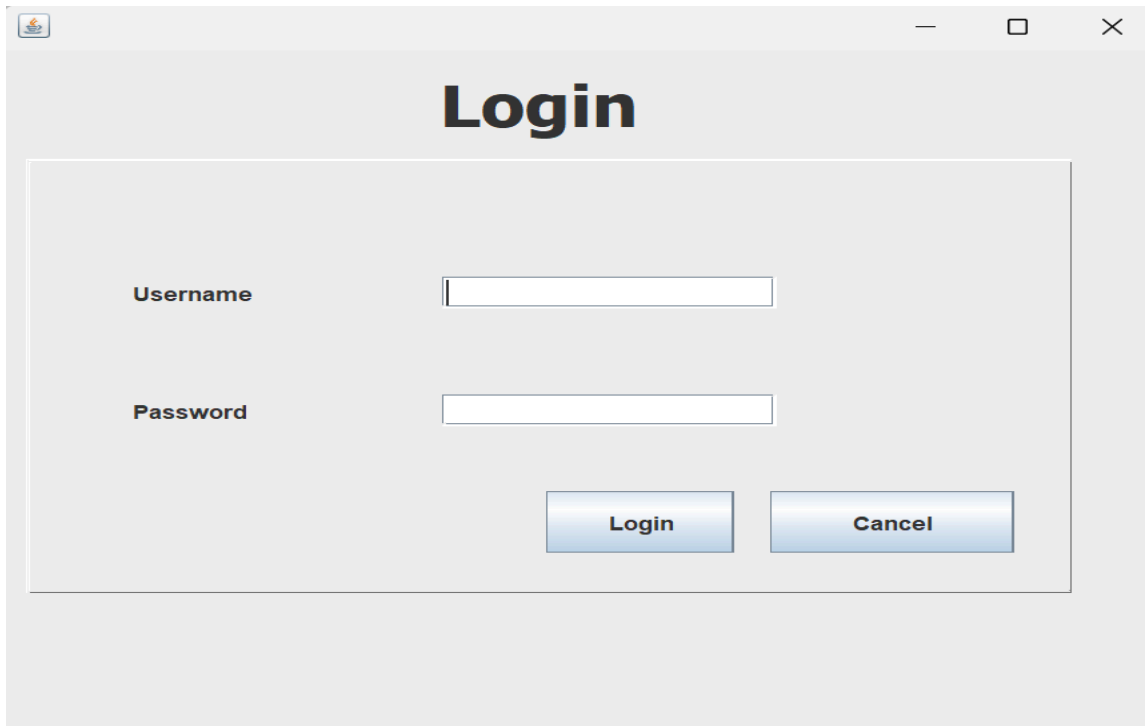
Interactions:

- The admin or hotel staff can generate reports to monitor hotel performance, including financial data and room occupancy.
- Reports can be filtered by date, room type, or guest to provide detailed insights into hotel operations.

CHAPTER 4

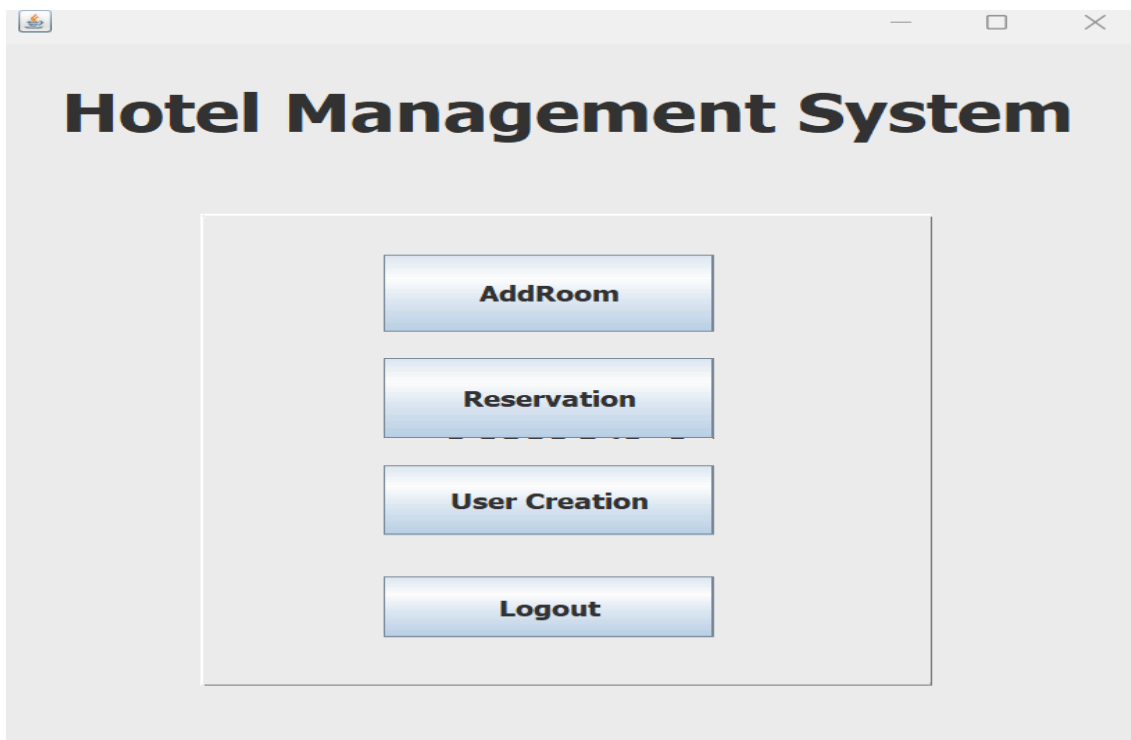
4.1 Implementation of Design:

- Home/Login page:



A screenshot of a 'Login' window. The window has a title bar with a small icon on the left and standard minimize, maximize, and close buttons on the right. The main content area is light gray and contains a white rectangular box. Inside this box, the word 'Login' is centered at the top in a large, bold, black font. Below the title, there are two labels: 'Username' and 'Password', each followed by a white text input field. At the bottom of the white box, there are two blue buttons with white text: 'Login' and 'Cancel'.

Main page:



A screenshot of the 'Hotel Management System' main page. The window has a title bar with a small icon on the left and standard minimize, maximize, and close buttons on the right. The main content area is light gray and contains a large, bold, black title 'Hotel Management System' at the top. Below the title, there is a white rectangular box. Inside this box, there are four blue buttons with white text, stacked vertically: 'AddRoom', 'Reservation', 'User Creation', and 'Logout'.

4.2 Database Design:

The Database Design for the Hotel Management System (HMS) is an essential part of the system that ensures efficient data storage, retrieval, and management. The database will store various types of information such as room details, guest information, reservations, billing data, and more.

In this design, we will be using a relational database management system (RDBMS) such as MySQL, SQLite, or PostgreSQL. The main goal is to create an optimized schema with normalized tables to avoid data redundancy and ensure data integrity.

Entities and Tables

The database will consist of the following main entities (tables):

1. Guest Table: To store information about guests.
2. Room Table: To store details about rooms in the hotel.
3. Reservation Table: To manage reservations made by guests.
4. Check-in/Check-out Table: To handle check-in and check-out transactions.
5. Billing Table: To store billing details of guests.
6. Room Service Table (optional): To store additional services availed by the guest.
7. User Table: To store hotel staff and administrator details (for login).

Each table will have a set of attributes (columns) that will store the data associated with that entity. Here's a breakdown of the database schema and relationships between the tables:

1. Guest Table

Table Name: **Guests**

| Column Name | Data Type | Description |
|-------------|--------------|---------------------------------------|
| GuestID | INT | Primary Key, unique ID for each guest |
| FirstName | VARCHAR(100) | First name of the guest |
| LastName | VARCHAR(100) | Last name of the guest |
| Email | VARCHAR(100) | Email address of the guest |
| PhoneNumber | VARCHAR(15) | Contact phone number of the guest |
| Address | TEXT | Residential address of the guest |

2. Room Table

Table Name: `Rooms`

| Column Name | Data Type | Description |
|----------------------------|--|--|
| <code>RoomID</code> | INT | Primary Key, unique ID for each room |
| <code>RoomType</code> | VARCHAR(50) | Type of the room (e.g., Single, Double, Suite) |
| <code>PricePerNight</code> | DECIMAL(10, 2) | Price per night for the room |
| <code>Status</code> | ENUM('Available', 'Occupied', 'Maintenance') | Current status of the room |
| <code>Capacity</code> | INT | Number of people the room can accommodate |



3. Reservation Table

Table Name: `Reservations`

| Column Name | Data Type | Description |
|----------------------------|---|---|
| <code>ReservationID</code> | INT | Primary Key, unique ID for each reservation |
| <code>GuestID</code> | INT | Foreign Key referencing <code>Guests.GuestID</code> |
| <code>RoomID</code> | INT | Foreign Key referencing <code>Rooms.RoomID</code> |
| <code>CheckInDate</code> | DATE | The date the guest will check-in |
| <code>CheckOutDate</code> | DATE | The date the guest will check-out |
| <code>Status</code> | ENUM('Confirmed', 'Cancelled', 'Completed') | Reservation status |

4. Check-in/Check-out Table

Table Name: `CheckInOut`

| Column Name | Data Type | Description |
|----------------------------|-----------|---|
| <code>CheckInOutID</code> | INT | Primary Key, unique ID for each check-in/check-out |
| <code>ReservationID</code> | INT | Foreign Key referencing <code>Reservations.ReservationID</code> |
| <code>CheckInDate</code> | DATE | Date and time when the guest checked in |
| <code>CheckOutDate</code> | DATE | Date and time when the guest checked out |
| <code>RoomID</code> | INT | Foreign Key referencing <code>Rooms.RoomID</code> |

5. Billing Table

Table Name: `Billing`

| Column Name | Data Type | Description |
|----------------------------|-------------------------|---|
| <code>BillID</code> | INT | Primary Key, unique ID for each bill |
| <code>ReservationID</code> | INT | Foreign Key referencing <code>Reservations.ReservationID</code> |
| <code>TotalAmount</code> | DECIMAL(10, 2) | Total amount of the bill |
| <code>PaymentStatus</code> | ENUM('Pending', 'Paid') | Payment status (pending or completed) |
| <code>PaymentMethod</code> | VARCHAR(50) | Method of payment (e.g., Cash, Card, Online) |
| <code>PaymentDate</code> | DATE | The date when payment was made |

6. Room Service Table (Optional)

Table Name: `RoomService`

| Column Name | Data Type | Description |
|----------------------------|----------------|---|
| <code>ServiceID</code> | INT | Primary Key, unique ID for each service |
| <code>ReservationID</code> | INT | Foreign Key referencing <code>Reservations.ReservationID</code> |
| <code>ServiceName</code> | VARCHAR(100) | Name of the service availed (e.g., Laundry, Food) |
| <code>Amount</code> | DECIMAL(10, 2) | Charge for the service |
| <code>ServiceDate</code> | DATE | Date the service was requested |

7. User Table (Hotel Staff)

Table Name: `Users`

| Column Name | Data Type | Description |
|------------------------|------------------------|--|
| <code>UserID</code> | INT | Primary Key, unique ID for each user (staff/admin) |
| <code>FirstName</code> | VARCHAR(100) | First name of the user |
| <code>LastName</code> | VARCHAR(100) | Last name of the user |
| <code>Email</code> | VARCHAR(100) | Email address for login |
| <code>Password</code> | VARCHAR(255) | Encrypted password for login |
| <code>Role</code> | ENUM('Admin', 'Staff') | Role of the user (Admin or Hotel Staff) |

4.3 Implementation of Code:

```
package HotelManagement;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 * @author Thillai
 */
public final class reservation extends javax.swing.JFrame {

    public reservation() {
        initComponents();
        Connect();
        autoID();
        RoomTypeL();
        RoomNo();
        BedType();
        Load_reservation();
    }

    Connection con;
```

```

PreparedStatement pst;

DefaultTableModel d;

public void Connect() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");

        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/hotelmanagement", "root",
"Akas@2005");
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public void autoID() {
    try {
        Statement s = con.createStatement();

        ResultSet rs = s.executeQuery("select MAX(reid) from reservation");

        rs.next();

        rs.getString("MAX(reid)");

        if (rs.getString("MAX(reid)") == null) {
            jLabel12.setText("RE001");
        } else {
            long id = Long.parseLong(rs.getString("MAX(reid)").substring(2,
rs.getString("MAX(reid)").length()));

            id++;

            jLabel12.setText("RE" + String.format("%03d", id));
        }
    } catch (SQLException ex) {
        Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

```

public void RoomTypeL() {
    try {
        pst = con.prepareStatement("select Distinct rtype from room");
        ResultSet rs = pst.executeQuery();
        txtrtype.removeAllItems();
        while(rs.next()) {
            txtrtype.addItem(rs.getString("rtype"));
        }
    } catch (SQLException ex) {
        Logger.getLogger(reservation.class.getName()).log(Level.SEVERE, null, ex);
    }
}

@SuppressWarnings("unchecked")
public void Load_reservation() {
    int c;
    try {
        pst = con.prepareStatement("select * from reservation");
        ResultSet rs = pst.executeQuery();
        ResultSetMetaData rsd = rs.getMetaData();
        c = rsd.getColumnCount();
        d = (DefaultTableModel) jTable1.getModel();
        d.setRowCount(0);
        while (rs.next()) {
            Vector v2 = new Vector();
            for (int i = 1; i <= c; i++) {
                boolean add = v2.add(rs.getString("reid"));
                v2.add(rs.getString("name"));
                v2.add(rs.getString("mobile"));
                v2.add(rs.getString("checkin"));
                v2.add(rs.getString("checkout"));
                v2.add(rs.getString("rtype"));
            }
        }
    }
}

```

```

        v2.add(rs.getString("roomno"));

        v2.add(rs.getString("bedtype"));

        v2.add(rs.getString("amount"));

    }

    d.addRow(v2);

}

} catch (SQLException ex) {

    Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);

}

}

public void RoomNo() {

    try {

        pst = con.prepareStatement("select Distinct rid from room");

        ResultSet rs = pst.executeQuery();

        txtro.removeAllItems();

        while(rs.next()) {

            txtro.addItem(rs.getString("rid"));

        }

    } catch (SQLException ex) {

        Logger.getLogger(reservation.class.getName()).log(Level.SEVERE, null, ex);

    }

}

public void BedType() {

    try {

        pst = con.prepareStatement("select Distinct btype from room");

        ResultSet rs = pst.executeQuery();

        txtbtype.removeAllItems();

        while(rs.next())

        {

            txtbtype.addItem(rs.getString("btype"));

        }

    }

}

```

```

    } catch (SQLException ex) {

        Logger.getLogger(reservation.class.getName()).log(Level.SEVERE, null, ex);

    }

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jPanel2 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jLabel9 = new javax.swing.JLabel();
    jLabel10 = new javax.swing.JLabel();
    jLabel11 = new javax.swing.JLabel();
    txtname = new javax.swing.JTextField();
    txtaddress = new javax.swing.JTextField();
    txtmobile = new javax.swing.JTextField();
    txtcheckin = new com.toedter.calendar.JDateChooser();
    txtcheckout = new com.toedter.calendar.JDateChooser();
    txtrtype = new javax.swing.JComboBox<>();

```



```

txtro = new javax.swing.JComboBox<>();
txtbtype = new javax.swing.JComboBox<>();
txtamount = new javax.swing.JTextField();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
jButton4 = new javax.swing.JButton();
jScrollPane1 = new javax.swing.JScrollPane();
jTable1 = new javax.swing.JTable();
jLabel12 = new javax.swing.JLabel();
jButton5 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jPanel2.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N
jLabel1.setText("Reservation");

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel2.setText("Reservation No");

jLabel3.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel3.setText("Name");

jLabel4.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel4.setText("Address");

jLabel5.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel5.setText("Mobile");

```

```
jLabel6.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel6.setText("Check In");
```

```
jLabel7.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel7.setText("Check Out");
```

```
jLabel8.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel8.setText("Room Type");
```

```
jLabel9.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel9.setText("Room No");
```

```
jLabel10.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel10.setText("Bed Type");
```

```
jLabel11.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel11.setText("Amount");
```

```
txtbtype.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        txtbtypeActionPerformed(evt);
    }
});
```

```
jButton1.setText("Save");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
```

```

jButton2.setText("Edit");

jButton3.setText("Delete");

jButton4.setText("Clear");

jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {
        "RID", "Name", "Mobile", "CheckIn", "CheckOut", "RoomType", "RoomNo", "BedType",
"Amount"
    }
) {
    Class[] types = new Class [] {
        java.lang.String.class, java.lang.String.class, java.lang.String.class, java.lang.String.class,
java.lang.String.class, java.lang.Object.class, java.lang.Object.class, java.lang.Object.class,
java.lang.Object.class
    };

    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});

jScrollPane1.setViewportView(jTable1);

jLabel12.setFont(new java.awt.Font("Tahoma", 1, 13)); // NOI18N
jLabel12.setForeground(new java.awt.Color(204, 0, 0));
jLabel12.setText("jLabel12");

```



```

        .addComponent(txtname)

        .addComponent(txtmobile)

        .addComponent(txtcheckin, javax.swing.GroupLayout.DEFAULT_SIZE, 198,
Short.MAX_VALUE)

        .addComponent(txtcheckout, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(txtrtype, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

        .addComponent(txtamount)

        .addComponent(txtbtype, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

        .addComponent(txtro, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))

    .addGroup(jPanel2Layout.createSequentialGroup())

        .addComponent(jLabel2)

        .addGap(18, 18, 18)

        .addComponent(jLabel12)))

    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(jPanel2Layout.createSequentialGroup())

            .addGap(54, 54, 54)

            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 101,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(18, 18, 18)

            .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 94,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(18, 18, 18)

            .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 94,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

            .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

            .addComponent(jButton5)

```

```

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup())

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 13,
Short.MAX_VALUE)

        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 621,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addContainerGap()))))

    .addGroup(jPanel2Layout.createSequentialGroup())

        .addGap(289, 289, 289)

        .addComponent(jLabel1)

        .addGap(0, 0, Short.MAX_VALUE))

    );

jPanel2Layout.setVerticalGroup(

    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(jPanel2Layout.createSequentialGroup())

            .addContainerGap()

            .addComponent(jLabel1)

            .addGap(57, 57, 57)

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(jLabel2)

            .addComponent(jLabel12))

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(jPanel2Layout.createSequentialGroup())

                .addGap(25, 25, 25)

            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

                .addGroup(jPanel2Layout.createSequentialGroup())

            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

                .addGroup(jPanel2Layout.createSequentialGroup())

```

```

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel3)
    .addComponent(txtname, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(22, 22, 22)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel4)
    .addComponent(txtaddress,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(21, 21, 21)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel5)
    .addComponent(txtmobile,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(24, 24, 24)
.addComponent(jLabel6))
.addComponent(txtcheckin, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(33, 33, 33)
.addComponent(jLabel7))
.addComponent(txtcheckout, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(26, 26, 26)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel8)
    .addComponent(txtrtype, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(25, 25, 25)

```

```

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel9)
    .addComponent(txtro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(18, 18, 18)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel10)
    .addComponent(txtbtype, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(35, 35, 35)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup())

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
    .addComponent(jButton2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
    .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jButton4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(jButton5)))
.addGap(35, 35, 35))
.addGroup(jPanel2Layout.createSequentialGroup())

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel11)

```



```

        .addComponent(txtamount, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))

    .addGroup(jPanel2Layout.createSequentialGroup())

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 328,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addGap(0, 0, Short.MAX_VALUE))))

    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
            );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel2, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            );

    pack();

    setLocationRelativeTo(null);
} // </editor-fold>

private void txtbtypeActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here

```

```

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String reno = jLabel12.getText();
    String name = txtname.getText();
    String address = txtaddress.getText();
    String mobile = txtmobile.getText();
    SimpleDateFormat df1 = new SimpleDateFormat("yyyy-MM-dd");
    String StartDate = df1.format(txtcheckin.getDate());
    SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd");
    String EndDate = df1.format(txtcheckout.getDate());
    String rtype = txtrtype.getSelectedItem().toString();
    String roomno = txtro.getSelectedItem().toString();
    String bedtype = txtbtype.getSelectedItem().toString();
    String amount = txtamount.getText();
    try {
        pst = con.prepareStatement("insert into
reservation(reid,name,address,mobile,checkin,checkout,bedtype,roomno,rtype,amount)
values(?,?,?,?,?,?,?,?,?,?)");
        pst.setString(1, reno);
        pst.setString(2, name);
        pst.setString(3, address);
        pst.setString(4, mobile);
        pst.setString(5, StartDate);
        pst.setString(6, EndDate);
        pst.setString(7, bedtype);
        pst.setString(8, roomno);
        pst.setString(9, rtype);
        pst.setString(10, amount);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, "Reservation Added");
    }
}

```

```

        txtname.setText("");
        txtaddress.setText("");
        txtmobile.setText("");
        txtaddress.setText("");
        txtmobile.setText("");
        txtrtype.setSelectedIndex(-1);
        txtro.setSelectedIndex(-1);
        txtbtype.setSelectedIndex(-1);
        txtamount.setText("");
        autoID();
        // Load_room();

    } catch (SQLException ex) {
        Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.setVisible(false);

    // String roomno = jLabel6.getText();
    // String roomtype = txtrtype.getSelectedItem().toString();
    // String bedtype = txtbtype.getSelectedItem().toString();
    // String amount = txtamount.getText();
    // String name = txtamount.getText();
    // String mobile = txtamount.getText();
    // try {
    //     pst = con.prepareStatement("update room set roomno = ?,rtype = ?, bedtype = ? , amount = ?
    // ,amount= ?, mobile = ?, name = ?, where reid = ?");
    //     pst.setString(1, roomtype);
    //     pst.setString(2, bedtype);
    //     pst.setString(3, amount);

```

```

//      pst.setString(4, roomno);
//      pst.setString(5, name);
//      pst.setString(6, roomno);
//      pst.setString(7, mobile);
//      pst.executeUpdate();
//      JOptionPane.showMessageDialog(this, "Room Edited");
//      txttrtype.setSelectedIndex(-1);
//      txtbtype.setSelectedIndex(-1);
//      txtamount.setText("");
//      autoID();
//      //Load_room();
//      jButton1.setEnabled(true);
//  } catch (SQLException ex) {
//      Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
//  }

```

```

}

```

```

/**

```

```

 * @param args the command line arguments

```

```

 */

```

```

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(() -> {
        new reservation().setVisible(true);
    });
}

```

```

// Variables declaration - do not modify

```

```

private javax.swing.JButton jButton1;

```

```

private javax.swing.JButton jButton2;

```

```

private javax.swing.JButton jButton3;

```

```
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTextField txtaddress;
private javax.swing.JTextField txtamount;
private javax.swing.JComboBox<String> txtbtype;
private com.toedter.calendar.JDateChooser txtcheckin;
private com.toedter.calendar.JDateChooser txtcheckout;
private javax.swing.JTextField txtmobile;
private javax.swing.JTextField txtname;
private javax.swing.JComboBox<String> txtro;
private javax.swing.JComboBox<String> txtrtype;
// End of variables declaration
}
```

```

package HotelManagement;

/**
 * @author Thillai
 */

public class Main extends javax.swing.JFrame {

    public Main() {
        initComponents();
        //Login();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N
        jLabel1.setText("Hotel Management System");

        jPanel1.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

        jButton1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jButton1.setText("AddRoom");

```

```
jButton1.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton1ActionPerformed(evt);  
    }  
});
```

```
jButton2.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N  
jButton2.setText("Reservation");  
jButton2.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton2ActionPerformed(evt);  
    }  
});
```

```
jButton3.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N  
jButton3.setText("User Creation");  
jButton3.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton3ActionPerformed(evt);  
    }  
});
```

```
jButton4.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N  
jButton4.setText("Logout");  
jButton4.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton4ActionPerformed(evt);  
    }  
});
```

```
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
```

```

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(85, 85, 85)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

            .addComponent(jButton4, javax.swing.GroupLayout.DEFAULT_SIZE, 159,
Short.MAX_VALUE)

            .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE, 159,
Short.MAX_VALUE)

            .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addContainerGap(102, Short.MAX_VALUE))
);

jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(25, 25, 25)

            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 53,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(18, 18, 18)

            .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(18, 18, 18)

            .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(28, 28, 28)

            .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addContainerGap(30, Short.MAX_VALUE))

```



```

);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(28, 28, 28)
                    .addComponent(jLabel1))
                .addGroup(layout.createSequentialGroup()
                    .addGap(95, 95, 95)
                    .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addGap(34, Short.MAX_VALUE))
        );
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(24, 24, 24)
            .addComponent(jLabel1)
            .addGap(48, 48, 48)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(61, Short.MAX_VALUE))
        );

pack();

setLocationRelativeTo(null);
} // </editor-fold>

```

```

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    Login l = new Login();
    this.setVisible(true);
    l.setVisible(true);

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    room r = new room();
    r.setVisible(true);

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    reservation r = new reservation();
    r.setVisible(true);

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

```

```

        user r = new user();
        r.setVisible(true);

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
                javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

            java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null,
                ex);

        } catch (InstantiationException ex) {

```

```
java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
```

```
    } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
```

```
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
```

```
    }
```

```
//</editor-fold>
```

```
/* Create and display the form */
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
```

```
    public void run() {
```

```
        new Main().setVisible(true);
```

```
    }
```

```
});
```

```
}
```

```
// Variables declaration - do not modify
```

```
private javax.swing.JButton jButton1;
```

```
private javax.swing.JButton jButton2;
```

```
private javax.swing.JButton jButton3;
```

```
private javax.swing.JButton jButton4;
```

```
private javax.swing.JLabel jLabel1;
```

```
private javax.swing.JPanel jPanel1;
```

```
// End of variables declaration
```

```
}
```

```
package HotelManagement;
```

```

/**
 * @author Thillai
 */

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class user extends javax.swing.JFrame {

    public user() {
        initComponents();
        Connect();
        Load_user();
    }

    Connection con;

    PreparedStatement pst;

    DefaultTableModel d;

    public void Connect() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/hotelmanagement", "root",
"Akas@2005");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {

```

```

        Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public void Load_user() {
    int c;
    try {
        pst = con.prepareStatement("select * from user");
        ResultSet rs = pst.executeQuery();
        ResultSetMetaData rsd = rs.getMetaData();
        c = rsd.getColumnCount();
        d = (DefaultTableModel)jTable1.getModel();
        d.setRowCount(0);
        while(rs.next()) {
            Vector v2 = new Vector();
            for(int i =1; i<=c; i++) {
                v2.add(rs.getString("uid"));
                v2.add(rs.getString("name"));
                v2.add(rs.getString("username"));
            }
            d.addRow(v2);
        }
    } catch (SQLException ex) {
        Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
    }
}

/**
 * WARNING: Do NOT modify this code. The content of this method is always
 */

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

```

```

jPanel1 = new javax.swing.JPanel();
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
txtname = new javax.swing.JTextField();
txtuname = new javax.swing.JTextField();
txtpass = new javax.swing.JPasswordField();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
jButton4 = new javax.swing.JButton();
jButton5 = new javax.swing.JButton();
jScrollPane1 = new javax.swing.JScrollPane();
jTable1 = new javax.swing.JTable();
jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jPanel1.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

jLabel2.setText("Name");

jLabel3.setText("Username");

jLabel4.setText("Password");

jButton1.setText("Add");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {

```

```

        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("Modify");

jButton3.setText("Remove");

jButton4.setText("Clear");

jButton5.setText("Close");
jButton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton5ActionPerformed(evt);
    }
});

jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {
        "UserID", "Name", "Username"
    }
) {
    Class[] types = new Class [] {
        java.lang.String.class, java.lang.Object.class, java.lang.Object.class
    };

    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});

```



```

        .addComponent(jLabel3))

        .addComponent(txtuname, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGap(35, 35, 35)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

        .addComponent(jLabel4)

        .addComponent(txtpass, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGap(42, 42, 42)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

        .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE, 41,
Short.MAX_VALUE)

        .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))

        .addGroup(jPanel1Layout.createSequentialGroup()

            .addContainerGap()

            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 235,
javax.swing.GroupLayout.PREFERRED_SIZE)))

        .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

        .addComponent(jButton5, javax.swing.GroupLayout.DEFAULT_SIZE, 43,
Short.MAX_VALUE)

        .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(jButton4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

        .addContainerGap(28, Short.MAX_VALUE))

);

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

```

```

jLabel1.setText("User Creation");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(180, 180, 180)
            .addComponent(jLabel1)
            .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap())
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(31, 31, 31)
            .addComponent(jLabel1)
            .addGap(33, 33, 33)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(45, Short.MAX_VALUE))
);

pack();
setLocationRelativeTo(null);
} // </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

```

```

// TODO add your handling code here:

String name = txtname.getText();
String username = txtuname.getText();
String password = txtpass.getText();

try {

    pst = con.prepareStatement("insert into user(name,username,password) values(?,?,?)");
    pst.setString(1, name);
    pst.setString(2, username);
    pst.setString(3, password);
    pst.executeUpdate();

    JOptionPane.showMessageDialog(this, "User Added");

    txtname.setText("");
    txtuname.setText("");
    txtpass.setText("");
    txtname.requestFocus();

    //Load_room();
} catch (SQLException ex) {

    Logger.getLogger(room.class.getName()).log(Level.SEVERE, null, ex);
}

}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    this.setVisible(false);

}

/**
 * @param args the command line arguments
 */

public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {

            new user().setVisible(true);

```

```

        }
    });
}

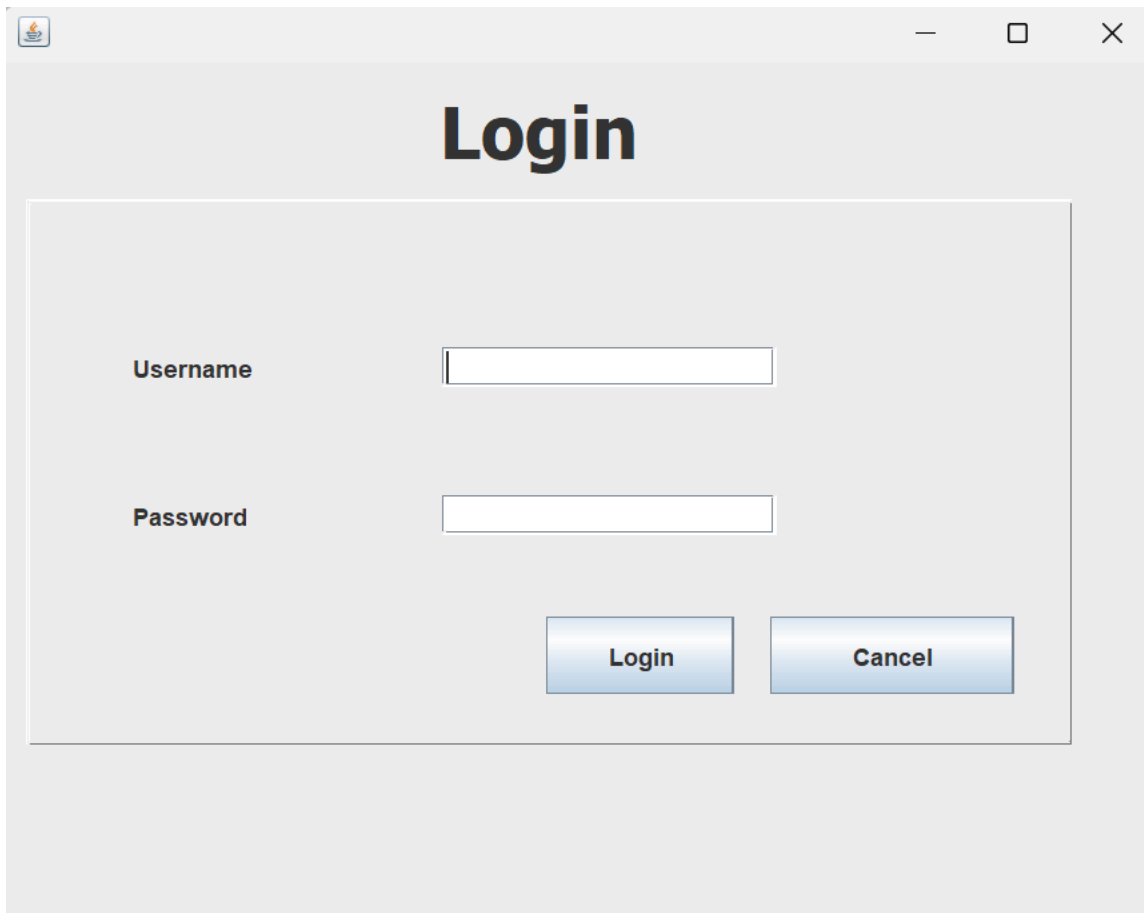
// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTextField txtname;
private javax.swing.JPasswordField txtpass;
private javax.swing.JTextField txtuname;

// End of variables declaration
}

```

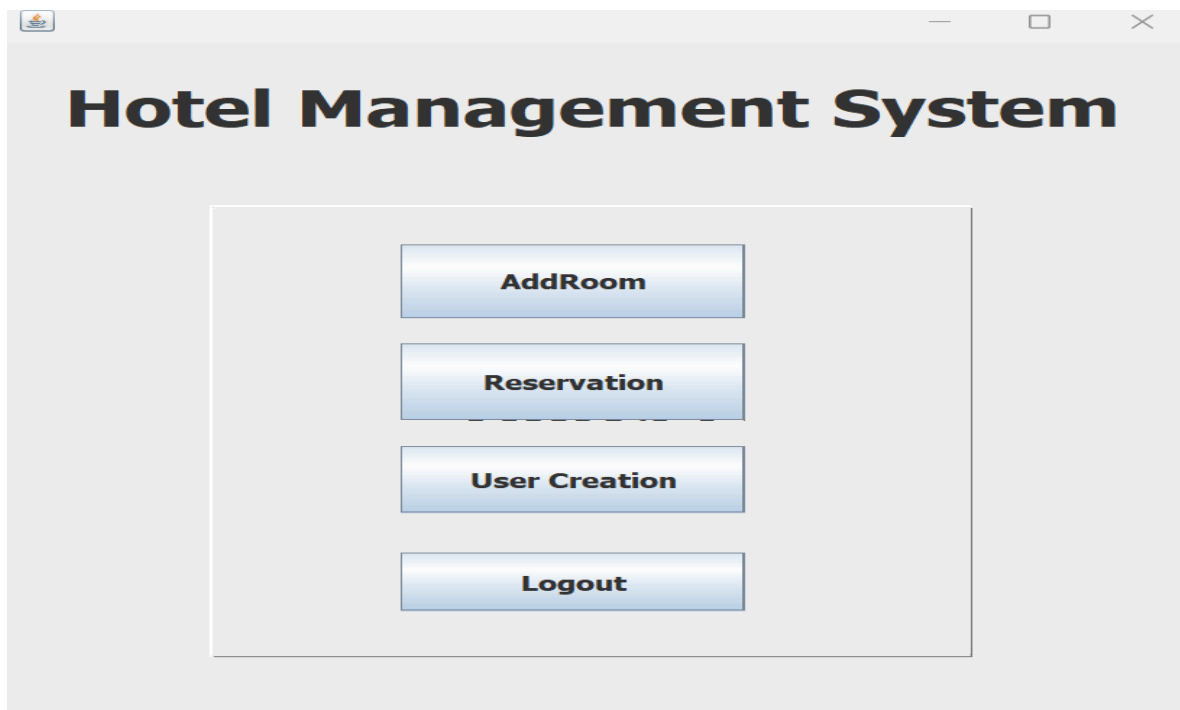
4.4 OUTPUT SCREEN

Login Page:



A screenshot of a web application window titled "Login". The window has a standard OS-style title bar with a small icon on the left and minimize, maximize, and close buttons on the right. The main content area has a light gray background. At the top center, the word "Login" is displayed in a large, bold, black font. Below this, there is a white rectangular box containing the login form. Inside the box, the label "Username" is followed by a text input field. Below that, the label "Password" is followed by a text input field. At the bottom right of the box, there are two blue buttons with white text: "Login" and "Cancel".

Navigation Page:



A screenshot of a web application window titled "Hotel Management System". The window has a standard OS-style title bar with a small icon on the left and minimize, maximize, and close buttons on the right. The main content area has a light gray background. At the top center, the text "Hotel Management System" is displayed in a large, bold, black font. Below this, there is a white rectangular box containing four blue buttons with white text, stacked vertically: "AddRoom", "Reservation", "User Creation", and "Logout".

Hotel Management System

AddRoom

Reservation

User Creation

Logout

Room
Page:

Room

Room No **R0004**

Room Type

Bed Type

Amount

| Room No | Room Type | Bed Type | Amount |
|---------|-----------|----------|--------|
| R0001 | A/C | Single | 10000 |
| R0002 | A/C | Double | 10000 |
| R0003 | NON A/C | Single | 1000 |

Save

Edit

Delete

Clear

Close

User Registration Page:

User Creation

Name

Username

Password

Add

Modify

Remove

Clear

Close

| UserID | Name | Username |
|--------|--------|----------|
| 1 | Peter | peter |
| 2 | sathis | sathis |
| 3 | Ravi | ravi |
| 4 | ji | ji |

CHAPTER 5

5.1 CONCLUSION

The Hotel Management System (HMS) developed as part of this mini project provides a comprehensive and efficient solution to the various challenges faced by hotel management. The system aims to automate and streamline key hotel operations such as room reservations, guest check-ins and check-outs, billing, and room management. This project serves as a foundational tool for hotel management, allowing hotel staff to manage their operations with greater ease and accuracy.

Key Accomplishments of the Project:

1. **Efficient Room Management:** The system allows hotel staff to track the status of rooms (available, occupied, or under maintenance), making it easier to allocate rooms to guests and manage occupancy levels in real-time.
2. **Streamlined Reservation Process:** The reservation module enables guests to view available rooms, make reservations, and receive confirmation, while hotel staff can manage and update reservations seamlessly.
3. **Automated Check-in/Check-out:** The system simplifies the check-in and check-out process, updating room availability and managing guest information efficiently, which leads to reduced errors and quicker processing times.
4. **Billing and Payment Integration:** The billing module automatically generates bills for guests, considering room charges and additional services. It supports different payment methods, ensuring that the hotel's accounting is streamlined and accurate.
5. **Data Integrity and Security:** By using a relational database design with appropriate foreign key relationships, the system ensures data consistency and integrity. Sensitive data, such as guest information and payments, is securely handled.
6. **User-friendly Interface:** The user interface has been designed to be simple and easy to navigate for both hotel staff and guests, allowing quick access to essential features without unnecessary complexity.
7. **Scalability and Flexibility:** The modular design of the system allows for easy addition of new features and can be scaled for larger hotels with more rooms and greater user interaction.

Challenges and Limitations:

- While the system is functional for a small hotel setup, it lacks advanced features such as multi-location hotel management, support for multiple languages, or integration with third-party systems for payment processing or online bookings.
- The user interface could be further enhanced to provide a more visually appealing and interactive experience.

Future Enhancements:

- **Online Booking Integration:** Adding an online booking system would allow guests to make reservations directly through a website or mobile app, improving guest experience and reducing manual work for the hotel staff.
- **Analytics and Reporting:** A more advanced reporting module could be developed to generate detailed insights into hotel performance, such as occupancy rates, financial performance, and guest preferences.
- **Mobile App Development:** To further enhance the user experience, a mobile app for guests to view room availability, make reservations, and manage their stay could be developed.
- **Integration with Payment Gateways:** To make the payment process smoother, integrating the system with online payment gateways for credit/debit card processing would be an important addition.

Final Thoughts:

The Hotel Management System (HMS) developed for this project addresses the fundamental operational needs of a hotel. By automating the core hotel management processes, it helps in reducing manual errors, improving efficiency, and enhancing customer satisfaction. With further enhancements and integration, this system has the potential to be used in larger hotel chains, offering a scalable and robust solution to hotel management. This project serves as a solid foundation for future improvements and can be expanded upon to cater to the growing needs of the hospitality industry.

CHAPTER 6

6.1 REFERENCES

1. Hotel Management System - GeeksforGeeks

- A basic outline and tutorial on how to build a hotel management system using Java, focusing on the concepts of room management, billing, and user interaction.
- [GeeksforGeeks Hotel Management System](#)
- <https://www.geeksforgeeks.org/design-online-hotel-booking-system-like-oyo-rooms/>

2. Hotel Management System - TutorialsPoint

- Provides a detailed overview of how to implement a hotel management system in Java with relevant features like booking, billing, and customer management.
- [TutorialsPoint Hotel Management System](#)
- https://www.tutorialspoint.com/build_hotel_management_system_with_tkinter_and_python_3/index.asp

3. Hotel Management System - Java Code

- A detailed example of a Java-based hotel management system with a graphical user interface (GUI). This can provide a good starting point for projects requiring user interaction.
- [Hotel Management System Java Code Example](#)
- <https://code-projects.org/hotel-management-system-in-java-with-source-code/>

4. Java Hotel Management System - Source Code

- Offers source code for a hotel management system built using Java, including database interaction for storing room and reservation information.
- [Java Hotel Management System](#)
- <https://github.com/topics/hotel-management?l=java>

5. Hotel Management System Database Design

- A detailed explanation and design of the database schema for hotel management systems. This can be used as a template to understand relationships between tables like Guest, Reservation, Room, etc.
- [Database Design for Hotel Management System](#)
- <https://vertabelo.com/blog/data-model-for-hotel-management-system/>

6. Online Hotel Management System - GitHub

- A practical project repository available on GitHub that demonstrates a hotel management system with features such as room booking, management, and admin functionalities.
- [Hotel Management System GitHub](#)
- <https://github.com/topics/hotel-management-system>

7. Hotel Management System - Tutorialspoint (MySQL Database)

- Provides insights into using MySQL for managing the data of hotel management systems, covering room allocation, guest information, billing, etc.
- [Hotel Management System MySQL Database](#)
- <https://phpgurukul.com/hotel-booking-management-system-using-php-and-mysql/>