

Status Finished**Started** Tuesday, 1 October 2024, 9:25 AM**Completed** Tuesday, 1 October 2024, 9:51 AM**Duration** 25 mins 58 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.util.Scanner;
2 public class odd{
3     public static void main(String arg[])
4     {
5         Scanner s=new Scanner(System.in);
6         int n=s.nextInt();
7         if(n%2==0 || n==0)
8         {
9             System.out.print("1");
10        }
11        else
12        {
13            System.out.print("2");
14        }
15    }
16 }
17

```

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

Passed all tests! ✓

Question 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.util.Scanner;
2 public class lastdigit
3 {
4     public static void main(String args[])
5     {
6         Scanner s= new Scanner(System.in);
7         int n = s.nextInt();
8         int lastdigit=lastdigit(n);
9         System.out.println(lastdigit);
10    }
11    public static int lastdigit(int n)
12    {
13        return Math.abs(n%10);
14    }
15 }
16

```

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

Passed all tests! ✓

Question 3

Correct

Marked out of 5.00

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

For example,

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

Below is the explanation:

Last digit of the 267 is 7

Last digit of the 154 is 4

Sum of 7 and 4 = 11

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

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

i.e.

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

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

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

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

For example:

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

Answer: (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class lasttwo
3 {
4     public static void main(String args[])
5     {
6         Scanner s= new Scanner(System.in);
7         int n1=s.nextInt();
8         int n2=s.nextInt();
9         System.out.println(Math.abs(n1%10)+Math.abs(n2%10));
10    }
11 }
```

	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-OOPUJ-2023](#) / [Lab-02-Flow Control Statements](#) / [Lab-02-Logic Building](#)

Status Finished

Started Saturday, 5 October 2024, 12:40 PM

Completed Saturday, 5 October 2024, 12:52 PM

Duration 12 mins 4 secs

Question 1

Correct

Marked out of 5.00

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

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

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

Input Format

Input containing an integer and a boolean value.

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

You have to print the time you should get up.

Example Input:

1 false

Output:

6:00

Example Input:

5 false

Output:

5:00

Example Input:

1 true

Output:

9:00

For example:

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

Answer: (penalty regime: 0 %)

```

1 import java.util.Scanner;
2
3 public class WakeUpTime {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         int day = scanner.nextInt();
7         boolean vacation = scanner.nextBoolean();
8
9         String wakeUpTime;
10
11        if (day >= 2 && day <= 6) {
12            wakeUpTime = vacation ? "7:00" : "5:00";
13        } else {
14            wakeUpTime = vacation ? "9:00" : "6:00";
15        }
16
17        System.out.println(wakeUpTime);
18    }
19 }
```

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

Passed all tests! ✓

Question 2

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 import java.util.Scanner;
2 public class FactorialTrailingZeros {
3     public static void main(String[] args) {
4         Scanner scanner = new Scanner(System.in);
5         int n = scanner.nextInt();
6         System.out.println(countTrailingZeros(n));
7         scanner.close();
8     }
9     public static int countTrailingZeros(int n) {
10        int count = 0;
11        for (int i = 5; n / i >= 1; i *= 5) {
12            count += n / i;
13        }
14        return count;
15    }
16 }
17

```

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

Passed all tests! ✓

Question 3

Correct

Marked out of 5.00

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

The movie's success formula depends on 2 parameters:

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

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

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

Write a program that takes 2 integers:

the first integer is the acting power

second integer is the critic's rating.

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

Example input:

9 5

Output:

Yes

Example input:

1 9

Output:

No

Example input:

6 4

Output:

Maybe

For example:

Input	Result
9 5	Yes
1 9	No
6 4	Maybe

Answer: (penalty regime: 0 %)

```

1 import java.util.Scanner;
2
3 public class MoviePrediction{
4     public static void main(String[] args){
5         Scanner scanner = new Scanner(System.in);
6         int actingPower = scanner.nextInt();
7         int rating=scanner.nextInt();
8         if(actingPower < 2 || rating < 2){
9             System.out.println("No");
10        }
11        else if(actingPower > 8 || rating > 8){
12            System.out.println("Yes");
13        }
14        else{
15            System.out.println("Maybe");
16        }
17    }
18 }
```

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

Passed all tests! ✓

◀ Lab-02-MCQ

Jump to...

Lab-03-MCQ ► //

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

Status Finished

Started Saturday, 5 October 2024, 12:52 PM

Completed Saturday, 5 October 2024, 1:19 PM

Duration 26 mins 57 secs

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.util.Scanner;
2 public class SumofDigitPositions{
3     public static void main(String[] args){
4         Scanner scanner=new Scanner(System.in);
5         System.out.print("");

```

```

6   int input2=scanner.nextInt();
7   int[] input1=new int[input2];
8   System.out.println("");
9   for(int i=0;i<input2;i++){
10      input1[i]=scanner.nextInt();
11   }
12   int[] digitPositions=new int[input2];
13   for(int i=0;i<input2;i++){
14      int number=input1[i];
15      int digit=(number/(int)Math.pow(10,i))%10;
16      digitPositions[i]=(number<Math.pow(10,i))?0:digit;
17   }
18   int sum=0;
19   for(int value : digitPositions){
20      sum+=value*value;
21   }
22   System.out.println(""+sum);
23   scanner.close();
24 }
25 }
26 }
27 }
```

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

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

Passed all tests! ✓

Question 3

Correct

Marked out of 5.00

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

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

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

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

input1 represents the number of elements in the array.

input2 represents the array of integers.

Example 1:

input1 = 16

input2 = {-12, -16, 12, 18, 18, 14, -4, -12, -13, 32, 34, -5, 66, 78, 78, -79}

Expected output = 62

Explanation:

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

Example 2:

input1 = 11

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

Expected output = -1

Explanation:

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

Example 3:

input1 = 16

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

Expected output = 174

Explanation:

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

For example:

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

Answer: (penalty regime: 0 %)

```

1 import java.util.Scanner;
2 public class LongestPositiveSequenceSum{
3     public static void main(String[] args){
4         Scanner scanner=new Scanner(System.in);
5         System.out.print("");
6         int input1=scanner.nextInt();
7         int[] input2=new int[input1];
8         System.out.println("");
9         for(int i=0;i<input1;i++){
10             ...
11         }
12     }
13 }
```

```

10     input2[i]=scanner.nextInt();
11
12 }
13 int maxLength=0;
14 int currentLength=0;
15 int currentSum=0;
16 int maxSum=0;
17 for(int num : input2){
18     if(num>=0){
19         currentLength++;
20         currentSum+=num;
21     }
22     else{
23         if(currentLength>maxLength){
24             maxLength=currentLength;
25             maxSum=currentSum;
26         }
27         else if(currentLength==maxLength){
28             maxSum+=currentSum;
29         }
30         currentLength=0;
31         currentSum=0;
32     }
33 }
34 if(currentLength>maxLength){
35     maxSum=currentSum;
36 }
37 else if(currentLength==maxLength){
38     maxSum+=currentSum;
39 }
40 if(maxLength==0){
41     System.out.println(-1);
42 }
43 else{
44     System.out.println(maxSum);
45 }
46 scanner.close();
47
48 }
49 }
```

	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-OOPUJ-2023](#) / [Lab-04-Classes and Objects](#) / [Lab-04-Logic Building](#)

Status Finished

Started Sunday, 6 October 2024, 11:32 PM

Completed Sunday, 6 October 2024, 11:36 PM

Duration 3 mins 38 secs

Question 1

Correct

Marked out of 5.00

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

Area of Circle = 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.Scanner;
3 class Circle
4 {
5     private double radius;
6     public Circle(double radius){
7         // set the instance variable radius
8         this.radius =radius;
9     }
10    public void setRadius(double radius){
11        // set the radius
12        this.radius=radius;
13    }
14    public double getRadius()    {
15        // return the radius
16        return radius;
17    }
18
19 }
20    public double calculateArea() { // complete the below statement
21        return Math.PI*radius*radius;
22
23    }
24    public double calculateCircumference()    {
25        // complete the statement
26        return 2*Math.PI*radius;
27    }
28 }
29 class prog{
30     public static void main(String[] args) {
31         int r;
32         Scanner sc= new Scanner(System.in);
33         r=sc.nextInt();
34         Circle c= new Circle(r);
35         System.out.println("Area = "+String.format("%.2f", c.calculateArea()));
36         // invoke the calculatecircumference method
37         System.out.println("Circumference = "+String.format("%.2f" , c.calculateCircumference()));
38
39         sc.close();
40     }
41 }
42
43

```

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

Passed all tests! ✓

Question 2

Correct

Marked out of 5.00

Create a class 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 stud{
2     private String name;
3     private int roll;
4     public stud(){
5         System.out.println("No-arg constructor is invoked");
6         name=null;
7         roll=0;
8     }
10    public stud(String name){
11        System.out.println("1 arg constructor is invoked");
12        this.name=name;
13        roll=0;
14    }
16    public stud(String name,int roll){
17        System.out.println("2 arg constructor is invoked");
18        this.name=name;
19        this.roll=roll;
20    }
22
23    public static void main (String[]args){
24        stud s1=new stud();
25        stud s2=new stud("Rajalakshmi");
26        stud s3=new stud("Lakshmi",101);
27        System.out.println("Name =" +s1.name+ " , Roll no = "+s2.roll);
28        System.out.println("Name =" +s2.name+ " , Roll no = "+s2.roll);
29        System.out.println("Name =" +s3.name+ " , Roll no = "+s3.roll);
30    }
31 }
32

```

	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 3

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 man;
3     private String os;
4     public String clr;
5     private int cost;
6     public mobile(String man,String os,String clr,int cost){
7         this.man=man;
8         this.os=os;
9         this.clr=clr;
10        this.cost=cost;
11    }
12    public String toString(){
13        return "manufacturer = "+man+"\n"+ "operating_system = "+os+"\n"+ "color = "+ clr+"\n"+ "cost = "+cost;
14    }
15    public static void main(String[]args){
16        mobile mobile=new mobile("Redmi","Andriod","Blue",34000);
17        System.out.println(mobile);
18    }
19}
20
21
```

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

Passed all tests! ✓

[◀ Lab-04-MCQ](#)

Jump to...

Number of Primes in a specified range ►

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

Status	Finished
Started	Sunday, 6 October 2024, 11:39 PM
Completed	Sunday, 6 October 2024, 11:41 PM
Duration	2 mins 27 secs

Question 1

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

```
36     System.out.println(s1.toString());  
37 }  
38 }  
39 }
```

	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 2

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 v class mob{
2 v     mob(){
3 v         System.out.println("Basic Mobile is Manufactured");
4 v     }
5 v     void basmob(){
6 v         System.out.println("Basic Mobile is Manufactured");
7 v     }
8 }
9 v class cam extends mob{
10 v    cam(){
11 v        super();
12 v        System.out.println("Camera Mobile is Manufactured");
13 v    }
14 v    void newm(){
15 v        System.out.println("Camera Mobile with 5MG px");
16 v    }
17 }
18 }
19 v class and extends cam{
20 v    and(){
21 v        super();
22 v        System.out.println("Android Mobile is Manufactured");
23 v    }
24 v    void andmob(){
25 v        System.out.println("Touch Screen Mobile is Manufactured");
26 v    }
27 }
28 v public class Main{
29 v     public static void main(String[]args){
30 v         and andmob=new and();
31 v         andmob.newm();
32 v         andmob.andmob();
33 v     }
34 }
35 }
```

	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! ✓

Question 3

Correct

Marked out of 5.00

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

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

For example:

Result

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

Answer: (penalty regime: 0 %)

[Reset answer](#)

```
1 class BankAccount {  
2     // Private field to store the account number  
3     private String accountNumber;  
4  
5     // Private field to store the balance  
6     private double balance;  
7  
8     // Constructor to initialize account number and balance  
9     public BankAccount(String accountNumber,double balance){  
10        this.accountNumber=accountNumber;  
11        this.balance=balance;  
12    }  
13  
14  
15  
16  
17     // Method to deposit an amount into the account  
18     public void deposit(double amount) {  
19         // Increase the balance by the deposit amount  
20         balance+=amount;  
21     }  
22  
23     // Method to withdraw an amount from the account  
24     public void withdraw(double amount) {  
25         // Check if the balance is sufficient for the withdrawal  
26         if (balance >= amount) {  
27             // Decrease the balance by the withdrawal amount  
28             balance -= amount;  
29         } else {  
30             // Print a message if the balance is insufficient  
31             System.out.println("Insufficient balance");  
32         }  
33     }  
34  
35     // Method to get the current balance  
36     public double getBalance() {  
37         // Return the current balance  
38         return balance;  
39     }  
40     public String getAccountNumber(){  
41         return accountNumber;  
42     }  
43 }  
44 class SavingsAccount extends BankAccount {  
45     // Constructor to initialize account number and balance  
46     public SavingsAccount(String accountNumber, double balance) {  
47         // Call the parent class constructor  
48         super(accountNumber,balance);  
49     }  
50 }
```

```
-- 51 // Override the withdraw method from the parent class
52 @Override
```

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

Passed all tests! ✓

◀ Lab-05-MCQ

Jump to...

Is Palindrome Number? ►

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

Status Finished

Started Sunday, 6 October 2024, 11:42 PM

Completed Sunday, 6 October 2024, 11:44 PM

Duration 2 mins 54 secs

Question 1

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
4 public class HelloWorld {
5     public static void main(String[] args) {
6         Scanner scan = new Scanner(System.in);
7         String a = scan.nextLine();
8         String b = scan.nextLine();
9         StringBuffer ab = new StringBuffer();
10        if(a.trim().isEmpty() && b.trim().isEmpty()){
11            System.out.print("null");
12        }
13        else{
14            for(int i = 0;i < a.length();i++){
15                if (a.charAt(i) != ' '){
16                    ab.append(Character.toString(a.charAt(i)));
17                }
18            }
19            for(int i = 0;i < b.length();i++){
20                if (b.charAt(i) != ' '){
21                    ab.append(Character.toString(b.charAt(i)));
22                }
23            }
24            char[] d = ab.toString().toCharArray();
25            Arrays.sort(d);
26            for(int i = d.length - 1;i >= 1;i--){
27                if(d[i] != d[i-1])

```

```
28     System.out.print(d[i]);
29 }
30 System.out.print(d[0]);
31 }
32 }
33 }
34 }
35 }
```

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

Passed all tests! ✓

Question 2

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 v import java.util.*;
2 v class diff{
3 v     char different(char a, char b){
4 v         if ((int)a != (int)b)
5 v             return (char)((int)'a' + ((int)a-(int)b) - 1);
6 v         return a;
7 v     }
8 }
9 v public class Main{
10 v     public static void main(String[] args){
11 v         Scanner scan = new Scanner(System.in);
12 v         diff z = new diff();
13 v         String q = scan.nextLine();
14 v         StringBuffer ans = new StringBuffer();
15 v         StringBuffer temp = new StringBuffer();
16 v         for(int i = 0;i < q.length();i++){
17 v             if(q.charAt(i) == ':'){
18 v                 temp.append(" ");
19 v             }
20 v             else{
21 v                 temp.append(Character.toString(q.charAt(i)));
22 v             }
23 v         }
24 v         String h = temp.toString();
25 v         for(int i = 0;i < temp.length();i++){
26 v             if(i%3 == 0){
27 v                 ans.append(Character.toString(z.different(h.cha
28 v                     )
29 v             }
30 v             System.out.print(ans.toString().toUpperCase());
31 v         }
32 }
33 }
```

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

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.*;
2 public class mix{
3     public static void main(String[] args){
4         Scanner scan = new Scanner(System.in);
5         String g = scan.nextLine();
6         int n = scan.nextInt(), ones, flag = 0;
7         StringBuffer temp = new StringBuffer();
8         StringBuffer temp1 = new StringBuffer();
9         int space = 0;
10        while (n > 0){

```

```

11     ones = (n %10) - 1;
12     for(int i = 0; i < g.length();i++){
13         if (g.charAt(i) == ' '){
14             space = space + 1;
15         }
16         else if(space == ones && flag == 0){
17             temp.append(Character.toString(g.cha
18         }
19         else if(space == ones && flag == 1){
20             temp1.append(Character.toString(g.ch
21         }
22     }
23     space = 0 ;
24     flag = 1;
25     n = n /10;
26 }
27 rew m = new rew();
28 System.out.println(m.r(temp1.toString()) + " " + m.r
29 }
30 }
31 class rew{
32 String r(String a){
33     int le = a.length(),n,q;
34     StringBuffer temp3 = new StringBuffer();
35     if(le % 2 == 1){
36         n = ((int)(le/2));
37         q = ((int)(le/2));
38     }
39     else{
40         n = ((int)(le/2)) - 1;
41         q = ((int)(le/2));
42     }
43     for(int i = n;i >= 0;i--){
44         temp3.append(Character.toString(a.charAt(i))
45     }
46     for(int i = q;i < le;i++){
47         temp3.append(Character.toString(a.charAt(i)));
48     }
49     return temp3.toString();
50 }
51 }
52

```

	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-OOPUJ-2023](#) / [Lab-07-Interfaces](#) / [Lab-07-Logic Building](#)

Status Finished

Started Sunday, 6 October 2024, 11:45 PM

Completed Sunday, 6 October 2024, 11:49 PM

Duration 3 mins 15 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:

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

For example:

Test	Input	Result
1	Sadvin Sanjay Sruthi	Sadvin is Playing football 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 // Define the Playable interface
4 interface Playable {
5     // Abstract method to play the respective sport
6     void play();
7 }
8
9 // Football class implementing Playable interface
10 class Football implements Playable {
11     String name;
12
13     // Constructor
14     public Football(String name) {
15         this.name = name;
16     }
17
18     // Override the play method
19     public void play() {
20         System.out.println(name + " is Playing football");
21     }
22 }
23
24 // Volleyball class implementing Playable interface
25 class Volleyball implements Playable {
26     String name;
27
28     // Constructor
29     public Volleyball(String name) {
30         this.name = name;
31     }
32 }
```

```

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

```

	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

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 // Define the RBI interface
2 interface RBI {
3     // Variable declaration
4     String parentBank = "RBI";
5
6     // Abstract method
7     double rateOfInterest();
8
9     // Default method
10    default void policyNote() {
11        System.out.println("RBI has a new Policy issued in 2023");
12    }
13
14     // Static method
15    static void regulations() {
16        System.out.println("RBI has updated new regulations on 2024");
17    }
18 }
19
20 // SBI class implementing RBI interface
21 class SBI implements RBI {
22     // Implementing the abstract method
23     public double rateOfInterest() {
24         return 7.6;
25     }
26 }
27
28 // Karur class implementing RBI interface
29 class Karur implements RBI {
30     // Implementing the abstract method
31     public double rateOfInterest() {
32         return 7.4;
33     }
34 }
35
36 // Main class to test the functionality
```

```
37 public class Main {  
38     public static void main(String[] args) {  
39         // RBI policies and regulations  
40         RBI rbi = new SBI(); // Can be any class implementing  
41         rbi.policyNote(); // Default method  
42         RBI.regulations(); // Static method  
43  
44         // SBI bank details  
45         SBI sbi = new SBI();  
46         System.out.println("SBI rate of interest: " + sbi.ra  
47  
48         // Karur bank details  
49         Karur karur = new Karur();  
50         System.out.println("Karur rate of interest: " + karu  
51     }  
52 }
```

	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! ✓

Question 3

Correct

Marked out of 5.00

Create interfaces shown below.

```
interface Sports {
    public void setHomeTeam(String name);
    public void setVisitingTeam(String name);
}
```

```
interface Football extends Sports {
    public void homeTeamScored(int points);
    public void visitingTeamScored(int points);}
```

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

sample Input:

Rajalakshmi
Saveetha
22
21

Output:

Rajalakshmi 22 scored
Saveetha 21 scored
Rajalakshmi is the Winner!

For example:

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

Answer: (penalty regime: 0 %)

Reset answer

```
1 import java.util.Scanner;
2
3 interface Sports {
4     void setHomeTeam(String name);
5     void setVisitingTeam(String name);
6 }
7
8 interface Football extends Sports {
9     void homeTeamScored(int points);
10    void visitingTeamScored(int points);
11 }
12
13 class College implements Football {
14     private String homeTeam;
15     private String visitingTeam;
16     private int homeTeamPoints = 0;
17     private int visitingTeamPoints = 0;
18
19     public void setHomeTeam(String name) {
20         this.homeTeam = name;
21     }
22
23     public void setVisitingTeam(String name) {
24         this.visitingTeam = name;
25     }
26
27     public void homeTeamScored(int points) {
28         homeTeamPoints += points;
29         System.out.println(homeTeam + " " + points + " score");
30     }
31
32     public void visitingTeamScored(int points) {
33         visitingTeamPoints += points;
34         System.out.println(visitingTeam + " " + points + " score");
35     }
36 }
```

```

35     }
36
37     public void winningTeam() {
38         if (homeTeamPoints > visitingTeamPoints) {
39             System.out.println(homeTeam + " is the winner!")
40         } else if (homeTeamPoints < visitingTeamPoints) {
41             System.out.println(visitingTeam + " is the winner!")
42         } else {
43             System.out.println("It's a tie match.");
44         }
45     }
46 }
47
48 public class Main {
49     public static void main(String[] args) {
50         Scanner sc = new Scanner(System.in);
51
52

```

	Test	Input	Expected	Got	
✓	1	Rajalakshmi Saveetha 22 21	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!	Rajalakshmi 22 scored Saveetha 21 scored Rajalakshmi is the winner!	✓
✓	2	Anna Balaji 21 21	Anna 21 scored Balaji 21 scored It's a tie match.	Anna 21 scored Balaji 21 scored It's a tie match.	✓
✓	3	SRM VIT 20 21	SRM 20 scored VIT 21 scored VIT is the winner!	SRM 20 scored VIT 21 scored VIT is the winner!	✓

Passed all tests! ✓

◀ Lab-07-MCQ

Jump to...

Generate series and find Nth element ►

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

Status Finished

Started Monday, 21 October 2024, 8:06 PM

Completed Monday, 21 October 2024, 8:50 PM

Duration 44 mins 37 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     int maxSpeed = 120;
4
5
6
7     public void displayMaxSpeed() {
8         System.out.println("The maximum speed is: " + maxSpeed
9     }
10 }
11
12 class SubClass extends FinalExample {
13
14     public void displayMaxSpeed() {
15         System.out.println("Cannot override a final method");
16     }
17
18
19     public void showDetails() {
20         System.out.println("This is a subclass of FinalExample.
21     }
22 }
23
24 class prog {
25     public static void main(String[] args) {
26         FinalExample obj = new FinalExample();
27         obj.displayMaxSpeed();
28
29         SubClass subObj = new SubClass();
30         subObj.showDetails();
31 }
```

```
31 }  
32 }  
33 }
```

	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

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 VowelStringExtractor {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6
7         int n = scanner.nextInt();
8         scanner.nextLine();
9
10        String[] inputStrings = scanner.nextLine().split(" ");
11
12
13        StringBuilder concatenatedString = new StringBuilder();
14        for (String str : inputStrings) {
15            if (isVowel(str.charAt(0)) && isVowel(str.charAt(str.length() - 1))) {
16                concatenatedString.append(str);
17            }
18        }
19    }
20 }
```

```
19     }
20
21
22     String result = concatenatedString.length() > 0 ? concat
23     System.out.println(result);
24
25     scanner.close();
26 }
27
28
29 private static boolean isVowel(char c) {
30     c = Character.toLowerCase(c);
31     return c == 'a' || c == 'e' || c == 'i' || c == 'o' ||
32 }
33
34
35
36
37
```

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

Passed all tests! ✓

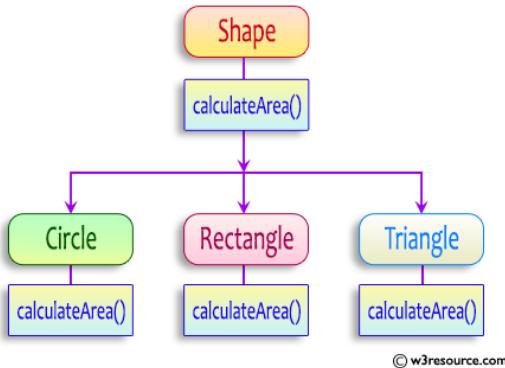
Question 3

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
8 class Circle extends Shape {
9     private double radius;
10
11 public Circle(double radius) {
12     this.radius = radius;
13 }
```

```

12     this.radius = radius;
13 }
14
15 @Override
16 public double calculateArea() {
17     return Math.PI * radius * radius; // Area = π * r^2
18 }
19 }
20
21
22 class Rectangle extends Shape {
23     private double length;
24     private double breadth;
25
26     public Rectangle(double length, double breadth) {
27         this.length = length;
28         this.breadth = breadth;
29     }
30
31     @Override
32     public double calculateArea() {
33         return length * breadth; // Area = length * breadth
34     }
35 }
36
37 class Triangle extends Shape {
38     private double base;
39     private double height;
40
41     public Triangle(double base, double height) {
42         this.base = base;
43         this.height = height;
44     }
45
46     @Override
47     public double calculateArea() {
48         return 0.5 * base * height; // Area = 0.5 * base *
49     }
50 }
51
52

```

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

Passed all tests! ✓

◀ Lab-08-MCQ

Jump to...

FindStringCode ▶

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

Status Finished

Started Monday, 21 October 2024, 8:52 PM

Completed Monday, 21 October 2024, 9:12 PM

Duration 20 mins 22 secs

Question 1

Correct

Marked out of 5.00

Write a Java program to handle `ArithmaticException` and `ArrayIndexOutOfBoundsException`.

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

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

If the 1st element is zero, it will throw an exception.

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

Input:

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

Output:

`java.lang.ArithmaticException: / by zero`

I am always executed

Input:

```
3
10 20 30
```

Output

`java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3`

I am always executed

For example:

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

Answer: (penalty regime: 0 %)

```

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

```
38     ,
39     scanner.close();
40 }
41 }
42 }
43 }
44 }
```

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

Passed all tests! ✓

Question 2

Correct

Marked out of 5.00

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

Sample input and Output:

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

Fill the preloaded answer to get the expected output.

For example:**Result**

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

Answer: (penalty regime: 0 %)

[Reset answer](#)

```
1 public class EvenOddChecker {  
2  
3     // Custom exception class  
4     static class OddNumberException extends Exception {  
5         public OddNumberException(String message) {  
6             super(message);  
7         }  
8     }  
9  
10    public static void main(String[] args) {  
11        try {  
12            checkEvenOrOdd(82); // Test with an even number  
13            checkEvenOrOdd(37); // Test with an odd number  
14        } catch (OddNumberException e) {  
15            System.out.println(e.getMessage()); // Print the error message  
16        }  
17    }  
18  
19    // Method to check if a number is even or odd  
20    public static void checkEvenOrOdd(int number) throws OddNumberException {  
21        if (number % 2 == 0) {  
22            System.out.println(number + " is even.");  
23        } else {  
24            throw new OddNumberException("Error: " + number + " is odd.");  
25        }  
26    }  
27}  
28  
29
```

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

Passed all tests! ✓

Question 3

Correct

Marked out of 5.00

In the following program, an array of integer data is to be initialized.

During the initialization, if a user enters a value other than an integer, it will throw an InputMismatchException exception.

On the occurrence of such an exception, your program should print "You entered bad data."

If there is no such exception it will print the total sum of the array.

```
/* Define try-catch block to save user input in the array "name"
```

```
If there is an exception then catch the exception otherwise print the total sum of the array. */
```

Sample Input:

```
3  
5 2 1
```

Sample Output:

```
8
```

Sample Input:

```
2  
1 g
```

Sample Output:

```
You entered bad data.
```

For example:

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

Answer: (penalty regime: 0 %)

```
1 import java.util.Scanner;  
2 import java.util.InputMismatchException;  
3  
4 public class ArraySumCalculator {  
5     public static void main(String[] args) {  
6         Scanner scanner = new Scanner(System.in);  
7  
8         int size = scanner.nextInt();  
9         int[] numbers = new int[size];  
10        int sum = 0;  
11  
12        try {  
13  
16            for (int i = 0; i < size; i++) {  
17                numbers[i] = scanner.nextInt();  
18                sum += numbers[i];  
19            }  
20  
21            System.out.println(sum);  
22        } catch (InputMismatchException e) {  
23            // Handle the InputMismatchException  
24            System.out.println("You entered bad data.");  
25        } finally {  
26            scanner.close(); // Close the scanner  
27        }  
28    }  
29}
```

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

Passed all tests! ✓

[◀ Lab-09-MCQ](#)

Jump to...

[The “Nambiar Number” Generator ►](#)

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

Status Finished

Started Tuesday, 5 November 2024, 8:22 AM

Completed Tuesday, 5 November 2024, 8:44 AM

Duration 21 mins 32 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.Scanner;
2 import java.util.Collections;
3 import java.util.ArrayList;
4 public class main{
5     public static void main(String args[])
6     {
7         ArrayList<Integer>l=new ArrayList<>();
8         Scanner s=new Scanner(System.in);
9         int n=s.nextInt();
10        for(int i=0;i<n;i++)
11        {
12            l.add(s.nextInt());
13        }
14        System.out.println("ArrayList: "+l+"\nFirst : "+l.get(0));
15    }
16}
17 }
```

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

	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]	✓

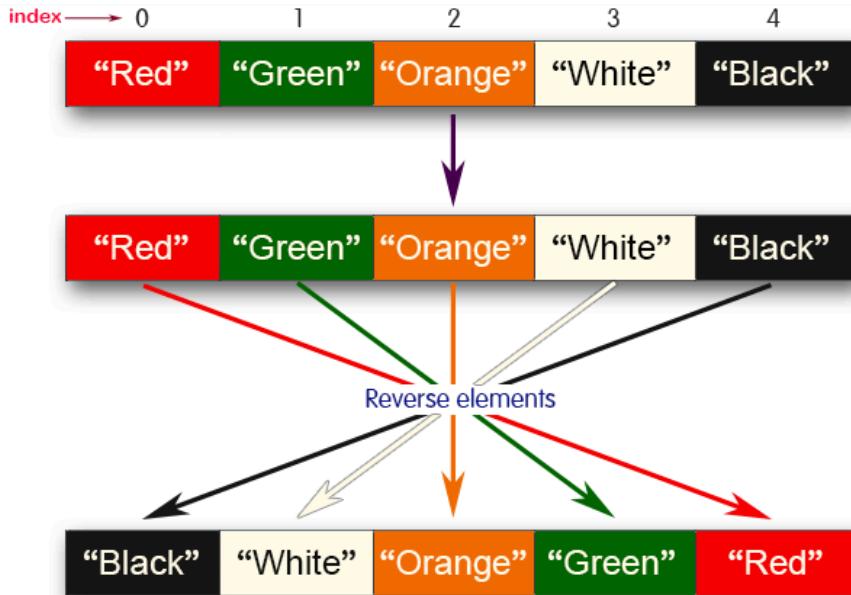
Passed all tests! ✓

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.Scanner;
2 import java.util.ArrayList;
3 public class main{
4     public static void main(String args[])
5     {
6         Scanner s=new Scanner(System.in);
7         int n=s.nextInt();
8         ArrayList<String>l=new ArrayList<>();
9         for(int i=0;i<n;i++)
10            l.add(s.next());
11         ArrayList<String>l2=new ArrayList<>();
12         System.out.println("List before reversing :\n"+l);
13         for(int i=l.size()-1;i>=0;i--)
14         {
15             l2.add(l.get(i));
16         }
17         System.out.println("List after reversing :\n"+l2);
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-OOPUJ-2023](#) / [Lab-11-Set, Map](#) / [Lab-11-Logic Building](#)

Status	Finished
Started	Saturday, 16 November 2024, 8:55 PM
Completed	Saturday, 16 November 2024, 9:18 PM
Duration	22 mins 28 secs

Question 1

Correct

Marked out of 1.00

Java HashSet class implements the Set interface, backed by a hash table which is actually a [HashMap](#) instance.

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

This class permits the null element.

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

Java HashSet Features

A few important features of HashSet are mentioned below:

- Implements [Set Interface](#).
- The underlying data structure for HashSet is [Hashtable](#).
- As it implements the Set Interface, duplicate values are not allowed.
- Objects that you insert in HashSet are not guaranteed to be inserted in the same order. Objects are inserted based on their hash code.
- NULL elements are allowed in HashSet.
- HashSet also implements **Serializable** and **Cloneable** interfaces.

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

Sample Input and Output:

5

90

56

45

78

25

78

Sample Output:

78 was found in the set.

Sample Input and output:

3

2

7

9

5

Sample Input and output:

5 was not found in the set.

Answer: (penalty regime: 0 %)

Reset answer

```

1 import java.util.HashSet;
2 import java.util.Scanner;
3
4 public class HashSetExample {
5     public static void main(String[] args) {
6         HashSet<Integer> set = new HashSet<>();
7         Scanner scanner = new Scanner(System.in);
8         int n = scanner.nextInt();
9         for (int i = 0; i < n; i++) {
10             int element = scanner.nextInt();
11             set.add(element);
12         }
13
14         int searchElement = scanner.nextInt();
15
16         if (set.contains(searchElement)) {
17             System.out.println(searchElement + " was found in the set.");
18         } else {
19             System.out.println(searchElement + " was not found in the set.");
20         }
21         scanner.close();
22     }
23 }

```

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

Passed all tests! ✓

Question 2

Correct

Marked out of 1.00

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

Sample Input and Output:

```
5
Football
Hockey
Cricket
Volleyball
Basketball
7 // HashSet 2:
```

```
Golf
Cricket
Badminton
Football
Hockey
Volleyball
Handball
```

SAMPLE OUTPUT:

```
Football
Hockey
Cricket
Volleyball
Basketball
```

Answer: (penalty regime: 0 %)

```
1 import java.util.HashSet;
2 import java.util.Scanner;
3 import java.util.Set;
4
5 public class CompareSets {
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8
9         Set<String> set1 = new HashSet<>();
10        Set<String> set2 = new HashSet<>();
11
12        int n1 = scanner.nextInt();
13        scanner.nextLine();
14        for (int i = 0; i < n1; i++) {
15            set1.add(scanner.nextLine());
16        }
17        int n2 = scanner.nextInt();
18        scanner.nextLine();
19        for (int i = 0; i < n2; i++) {
20            set2.add(scanner.nextLine());
21        }
22
23
24        set1.retainAll(set2);
25
26        for (String element : set1) {
27            System.out.println(element);
28        }
29
30
31        scanner.close();
32    }
33}
34
35}
```

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

Passed all tests! ✓

//

Question 3

Correct

Marked out of 1.00

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

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

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

```

14 // Getting the number of key-value pairs
15 int n = sc.nextInt();
16
17 // Inserting key-value pairs into the map
18 for (int i = 0; i < n; i++) {
19     name = sc.next();
20     num = sc.nextInt();
21     map.put(name, num);
22 }
23
24 // Printing key-value pairs of map
25 Set<Entry<String, Integer>> entrySet = map.entrySet();
26 for (Entry<String, Integer> entry : entrySet) {
27     System.out.println(entry.getKey() + " : " + entry.getValue());
28 }
29 System.out.println("-----");
30
31 // Creating another HashMap
32 HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();
33
34 // Inserting key-value pairs to anotherMap using put() method
35 anotherMap.put("SIX", 6);
36 anotherMap.put("SEVEN", 7);
37
38 // Inserting key-value pairs of map to anotherMap using putAll() method
39 anotherMap.putAll(map); // Insert all entries from map into anotherMap
40
41 // Printing key-value pairs of anotherMap
42 entrySet = anotherMap.entrySet();
43 for (Entry<String, Integer> entry : entrySet) {
44     System.out.println(entry.getKey() + " : " + entry.getValue());
45 }
46
47 // Adds key-value pair 'FIVE-5' only if it is not present in map
48 map.putIfAbsent("FIVE", 5);
49
50 // Retrieving a value associated with key 'TWO'
51 int value = map.get("TWO");
52 System.out.println(value);
53
54 // Checking whether key 'ONE' exists in map
55 System.out.println(map.containsKey("ONE"));
56
57 // Checking whether value '3' exists in map
58 System.out.println(map.containsValue(3));
59
60 // Retrieving the number of key-value pairs present in map
61 System.out.println(map.size());
62
63 }
64 }
```

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

Passed all tests! ✓

◀ Lab-11-MCQ

Jump to...

TreeSet example ►

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

Status Finished

Started Tuesday, 19 November 2024, 1:13 AM

Completed Tuesday, 19 November 2024, 1:27 AM

Duration 14 mins 5 secs

Question 1

Correct

Marked out of 5.00

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

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

Each alphabet is represented by a sequence of 0s.

This is as mentioned below:

Z: 0

Y: 00

x · 000

w : 0000

V:00000

U · 000000

T: 0000000

and so on upto A having 26 0's (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:

The decoded string (original word) will be: WIPBO

Note: The decoded string must always be in UPPER case

For example:

Input	Result
010010001	ZYX
0000100000000000000000000000000100000000000010000000000001000000000000001	WIPRO

Answer: (penalty regime: 0 %)

```
1 import java.util.Scanner;
2
3 public class DecodeSequence {
4
5     public static String decodeString(String encodedString) {
6         // Split the string by '1' to separate groups of zeros
7         String[] groups = encodedString.split("1");
8         StringBuilder decodedWord = new StringBuilder();
9
10        // Decode each group of zeros
11        for (String group : groups) {
12            if (!group.isEmpty()) { // Skip empty groups
13                int length = group.length();
14                // Calculate the corresponding letter (Z=1 zero)
15                char letter = (char) ('Z' - length + 1);
16                decodedWord.append(letter);
17            }
18        }
19
20        return decodedWord.toString();
21    }
22
23    public static void main(String[] args) {
24        Scanner scanner = new Scanner(System.in);
25        String input = scanner.nextLine();
26    }
}
```

```
27     // Decode and print the output
28     String decodedWord = decodeString(input);
29     System.out.println(decodedWord);
30
31     scanner.close();
32 }
33 }
34 }
```

	Input	Expected	Got	
✓	010010001	ZYX	ZYX	✓
✓	0000100000000000000000001000000000001000000000010000000000001	WIPRO	WIPRO	✓

Passed all tests! ✓

Question 2

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 Seigolonhcet Erolagnab".

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

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

NOTE:

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

2. Non-alphabetic characters in the String should not be subjected to case changes. For example, if case option = 1 and the original sentence is "Wipro TechNologies, Bangalore" the new reversed sentence should be "Orpiw ,seiGolonhceT Erolagnab". Note that comma has been treated as part of the word "Technologies," and when comma had to take the position of uppercase T it remained as a comma and uppercase T took the position of comma. However, the words "Wipro and Bangalore" have changed to "Orpiw" and "Erolagnab".

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

Examples:

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

For example:

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

Answer: (penalty regime: 0 %)

```

1 import java.util.*;
2
3 public class prog {
4
5     public static String reverseWords(String sentence, int case
6         String[] words = sentence.split(" "); // Split sentence
7         StringBuilder result = new StringBuilder();
8
9         for (String word : words) {
10             if (caseOption == 0) {
11                 // Normal reversal without retaining case
12                 result.append(new StringBuilder(word).reverse())
13             } else if (caseOption == 1) {
14                 // Reverse while retaining the case
15                 result.append(reverseWithCaseRetention(word)).a
16             }
}

```

```

17     }
18
19     return result.toString().trim(); // Remove trailing space
20 }
21
22 private static String reverseWithCaseRetention(String word)
23     char[] original = word.toCharArray();
24     char[] reversed = new char[original.length];
25
26     // Reverse the word
27     for (int i = 0, j = original.length - 1; i < original.l
28         reversed[i] = original[j];
29     }
30
31     // Retain the case of the original word
32     for (int i = 0; i < original.length; i++) {
33         if (Character.isUpperCase(original[i])) {
34             reversed[i] = Character.toUpperCase(reversed[i])
35         } else if (Character.isLowerCase(original[i])) {
36             reversed[i] = Character.toLowerCase(reversed[i])
37         }
38     }
39
40     return new String(reversed);
41 }
42
43 public static void main(String[] args) {
44     Scanner scan =new Scanner(System.in);
45     System.out.println(reverseWords(scan.nextLine(),scan.ne
46 }
47 }
```

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

Passed all tests! ✓

Question 3

Correct

Marked out of 5.00

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

Get the ASCII values of all the extracted alphabets.

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

Return that single digit as output.

Note:

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

Example 1:

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

input2: {'b', 'c'}

output: 8

Explanation:

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

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

$$98 + 99 = 197$$

$$1 + 9 + 7 = 17$$

$$1 + 7 = 8$$

For example:

Input	Result
a b c	8
b c	

Answer: (penalty regime: 0 %)

```

1 import java.util.HashSet;
2
3 public class CommonAlphabets {
4
5     public static void main(String[] args) {
6         char[] input1 = {'a', 'b', 'c'};
7         char[] input2 = {'b', 'c'};
8
9         // Step 1: Find common alphabets
10        HashSet<Character> commonChars = new HashSet<>();
11        for (char ch : input1) {
12            for (char ch2 : input2) {
13                if (ch == ch2) {
14                    commonChars.add(ch);
15                }
16            }
17        }
18
19        // Step 2: Calculate the sum of ASCII values of common
20        int sum1 = 0;
21        for (char ch : commonChars) {
22            sum1 += (int) ch;
23        }
24
25        // Step 3: Calculate the single digit sum of sum1
26        int singleDigitSum = getSingleDigitSum(sum1);
27
28        // Step 4: Output the result
29        System.out.println(singleDigitSum);

```

```
30     }
31
32     // Method to calculate the single digit sum
33     public static int getSingleDigitSum(int number) {
34         while (number >= 10) {
35             int tempSum = 0;
36             while (number > 0) {
37                 tempSum += number % 10;
38                 number /= 10;
39             }
40             number = tempSum;
41         }
42         return number;
43     }
44 }
```

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

Passed all tests! ✓

◀ Lab-12-MCQ

Jump to...

Identify possible words ►

LOGISTICS MANAGEMENT

A MINI-PROJECT BY:

Krithika B 230701156

Kommana sai leela Yuktha 230701154

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2023

BONAFIDE CERTIFICATE

Certified that this project "**LOGISTICS MANAGEMENT**" is the bona fide work of "**KRITHIKA .B, KOMMANA SAI LEELA YUKTHA**" who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Ms.DHARSHINI BS
Assistant Professor(SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

SIGNATURE

Ms.V.JANANEE
Assistant Professor(SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Logistics Management Mini Project

Efficient logistics management is essential for businesses to maintain smooth operations and meet customer demands. This mini project presents a comprehensive solution to optimize inventory control, supplier coordination, and order processing. The system leverages real-time stock tracking with automatic low-stock alerts to ensure businesses can prevent stockouts or overstocking, thereby maintaining a balanced inventory.

The project integrates a centralized platform for supplier information, simplifying communication and restocking processes. The order management feature streamlines order placement and fulfillment, linking each order to specific customers and providing accurate tracking of payments and delivery statuses. These functionalities reduce errors and improve overall process efficiency.

Comprehensive inventory logs record all stock changes, including sales, restocking, and adjustments, ensuring transparency and accountability. Additionally, the system provides detailed analytics and reporting, offering insights into sales trends, inventory levels, and critical alerts. This empowers businesses to make informed, data-driven decisions.

By addressing common logistical challenges, this mini project aims to enhance operational efficiency, improve customer satisfaction, and support businesses in staying competitive in dynamic markets.

TABLE OF CONTENTS

1. INTRODUCTION

- 1.1 INTRODUCTION
- 1.2 IMPLEMENTATION
- 1.3 SCOPE OF THE PROJECT
- 1.4 WEBSITE FEATURES

2. SYSTEM SPECIFICATION

- 2.1 HARDWARE SPECIFICATION
- 2.2 SOFTWARE SPECIFICATION

3. SAMPLE CODE

- 3.1 LOGIN PAGE
- 3.2 DASHBOARD
- 3.3 APP
- 3.4 CUSTOMER DATA
- 3.5 PRIMARY
- 3.6 PRODUCTS MODEL
- 3.7 MODULE INFO

4. SNAPSHOTS

- 4.1 LOGIN PAGE
- 4.2 LOGIN SUCCESSFULLY
- 4.3 HOME PAGE
- 4.4 ENTERING ORDERS
- 4.5 ADDING PRODUCTS
- 4.6 LOGOUT SUCCESSFUL

5. CONCLUSION

6. REFERENCES

INTRODUCTION

1.1 INTRODUCTION

The project streamlines logistics management by offering businesses a centralized platform for inventory tracking, supplier coordination, and order processing. Users can monitor stock levels in real-time with alerts, ensuring optimal inventory control. Supplier details are centralized for seamless restocking, while orders are accurately tracked with payment and delivery status updates. Comprehensive logs and detailed analytics empower businesses to make informed decisions, improve efficiency, and enhance customer satisfaction.

1.1 IMPLEMENTATION

The **LOGISTICS MANAGEMENT** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

1.2 SCOPE OF THE PROJECT

The logistics management system is designed for businesses to register and create accounts, allowing them to manage all their inventory and order data in one organized platform. This ensures streamlined operations, saves time, and promotes a sense of professionalism.

1.3 WEBSITE FEATURES

- Registering and login page.
- Income data chart .
- Dashboard showing possible actions.
- Overview of all the products .

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR : Intel i5

MEMORY SIZE : 4GB(Minimum)

HARD DISK : 500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE : Java, MySQL

FRONT-END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 11

SAMPLE CODE

3.1 LOGIN PAGE

```
4 package com.mycompany.yt_inventorymanagementsystem_javafx;
5
6 import java.net.URL;
7 import java.sql.Connection;
8 import java.sql.PreparedStatement;
9 import java.sql.ResultSet;
10 import java.util.ResourceBundle;
11 import javafx.fxml.FXML;
12 import javafx.fxml.FXMLLoader;
13 import javafx.fxml.Initializable;
14 import javafx.scene.Parent;
15 import javafx.scene.Scene;
16 import javafx.scene.control.Alert;
17 import javafx.scene.control.Button;
18 import javafx.scene.control.PasswordField;
19 import javafx.scene.control.TextField;
20 import javafx.scene.input.MouseEvent;
21 import javafx.scene.layout.AnchorPane;
22 import javafx.stage.Stage;
23 import javafx.stage.StageStyle;
24
25 /**
26 * FXML Controller class
27 *
28 * @author USER
29 */
30 public class LoginController implements Initializable {
31
32     @FXML
33     private Button close_btn;
34
35     @FXML
36     private Button login_btn;
37
38     @FXML
39     private AnchorPane main_form;
40
41     @FXML
42     private PasswordField password_txt;
43
44     @FXML
45     private TextField username_txt;
46
47     // DATABASE TOOLS
48     private Connection con;
49     private PreparedStatement prepare;
50     private ResultSet result;
51     private double x = 0;
52     private double y = 0;
53
54     /**
55      * Initializes the controller class.
56      */
57     @Override
58     public void initialize(URL url, ResourceBundle rb) {
59         // Optional: Initialize any necessary components here
60     }
```

```

61
62     // LOGIN METHOD
63     public void loginAdmin() {
64         String sql = "select * from users where username = ? and password = ?";
65
66         // Try to establish a connection
67         con = databaseHandler.connectDb();
68
69         if (con == null) {
70             showAlert(Alert.AlertType.ERROR, "Database Connection Error", "Failed to connect to the
database.");
71             return;
72         }
73
74         try {
75             prepare = con.prepareStatement(sql);
76             prepare.setString(1, username_txt.getText());
77             prepare.setString(2, password_txt.getText());
78
79             result = prepare.executeQuery();
80
81             if (username_txt.getText().isEmpty() || password_txt.getText().isEmpty()) {
82                 showAlert(Alert.AlertType.ERROR, "Input Error", "Please fill all blank fields.");
83             } else {
84                 if (result.next()) {
85                     // If login is successful, store username for later use
86                     GetData.username = username_txt.getText();
87
88                     // Show success alert
89                     showAlert(Alert.AlertType.INFORMATION, "Login Success", "Successfully logged
in!");
90
91                     // Hide the login window
92                     login_btn.getScene().getWindow().hide();
93
94                     // Load the dashboard scene
95                     loadDashboard();
96                 } else {
97                     // If username/password is wrong
98                     showAlert(Alert.AlertType.ERROR, "Login Failed", "Wrong Username or Password.");
99                 }
100            }
101        } catch (Exception e) {
102            e.printStackTrace();
103            showAlert(Alert.AlertType.ERROR, "Error", "An error occurred: " + e.getMessage());
104        }
105    }
106
107    // Helper method to show alerts
108    private void showAlert(Alert.AlertType type, String title, String content) {
109        Alert alert = new Alert(type);
110        alert.setTitle(title);
111        alert.setHeaderText(null);
112        alert.setContentText(content);
113        alert.showAndWait();
114    }
115
116    // Load the dashboard
117    private void loadDashboard() {
118        try {
119            Parent root = FXMLLoader.load(getClass().getResource("/fxml/dashboard.fxml"));
120            Stage stage = new Stage();
121            Scene scene = new Scene(root);

```

```

122         // Drag functionality for moving the window
123         root.setOnMousePressed((MouseEvent event) -> {
124             x = event.getSceneX();
125             y = event.getSceneY();
126         });
127
128         root.setOnMouseDragged((MouseEvent event) -> {
129             stage.setX(event.getScreenX() - x);
130             stage.setY(event.getScreenY() - y);
131         });
132
133         stage.initStyle(StageStyle.TRANSPARENT);
134         stage.setScene(scene);
135         stage.show();
136     } catch (Exception e) {
137         e.printStackTrace();
138         showAlert(Alert.AlertType.ERROR, "Error", "Unable to load the dashboard: " +
139             e.getMessage());
140     }
141 }
142
143     // Close the application
144     public void close() {
145         System.exit(0);
146     }
147 }
```

3.2 Dashboard

```

4 package com.mycompany.yt_inventorymanagementsystem_javafx;
5
6 import de.jensd.fx.glyphs.fontawesome.FontAwesomeIconView;
7 import java.io.File;
8 import java.net.URL;
9 import java.sql.Connection;
10 import java.sql.PreparedStatement;
11 import java.sql.ResultSet;
12 import java.sql.Statement;
13 import java.util.ArrayList;
14 import java.util.Date;
15 import java.util.HashMap;
16 import java.util.List;
17 import java.util.Optional;
18 import java.util.ResourceBundle;
19 import javafx.collections.FXCollections;
20 import javafx.collections.ObservableList;
21 import javafx.collections.transformation.FilteredList;
22 import javafx.collections.transformation.SortedList;
23 import javafx.event.ActionEvent;
24 import javafx.fxml.FXML;
25 import javafx.fxml.FXMLLoader;
26 import javafx.fxml.Initializable;
27 import javafx.scene.Parent;
28 import javafx.scene.Scene;
29 import javafx.scene.chart.AreaChart;
30 import javafx.scene.chart.BarChart;
31 import javafx.scene.chart.XYChart;
32 import javafx.scene.control.Alert;
33 import javafx.scene.control.Button;
34 import javafx.scene.control.ButtonType;
35 import javafx.scene.control.ComboBox;
36 import javafx.scene.control.Label;
37 import javafx.scene.control.Spinner;
```

```
38 import javafx.scene.control.SpinnerValueFactory;
39 import javafx.scene.control.TableColumn;
40 import javafx.scene.control.TableView;
41 import javafx.scene.control.TextField;
42 import javafx.scene.control.cell.PropertyValueFactory;
43 import javafx.scene.image.Image;
44 import javafx.scene.image.ImageView;
45 import javafx.scene.input.MouseEvent;
46 import javafx.scene.layout.AnchorPane;
47 import javafx.stage.FileChooser;
48 import javafx.stage.Stage;
49 import javafx.stage.StageStyle;
50 import net.sf.jasperreports.engine.JRException;
51 import net.sf.jasperreports.engine.JasperCompileManager;
52 import net.sf.jasperreports.engine.JasperFillManager;
53 import net.sf.jasperreports.engine.JasperPrint;
54 import net.sf.jasperreports.engine.JasperReport;
55 import net.sf.jasperreports.engine.design.JasperDesign;
56 import net.sf.jasperreports.engine.xml.JRXmlLoader;
57 import net.sf.jasperreports.view.JasperViewer;
58
59 /**
60  * FXML Controller class
61  *
62  * @author USER
63  */
64 public class DashboardController implements Initializable {
65
66     /**
67      * Initializes the controller class.
68      */
69     @FXML
70     private Button addProduct_add_btn;
71
72     @FXML
73     private TextField addProduct_brand_txt;
74
75     @FXML
76     private FontAwesomeIconView addProduct_btn;
77
78     @FXML
79     private Button addProduct_delete_btn;
80
81     @FXML
82     private ImageView addProduct_imgView;
83
84     @FXML
85     private Button addProduct_import_btn;
86
87     @FXML
88     private TextField addProduct_price_txt;
89
90     @FXML
91     private TextField addProduct_productId_txt;
92
93     @FXML
94     private TextField addProduct_productName_txt;
95
96     @FXML
97     private ComboBox<?> addProduct_productType_combo;
98
99     @FXML
```

```
100     private Button addProduct_reset_btn;
101
102     @FXML
103     private TextField addProduct_search_txt;
104
105     @FXML
106     private ComboBox<?> addProduct_status_combo;
107
108     @FXML
109     private TableView<ProductsModel> addProduct_tableView;
110
111     @FXML
112     private TableColumn<ProductsModel, String> addProduct_tbvCol_brand;
113
114     @FXML
115     private TableColumn<ProductsModel, String> addProduct_tbvCol_price;
116
117     @FXML
118     private TableColumn<ProductsModel, String> addProduct_tbvCol_productID;
119
120     @FXML
121     private TableColumn<ProductsModel, String> addProduct_tbvCol_productName;
122
123     @FXML
124     private TableColumn<ProductsModel, String> addProduct_tbvCol_status;
125
126     @FXML
127     private TableColumn<ProductsModel, String> addProduct_tbvCol_type;
128
129     @FXML
130     private Button addProduct_update_btn;
131
132     @FXML
133     private Button addProducts_btn;
134
135     @FXML
136     private AnchorPane addProducts_form;
137
138     @FXML
139     private Button close_btn;
140
141     @FXML
142     private Label home_availableProducts_lbl;
143
144     @FXML
145     private Button home_btn;
146
147     @FXML
148     private AnchorPane home_form;
149
150     @FXML
151     private AreaChart<?, ?> home_incomeChart;
152
153     @FXML
154     private Label home_numberOfOrders_lbl;
155
156     @FXML
157     private BarChart<?, ?> home_orderChart;
158
159     @FXML
160     private Label home_totalIncome_lbl;
161
162     @FXML
```

```
163     private Button logout;
164
165     @FXML
166     private AnchorPane main_form;
167
168     @FXML
169     private FontAwesomeIconView minimize_btn;
170
171     @FXML
172     private Button orders_add_btn;
173
174     @FXML
175     private TextField orders_amount_txt;
176
177     @FXML
178     private Label orders_balance_lbl;
179
180     @FXML
181     private ComboBox<?> orders_brandName_combo;
182
183     @FXML
184     private Button orders_btn;
185
186     @FXML
187     private TableColumn<CustomerData, String> orders_col_brand;
188
189     @FXML
190     private TableColumn<CustomerData, String> orders_col_price;
191
192     @FXML
193     private TableColumn<CustomerData, String> orders_col_productName;
194
195     @FXML
196     private TableColumn<CustomerData, String> orders_col_quantity;
197
198     @FXML
199     private TableColumn<CustomerData, String> orders_col_type;
200
201     @FXML
202     private AnchorPane orders_form;
203
204     @FXML
205     private Button orders_pay_btn;
206
207     @FXML
208     private ComboBox<?> orders_productName_combo;
209
210     @FXML
211     private ComboBox<?> orders_productType_combo;
212
213     @FXML
214     private Spinner<Integer> orders_quantity_spinner;
215
216     @FXML
217     private Button orders_receipt_btn;
218
219     @FXML
220     private Button orders_reset_btn;
221
222     @FXML
223     private TableView<CustomerData> orders_tableView;
224
225     @FXML
```

```
226     private Label orders_total_lbl;
227
228     @FXML
229     private Label username_lbl;
230
231     // DATABASE TOOLS
232     private Connection conn;
233     private PreparedStatement prepare;
234     private Statement statement;
235     private ResultSet resultSet;
236
237
238     public void homeDisplayTotalOrders(){
239     //
240     //        Date date = new Date();
241     //        java.sql.Date sqlDate = new java.sql.Date(date.getTime());
242     //        prepare.setString(8, String.valueOf(sqlDate));
243
244     Date date = new Date();
245     java.sql.Date sqlDate = new java.sql.Date(date.getTime());
246
247     //        String sql = "SELECT COUNT(id) AS count_of_id FROM customer_receipt";
248     String sql = "SELECT COUNT(id) FROM customer_receipt WHERE date = '"+sqlDate+"' ";
249
250     conn = databaseHandler.connectDb();
251
252     int countOrders = 0;
253
254     try {
255
256         prepare = conn.prepareStatement(sql);
257         resultSet = prepare.executeQuery();
258
259         if(resultSet.next()){
260     //            countOrders = resultSet.getInt("count_of_id");
261             countOrders = resultSet.getInt("COUNT(id)");
262         }
263
264         home_numberOfOrders_lbl.setText(String.valueOf(countOrders));
265
266     } catch (Exception e) {e.printStackTrace();}
267
268     }
269
270     public void homeTotalIncome(){
271
272         String sql = "SELECT SUM(total) AS total_income FROM customer_receipt";
273
274         conn = databaseHandler.connectDb();
275
276         double totalIncome = 0;
277
278         try {
279
280             prepare = conn.prepareStatement(sql);
281             resultSet = prepare.executeQuery();
282
283             if(resultSet.next()){
284                 totalIncome = resultSet.getDouble("total_income");
285             }
286
287             home_totalIncome_lbl.setText(String.valueOf("Rs " + totalIncome));

```

```
288     } catch (Exception e) {e.printStackTrace();}
289 }
290 }
291 }
292
293 public void homeAvailableProducts(){
294
295     String sql = "SELECT COUNT(id) AS ava_products FROM product WHERE status = 'Available'";
296
297     conn = databaseHandler.connectDb();
298
299     int availableProducts = 0;
300
301     try {
302
303         prepare = conn.prepareStatement(sql);
304         resultSet = prepare.executeQuery();
305
306         if(resultSet.next()){
307             availableProducts = resultSet.getInt("ava_products");
308         }
309
310         home_availableProducts_lbl.setText(String.valueOf(availableProducts));
311
312     } catch (Exception e) {e.printStackTrace();}
313 }
314 }
315
316 public void homeIncomeChart(){
317
318     home_incomeChart.getData().clear();
319
320 //     String sql = "SELECT date, SUM(total) FROM customer_receipt GROUP BY date GROUP BY
321 //     TIMESTAMP(date) ASC LIMIT 6";
322
323     String sql = "SELECT date, SUM(total) FROM customer_receipt GROUP BY date ORDER BY
324     TIMESTAMP(date) ASC LIMIT 6";
325
326     conn = databaseHandler.connectDb();
327
328     try {
329
330         XYChart.Series chart = new XYChart.Series();
331
332         prepare = conn.prepareStatement(sql);
333         resultSet = prepare.executeQuery();
334
335         while(resultSet.next()){
336             chart.getData().add(new XYChart.Data(resultSet.getString(1), resultSet.getInt(2)));
337         }
338
339         home_incomeChart.getData().add(chart);
340
341     } catch (Exception e) {e.printStackTrace();}
342 }
343
344 public void homeOrdersChart(){
345
346     home_orderChart.getData().clear();
347
348     String sql = "SELECT date, COUNT(id) FROM customer_receipt GROUP BY date ORDER BY
349     TIMESTAMP(date) ASC LIMIT 5";
```

```

348
349     /*
350      SELECT date, SUM(total): We ask for two things: the date and the sum of income for each date.
351      The SUM(total) adds up all income values for each date, so we get the total income on that day.
352      FROM customer_receipt: This tells the database to look in the customer_receipt table.
353      GROUP BY date: It groups the data by date, so it only shows one total amount per date.
354      ORDER BY TIMESTAMP(date) ASC: Orders the data by date from oldest to newest.
355      LIMIT 6: Only shows the last 6 days (or dates) of data.
356      */
357
358      conn = databaseHandler.connectDb();
359
360      try {
361
362          XYChart.Series chart = new XYChart.Series<>(); // This creates an empty series, which is
363          like a container that will hold all the data points for the chart.
364
365          prepare = conn.prepareStatement(sql);
366          resultSet = prepare.executeQuery();
367
368          while(resultSet.next()){
369              chart.getData().add(new XYChart.Data(resultSet.getString(1), resultSet.getInt(2)));
370
371              /*
372              resultSet.next(): Moves to the next row of data.
373              resultSet.getString(1): Gets the date from the first column.
374              resultSet.getInt(2): Gets the total income for that date from the second column.
375              */
376
377          home_orderChart.getData().add(chart);
378
379      } catch (Exception e) {e.printStackTrace();}
380
381  }
382
383  public void addProductsAdd() {
384
385      String sql = "INSERT INTO product (product_id, type, brand, product_name, price, status, image,
386      date) "
387          + "VALUES(?,?,?,?,?,?)";
388
389      conn = databaseHandler.connectDb();
390
391      try {
392
393          Alert alert;
394
395          if (addProduct_productId_txt.getText().isEmpty()
396              || addProduct_productType_combo.getSelectionModel().getSelectedItem() == null
397              || addProduct_brand_txt.getText().isEmpty()
398              || addProduct_productName_txt.getText().isEmpty()
399              || addProduct_price_txt.getText().isEmpty()
400              || addProduct_status_combo.getSelectionModel().getSelectedItem() == null
401              || GetData.path == "") {
402
403              alert = new Alert(Alert.AlertType.ERROR);
404              alert.setTitle("Error Message");
405              alert.setHeaderText(null);
406              alert.setContentText("Please fill all blank fields");
407              alert.showAndWait();

```

```

408         } else {
409
410             // CHECK IF THE PRODUCT ID IS ALREADY EXIST
411             String checkData = "SELECT product_id FROM product WHERE product_id = '" +
412                 + addProduct_productId_txt.getText() + "'";
413
414             statement = conn.createStatement();
415             resultSet = statement.executeQuery(checkData);
416
417             if (resultSet.next()) {
418
419                 alert = new Alert(Alert.AlertType.ERROR);
420                 alert.setTitle("Error Message");
421                 alert.setHeaderText(null);
422                 alert.setContentText("Product ID: " + addProduct_productId_txt.getText() + " was
already exist!");
423                 alert.showAndWait();
424             } else {
425
426                 prepare = conn.prepareStatement(sql);
427                 prepare.setString(1, addProduct_productId_txt.getText());
428                 prepare.setString(2, (String)
addProduct_productType_combo.getSelectionModel().getSelectedItem());
429                 prepare.setString(3, addProduct_brand_txt.getText());
430                 prepare.setString(4, addProduct_productName_txt.getText());
431                 prepare.setString(5, addProduct_price_txt.getText());
432                 prepare.setString(6, (String)
addProduct_status_combo.getSelectionModel().getSelectedItem());
433
434                 String uri = GetData.path;
435                 uri = uri.replace("\\", "\\\\");
436                 prepare.setString(7, uri);
437
438                 Date date = new Date();
439                 java.sql.Date sqlDate = new java.sql.Date(date.getTime());
440                 prepare.setString(8, String.valueOf(sqlDate));
441
442                 prepare.executeUpdate();
443
444             /*
445             Difference Between executeUpdate() and executeQuery():
446             executeUpdate():
447
448                 This method is used for modifying the database. It is typically used with SQL
statements such as INSERT, UPDATE, DELETE, and CREATE. These types of statements alter the contents or
structure of the database but do not return data in the form of a result set.
449                 It returns an integer representing the number of rows affected by the operation.
450                 executeQuery():
451
452                 This method is used for retrieving data from the database. It is typically used
with SELECT statements, where the query returns a result set (data retrieved from the database).
453                 It returns a ResultSet object, which contains the data retrieved by the query.
454                 */
455                 // TO BECOME UPDATED YOUR TABLEVIEW
456                 addProductsShowListData();
457
458                 // CLEAR THE FIELDS
459                 addProductsReset();
460
461             }
462         }
463     } catch (Exception e) {

```

```

465         e.printStackTrace();
466     }
467
468 }
469
470 private String[] listType = {"Snacks", "Drinks", "Dessert", "Gadgets", "Personal Product",
471 "Others"};
472
473     public void addProductsListTypes() {
474
475         List<String> listT = new ArrayList<>();
476
477         for (String data : listType) {
478             listT.add(data);
479         }
480
481         ObservableList listData = FXCollections.observableArrayList(listT);
482         addProduct_productType_combo.setItems(listData);
483     }
484
485 private String[] listStatus = {"Available", "Not Available"};
486
487     public void addProductsListStatus() {
488         List<String> listS = new ArrayList<>();
489
490         for (String data : listStatus) {
491             listS.add(data);
492         }
493
494         ObservableList listData = FXCollections.observableArrayList(listS);
495         addProduct_status_combo.setItems(listData);
496     }
497
498     public void addproductsSelectedItemsToInputFields() {
499
500         ProductsModel productsData = addProduct_tableView.getSelectionModel().getSelectedItem(); //  

      TableView<ProductsModel> addProduct_tableView; nisa return wenneth ProductsModel object ekak  

501         int num = addProduct_tableView.getSelectionModel().getSelectedIndex();
502
503         if ((num - 1) < -1) {
504             return;
505         } // This condition checks if the selected index (num) is invalid. If no row is selected (i.e.,  

      num is -1), the method returns and exits without doing anything. The num - 1 < -1 check prevents trying  

      to access an invalid row.
506
507         addProduct_productId_txt.setText(String.valueOf(productsData.getProductId())); //  

      addProductsShowListData() Read this method to understand how the variable of the ProductsModel.java  

      class matched to the column of the table
508 //         addProduct_productType_combo.getSelectionModel().clearSelection();
509         addProduct_brand_txt.setText(productsData.getBrand());
510         addProduct_productName_txt.setText(productsData.getProductName());
511         addProduct_price_txt.setText(String.valueOf(productsData.getPrice()));
512 //         addProduct_status_combo.getSelectionModel().clearSelection();
513
514         String uri = "file:" + productsData.getImage();
515
516         image = new Image(uri, 115, 128, false, true);
517         addProduct_imgView.setImage(image);
518
519         GetData.path = productsData.getImage(); // After clicking the add button and entering the data  

      into the database, the path of GetData.java is cleared due to the execution of addproductReset(). Since

```



```

580             // TO BECOME UPDATED YOUR TABLEVIEW
581             addProductsShowListData();
582
583             // CLEAR THE FIELDS
584             addProductsReset();
585         }
586     }
587
588 } catch (Exception e) {
589     e.printStackTrace();
590 }
591 }
592
593 public void addProductsDelete() {
594
595     String sql = "DELETE FROM product WHERE product_id = '" + addProduct_productId_txt.getText() +
596     "'";
596     conn = databaseHandler.connectDb();
597
598     try {
599
600         Alert alert;
601
602         if (addProduct_productId_txt.getText().isEmpty()
603             || addProduct_productType_combo.getSelectionModel().getSelectedItem() == null
604             || addProduct_brand_txt.getText().isEmpty()
605             || addProduct_productName_txt.getText().isEmpty()
606             || addProduct_price_txt.getText().isEmpty()
607             || addProduct_status_combo.getSelectionModel().getSelectedItem() == null
608             || GetData.path == "") {
609
610             alert = new Alert(Alert.AlertType.ERROR);
611             alert.setTitle("Error Message");
612             alert.setHeaderText(null);
613             alert.setContentText("Please fill all blank fields");
614             alert.showAndWait();
615
616         } else {
617
618             alert = new Alert(Alert.AlertType.CONFIRMATION);
619             alert.setTitle("Confirmation Message");
620             alert.setHeaderText(null);
621             alert.setContentText("Are you sure you want to DELETE Product ID: " +
622             addProduct_productId_txt.getText() + " ?");
623
623             Optional<ButtonType> option = alert.showAndWait();
624
625             if (option.get().equals(ButtonType.OK)) {
626
627                 statement = conn.createStatement();
628                 statement.executeUpdate(sql);
629
630                 alert = new Alert(Alert.AlertType.INFORMATION);
631                 alert.setTitle("Information Message");
632                 alert.setHeaderText(null);
633                 alert.setContentText("Successfully Deleted");
634                 alert.showAndWait();
635
636                 // TO BECOME UPDATED YOUR TABLEVIEW
637                 addProductsShowListData();
638
639                 // CLEAR THE FIELDS
640                 addProductsReset();

```

```

641             }
642         }
643     } catch (Exception e) {
644         e.printStackTrace();
645     }
646 }
647
648 public void addProductsReset() {
649
650     addProduct_productId_txt.setText("");
651     addProduct_productType_combo.getSelectionModel().clearSelection();
652     addProduct_brand_txt.setText("");
653     addProduct_productName_txt.setText("");
654     addProduct_price_txt.setText("");
655     addProduct_status_combo.getSelectionModel().clearSelection();
656     addProduct_imgView.setImage(null);
657     GetData.path = "";
658 }
659 }
660
661 private Image image;
662
663 public void addProductsImportImage() {
664
665     FileChooser open = new FileChooser(); // This creates a new FileChooser object, which is a
JavaFX component that opens a dialog window to let the user browse and choose files from their
computer.
666     open.setTitle("Open Image File");
667     open.getExtensionFilters().add(new FileChooser.ExtensionFilter("Image File", "*jpg", "*png"));
// This line adds a filter to the FileChooser, so only image files with the extensions .jpg and .png
are shown in the file browser. The user will only be able to select these types of image files.
668
669     File file = open.showOpenDialog(main_form.getScene().getWindow()); // This is the method that
opens the file dialog window, allowing the user to browse and select an image file.
670
671     /*
672      --What is a Modal Window?--
673      A modal window is a type of window that temporarily blocks the interaction with the parent
window (the main application window) until the modal window is closed. This means:
674
675      While the modal window (in this case, the file dialog) is open, the user cannot interact with
the main window (e.g., click buttons, type in text fields) until they either select a file or close the
dialog.
676
677      This is helpful to ensure that the user completes the current action (choosing an image) before
interacting with the rest of the application.
678
679      ---Why Specify the Parent Window?--
680      The argument main_form.getScene().getWindow() is used to indicate the parent window of the file
dialog, which ties the file dialog to the main application window. Here's why this is important:
681
682      Blocking interaction with the main window:
683
684      When the file dialog is modal, it prevents the user from interacting with the parent window
until they complete the task in the dialog (choosing a file or closing the dialog). This ensures the
user focuses on selecting an image before doing anything else in the app.
685
686      Dialog behavior and position:
687
688      By specifying the parent window, the file dialog will appear centered relative to the main
window. It also ensures that when you minimize or move the main window, the file dialog will minimize
or move along with it.
689
690      Maintaining application flow:
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
999

```

```
689     If the user could interact with the main window while the file dialog was open, they might
  trigger unintended behaviors (such as clicking other buttons or opening additional dialogs), which
  could lead to confusion or errors in the application.
690
691     */
692     if (file != null) {
693
694         GetData.path = file.getAbsolutePath(); // On Windows, an absolute path might look like
  this: C:\Users\John\Pictures\image.jpg
695
696         image = new Image(file.toURI().toString(), 115, 128, false, true);
697         addProduct_imgView.setImage(image);
698     }
699 }
700 }
701
702 public void addProductsSearch() {
703
704     FilteredList<ProductsModel> filter = new FilteredList<>(addProductList, e -> true);
705
706     addProduct_search_txt.textProperty().addListener((Observable, oldValue, newValue) -> {
707
708         filter.setPredicate(predicateProductData -> {
709
710             if (newValue == null || newValue.isEmpty()) {
711                 return true;
712             }
713
714             String searchKey = newValue.toLowerCase();
715
716             if (predicateProductData.getProductId().toString().contains(searchKey)) {
717                 return true;
718             } else if (predicateProductData.getType().toLowerCase().contains(searchKey)) {
719                 return true;
720             } else if (predicateProductData.getBrand().toLowerCase().contains(searchKey)) {
721                 return true;
722             } else if (predicateProductData.getProductName().toLowerCase().contains(searchKey)) {
723                 return true;
724             } else if (predicateProductData.getPrice().toString().contains(searchKey)) {
725                 return true;
726             } else if (predicateProductData.getStatus().toLowerCase().contains(searchKey)) {
727                 return true;
728             } else {
729                 return false;
730             }
731         });
732     });
733
734     SortedList<ProductsModel> sortList = new SortedList<>(filter);
735
736     sortList.comparatorProperty().bind(addProduct_tableView.comparatorProperty());
737     addProduct_tableView.setItems(sortList);
738
739 }
740
741 public ObservableList<ProductsModel> addProductListDataGet() {
742
743     ObservableList<ProductsModel> productList = FXCollections.observableArrayList();
744
745     String sql = "SELECT * FROM product";
746     conn = databaseHandler.connectDb();
747
748     try {
```

```

749         prepare = conn.prepareStatement(sql);
750         resultSet = prepare.executeQuery();
751         ProductsModel proData;
752
753         while (resultSet.next()) {
754
755             proData = new ProductsModel(
756                 resultSet.getInt("product_id"),
757                 resultSet.getString("type"),
758                 resultSet.getString("brand"),
759                 resultSet.getString("product_name"),
760                 resultSet.getDouble("price"),
761                 resultSet.getString("status"),
762                 resultSet.getString("image"),
763                 resultSet.getDate("date")
764             );
765
766             productList.add(proData);
767         }
768     } catch (Exception e) {
769         e.printStackTrace();
770     }
771
772     return productList;
773 }
774
775 }
776
777
778     private ObservableList<ProductsModel> addProductList;
779 //     public void addProductsShowListData(){
780 //
781 //         addProductList = addProductListDataGet();
782 //
783 //         addProduct_tbvCol_productID.setCellValueFactory(new PropertyValueFactory<>("product_id")); // table columns name of database
784 //         addProduct_tbvCol_type.setCellValueFactory(new PropertyValueFactory<>("type"));
785 //         addProduct_tbvCol_brand.setCellValueFactory(new PropertyValueFactory<>("brand"));
786 //         addProduct_tbvCol_productName.setCellValueFactory(new PropertyValueFactory<>("product_name"));
787 //         addProduct_tbvCol_price.setCellValueFactory(new PropertyValueFactory<>("price"));
788 //         addProduct_tbvCol_status.setCellValueFactory(new PropertyValueFactory<>("status"));
789 //
790 //         addProduct_tableView.setItems(addProductList);
791 //     }
792
793     public void addProductsShowListData() { // show on table
794
795         addProductList = addProductListDataGet();
796
797         addProduct_tbvCol_productID.setCellValueFactory(new PropertyValueFactory<>("productId")); // productId is the variable name of the ProductsModel class
798         addProduct_tbvCol_type.setCellValueFactory(new PropertyValueFactory<>("type")); // this does not immediately "put all the data related to the product ID at once." Instead, what it does is link the productId field from each product in your list to the Product ID column of the TableView
799         addProduct_tbvCol_brand.setCellValueFactory(new PropertyValueFactory<>("brand"));
800         addProduct_tbvCol_productName.setCellValueFactory(new PropertyValueFactory<>("productName"));
801         addProduct_tbvCol_price.setCellValueFactory(new PropertyValueFactory<>("price"));
802         addProduct_tbvCol_status.setCellValueFactory(new PropertyValueFactory<>("status"));
803
804         addProduct_tableView.setItems(addProductList);
805     }
806

```

```

807     public void ordersAdd() {
808
809         customerId();
810         String sql = "INSERT INTO customer (customer_id, type, brand, productName, quantity, price,
811 date) "
812             + "VALUES(?, ?, ?, ?, ?, ?)";
813
814         conn = databaseHandler.connectDb();
815
816         try {
817
818             String checkData = "SELECT * FROM product WHERE product_name = ''"
819                 + orders_productName_combo.getSelectionModel().getSelectedItem() + "'";
820
821             double priceData = 0;
822
823             statement = conn.createStatement();
824             resultSet = statement.executeQuery(checkData);
825
826             if (resultSet.next()) {
827                 priceData = resultSet.getDouble("price"); // if there are more prices more than 1 then
what happen?????
828 // -----
829 }
830
831             double totalPdata = (priceData * qty);
832
833             Alert alert;
834             if (orders_productType_combo.getSelectionModel().getSelectedItem() == null
835                 || orders_brandName_combo.getSelectionModel().getSelectedItem() == null
836                 || orders_productName_combo.getSelectionModel().getSelectedItem() == null
837                 || totalPdata == 0) {
838                 alert = new Alert(Alert.AlertType.ERROR);
839                 alert.setTitle("Error Message");
840                 alert.setHeaderText(null);
841                 alert.setContentText("Please choose product first");
842                 alert.showAndWait();
843             } else {
844
845                 prepare = conn.prepareStatement(sql);
846                 prepare.setInt(1, customer_Id);
847                 prepare.setString(2, (String)
orders_productType_combo.getSelectionModel().getSelectedItem());
848                 prepare.setString(3, (String)
orders_brandName_combo.getSelectionModel().getSelectedItem());
849                 prepare.setString(4, (String)
orders_productName_combo.getSelectionModel().getSelectedItem());
850                 prepare.setString(5, String.valueOf(qty));
851
852                 prepare.setString(6, String.valueOf(totalPdata));
853
854                 Date date = new Date();
855                 java.sql.Date sqlDate = new java.sql.Date(date.getTime());
856                 prepare.setString(7, String.valueOf(sqlDate));
857
858                 prepare.executeUpdate();
859
860                 ordersShowListData();
861 //             ordersDisplayTotal();
862
863             }
864         } catch (Exception e) {

```

```
865         e.printStackTrace();
866     }
867
868 }
869
870 public void ordersPay() {
871
872     customerId();
873     String sql = "INSERT INTO customer_receipt (customer_id, total, amount, balance, date) "
874         + "VALUES(?, ?, ?, ?, ?)" // WHERE customer_id = '"+ customer_Id +"'" // change this I
875
876     conn = databaseHandler.connectDb();
877
878     try {
879
880         Alert alert;
881         if (totalP > 0) { // || orders_amount_txt.getText().isEmpty() || amount == 0
882
883             alert = new Alert(Alert.AlertType.CONFIRMATION);
884             alert.setTitle("Confirmation Message");
885             alert.setHeaderText(null);
886             alert.setContentText("Are you sure?");
887             Optional<ButtonType> optional = alert.showAndWait();
888
889             if (optional.get().equals(ButtonType.OK)) {
890
891                 prepare = conn.prepareStatement(sql);
892
893                 prepare.setInt(1, customer_Id); // I change this
894                 prepare.setString(2, String.valueOf(totalP));
895                 prepare.setString(3, String.valueOf(amount));
896                 prepare.setString(4, String.valueOf(balance));
897
898                 Date date = new Date();
899                 java.sql.Date sqlDate = new java.sql.Date(date.getTime());
900                 prepare.setString(5, String.valueOf(sqlDate));
901
902                 prepare.executeUpdate();
903
904                 alert = new Alert(Alert.AlertType.INFORMATION);
905                 alert.setTitle("Information Message");
906                 alert.setHeaderText(null);
907                 alert.setContentText("Successfully Payed!");
908                 alert.showAndWait();
909
910 //                 ordersShowListData();
911
912 //                 totalP = 0;
913                 balance = 0;
914                 amount = 0;
915
916                 orders_balance_lbl.setText("Rs 0.0");
917                 orders_total_lbl.setText("Rs 0.0");
918                 orders_amount_txt.setText("");
919
920             } else {
921                 return;
922             }
923
924     } else {
925
926         alert = new Alert(Alert.AlertType.ERROR);
927         alert.setTitle("Error Message");
```

```

928         alert.setHeaderText(null);
929         alert.setContentText("Invalid : 3");
930         alert.showAndWait();
931     }
932 }
933 } catch (Exception e) {
934     e.printStackTrace();
935 }
936 }
937 }
938 }
939 }
940 public void ordersReceipt() throws JRException{
941     try {
942         customerId();
943         // customerId();
944         HashMap hash = new HashMap();
945         hash.put("inventoryP", customer_Id);
946
947         if(totalP == 0){
948             Alert alert = new Alert(Alert.AlertType.ERROR);
949             alert.setTitle("Error Message");
950             alert.setHeaderText(null);
951             alert.setContentText("Invalid :3"); // kalin order eka pay karala iware nam den totalP
952             = 0 ema nisa receipt ekak print kranna denne aye order ekak daddi
953             alert.showAndWait();
954
955         }else{
956             JasperDesign jDesign =
957             JRXmlLoader.load("C:\\\\Users\\\\krith\\\\Downloads\\\\YT_InventoryManagementSystem_JavaFx-
958             main\\\\YT_InventoryManagementSystem_JavaFx-main\\\\report.jrxml");
959             JasperReport jReport = JasperCompileManager.compileReport(jDesign);
960             JasperPrint jPrint = JasperFillManager.fillReport(jReport, hash, conn);
961
962             JasperViewer.viewReport(jPrint, false);
963
964         }
965
966     } catch (Exception e) {e.printStackTrace();}
967 }
968 }
969 }
970 }
971 }
972 public void ordersReset(){
973     customerId();
974
975     String sql = "DELETE FROM customer WHERE customer_id = '"+ customer_Id +"'";
976
977     conn = databaseHandler.connectDb();
978
979     try {
980         if(!orders_tableView.getItems().isEmpty()){
981
982             Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
983             alert.setTitle("Confirmation Message");
984             alert.setHeaderText(null);
985             alert.setContentText("Are you sure you want to reset?");
986             Optional<ButtonType> option = alert.showAndWait();
987

```

```

988
989         if(option.get().equals(ButtonType.OK)){
990
991             statement = conn.createStatement();
992             statement.executeUpdate(sql);
993
994             ordersShowListData();
995
996             totalP = 0;
997             balance = 0;
998             amount = 0;
999
1000            orders_balance_lbl.setText("Rs 0.0");
1001            orders_total_lbl.setText("Rs 0.0");
1002            orders_amount_txt.setText("");
1003        }
1004
1005    }
1006    } catch (Exception e) {e.printStackTrace();}
1007
1008 }
1009
1010 private double amount;
1011 private double balance;
1012
1013 public void ordersAmount() {
1014
1015     Alert alert;
1016
1017     if (!orders_amount_txt.getText().isEmpty()) {
1018
1019         amount = Double.parseDouble(orders_amount_txt.getText());
1020
1021         if (totalP > 0) {
1022             if (amount >= totalP) {
1023                 balance = (amount - totalP);
1024                 orders_balance_lbl.setText("Rs " + String.valueOf(balance));
1025             } else {
1026                 alert = new Alert(Alert.AlertType.ERROR);
1027                 alert.setHeaderText(null);
1028                 alert.setContentText("Enter Enough Amount");
1029                 alert.showAndWait();
1030
1031                 orders_amount_txt.setText("");
1032             }
1033         } else {
1034
1035             alert = new Alert(Alert.AlertType.ERROR);
1036             alert.setHeaderText(null);
1037             alert.setContentText("Invalid :3");
1038             alert.showAndWait();
1039
1040         }
1041     } else {
1042
1043         alert = new Alert(Alert.AlertType.ERROR);
1044         alert.setHeaderText(null);
1045         alert.setContentText("Amount Field is Empty");
1046         alert.showAndWait();
1047
1048     }
1049 }
1050

```

```

1051     private double totalP;
1052
1053     public void ordersDisplayTotal() {
1054         customerId();
1055
1056         String sql = "SELECT SUM(price) FROM customer WHERE customer_id = '" + customer_Id + "'";
1057         // mehidee cus id of customer table is 1, and cus id of customer_receipt is 0, customerId() ekedi
1058         // customer table eke customer id eka select krala gannwa, eka thamay methana thiyyenne "+customer_Id+"'
1059         // lesa.
1060         conn = databaseHandler.connectDb();
1061
1062         try {
1063
1064             prepare = conn.prepareStatement(sql);
1065             resultSet = prepare.executeQuery();
1066
1067             while (resultSet.next()) {
1068                 totalP = resultSet.getDouble("SUM(price)");
1069             }
1070
1071             orders_total_lbl.setText("Rs " + String.valueOf(totalP));
1072
1073         } catch (Exception e) {
1074             e.printStackTrace();
1075         }
1076
1077     }
1078
1079     private String[] OrderslistType = {"Snacks", "Drinks", "Dessert", "Gadgets", "Personal
1080     Product", "Others"};
1081
1082     public void OrdersListTypes() {
1083         // orders_productType_combo.getSelectionModel().clearSelection();
1084         List<String> listT = new ArrayList<>();
1085
1086         for (String data : OrderslistType) {
1087             listT.add(data);
1088         }
1089
1090         ObservableList listData = FXCollections.observableArrayList(listT);
1091         orders_productType_combo.setItems(listData);
1092
1093         ordersListBrands();
1094     }
1095
1096     public void ordersListBrands() { // this method call after type selected, but when selected
1097     // brand name combo we want call again this method, because we want to show product name combo list
1098
1099         // Clear the product name combo box when a new type is selected
1100         orders_productName_combo.getSelectionModel().clearSelection();
1101         orders_productName_combo.getItems().clear();
1102
1103         // Check if brand selection exists before executing the query
1104         if (orders_productType_combo.getSelectionModel().getSelectedItem() == null) {
1105             return; // Exit the method if no brand is selected
1106         }
1107
1108         String sql = "SELECT * FROM product WHERE type = "
1109             + orders_productType_combo.getSelectionModel().getSelectedItem()

```

```

1109             + "" and status = 'Available"'; // GROUP BY brand
1110
1111     conn = databaseHandler.connectDb();
1112
1113     try {
1114
1115         prepare = conn.prepareStatement(sql);
1116         resultSet = prepare.executeQuery();
1117
1118         ObservableList listData = FXCollections.observableArrayList();
1119
1120         while (resultSet.next()) {
1121             listData.add(resultSet.getString("brand"));
1122         }
1123
1124         orders_brandName_combo.setItems(listData);
1125
1126         ordersListProductName();
1127     } catch (Exception e) {
1128         e.printStackTrace();
1129     }
1130
1131 }
1132
1133 public void ordersListProductName() {
1134
1135     // Check if brand selection exists before executing the query
1136     if (orders_brandName_combo.getSelectionModel().getSelectedItem() == null) {
1137         return; // Exit the method if no brand is selected
1138     }
1139
1140     String sql = "SELECT * FROM product WHERE brand = ''"
1141         + orders_brandName_combo.getSelectionModel().getSelectedItem() + "'";
1142
1143     conn = databaseHandler.connectDb();
1144
1145     try {
1146
1147         prepare = conn.prepareStatement(sql);
1148         resultSet = prepare.executeQuery();
1149
1150         ObservableList listData = FXCollections.observableArrayList();
1151
1152         while (resultSet.next()) {
1153             listData.add(resultSet.getString("product_name"));
1154         }
1155
1156         orders_productName_combo.setItems(listData);
1157
1158     //     while(resultSet.next()){
1159     //         customerData = new CustomerData(
1160     //             resultSet.getInt("customer_id"),
1161     //             resultSet.getString("type"),
1162     //             resultSet.getString("brand"),
1163     //             resultSet.getString("productName"),
1164     //             resultSet.getInt("quantity"),
1165     //             resultSet.getDouble("price"),
1166     //             resultSet.getDate("date"));
1167     //     }
1168     //         listData.add(customerData);
1169     //     }
1170 } catch (Exception e) {
1171     e.printStackTrace();

```

```

1172         }
1173     }
1174
1175     private SpinnerValueFactory<Integer> spinner;
1176
1177     public void ordersSpinner() {
1178
1179         spinner = new SpinnerValueFactory.IntegerSpinnerValueFactory(0, 20, 0);
1180         orders_quantity_spinner.setValueFactory(spinner);
1181
1182         /*
1183
1184         --- ordersSpinner() Method ---
1185
1186         This method is used to set up a Spinner for ordering quantities, allowing the user to
1187         select a number between 0 and 20. Here's a detailed explanation of each part of this method:
1188
1189         1. Spinner Initialization:
1190
1191         spinner = new SpinnerValueFactory.IntegerSpinnerValueFactory(0, 20, 0);
1192
1193         SpinnerValueFactory.IntegerSpinnerValueFactory(0, 20, 0) creates an integer-based spinner
1194         with a minimum value of 0, a maximum value of 20, and an initial/default value of 0.
1195         The SpinnerValueFactory helps manage the range and value of the spinner for quantity
1196         selection.
1197
1198         2. Assigning the Spinner to the orders_quantity_spinner:
1199
1200         orders_quantity_spinner.setValueFactory(spinner);
1201
1202         Here, orders_quantity_spinner is a spinner UI element (likely created in the FXML file or
1203         elsewhere in your JavaFX UI code).
1204         This line assigns the spinner factory as the value factory for orders_quantity_spinner.
1205         This means that the spinner's value range (0 to 20) and starting value (0) are now applied to
1206         orders_quantity_spinner.
1207
1208         3. Placement in Initialization or Button Click:
1209
1210         ordersSpinner() is called in the initialize() method (often used in JavaFX controllers to
1211         initialize UI elements when the scene is loaded) or on a button click.
1212         When called in initialize(), it sets up the spinner with the specified range and default
1213         value as soon as the interface is loaded, making it ready for user interaction.
1214
1215         */
1216
1217     }
1218
1219     private int qty;
1220
1221     public void ordersShowSpinnerValue() {
1222         qty = orders_quantity_spinner.getValue();
1223     }
1224
1225     public ObservableList<CustomerData> ordersListData() {
1226         customerId(); // pay button eke explain ekata meka yodaganna
1227         ObservableList<CustomerData> listData = FXCollections.observableArrayList();
1228         String sql = "SELECT * FROM customer WHERE customer_id = '" + customer_Id + "' ";
1229
1230         conn = databaseHandler.connectDb();
1231
1232         try {

```

```
1227
1228     CustomerData customerData;
1229     prepare = conn.prepareStatement(sql);
1230     resultSet = prepare.executeQuery();
1231
1232     while (resultSet.next()) {
1233         customerData = new CustomerData(
1234             resultSet.getInt("customer_id"),
1235             resultSet.getString("type"),
1236             resultSet.getString("brand"),
1237             resultSet.getString("productName"),
1238             resultSet.getInt("quantity"),
1239             resultSet.getDouble("price"),
1240             resultSet.getDate("date"));
1241
1242         listData.add(customerData);
1243     }
1244
1245     } catch (Exception e) {
1246         e.printStackTrace();
1247     }
1248
1249     return listData;
1250 }
1251
1252 private ObservableList<CustomerData> orderList;
1253
1254 public void ordersShowListData() {
1255
1256     orderList = ordersListData();
1257
1258     orders_col_type.setCellValueFactory(new PropertyValueFactory<>("type"));
1259     orders_col_brand.setCellValueFactory(new PropertyValueFactory<>("brand"));
1260     orders_col_productName.setCellValueFactory(new PropertyValueFactory<>("productName"));
1261     orders_col_quantity.setCellValueFactory(new PropertyValueFactory<>("quantity"));
1262     orders_col_price.setCellValueFactory(new PropertyValueFactory<>("price"));
1263
1264     orders_tableView.setItems(orderList);
1265
1266     ordersDisplayTotal();
1267 }
1268
1269 private int customer_Id;
1270
1271 public void customerId() {
1272
1273     String customerId = "SELECT * FROM customer";
1274     conn = databaseHandler.connectDb();
1275
1276     try {
1277
1278         prepare = conn.prepareStatement(customerId);
1279         resultSet = prepare.executeQuery();
1280
1281         int checkId = 0;
1282
1283         while (resultSet.next()) {
1284
1285             // GET LAST CUSTOMER ID
1286             customer_Id = resultSet.getInt("customer_id");
1287
1288         }
1289

```

```

1290         String checkData = "SELECT * FROM customer_receipt";
1291
1292         statement = conn.createStatement();
1293         resultSet = statement.executeQuery(checkData);
1294
1295         while (resultSet.next()) {
1296             checkId = resultSet.getInt("customer_id");
1297         }
1298
1299         if (customer_Id == 0) {
1300             customer_Id += 1;
1301         } else if (checkId == customer_Id) {
1302             customer_Id += 1;
1303         }
1304
1305         /*
1306
1307             above block checks if a unique customer_Id should be assigned:
1308
1309             if(customer_Id == 0) checks if there were no customer_id records in the customer table
1310             and increments customer_Id by 1.
1311             else if(checkId == customer_Id) checks if the last customer_id from customer_receipt
1312             matches customer_Id. If they are the same, it increments customer_Id by 1 to avoid duplication.
1313
1314         */
1315         } catch (Exception e) {
1316             e.printStackTrace();
1317         }
1318     public void switchForm(ActionEvent event) {
1319
1320         if (event.getSource() == home_btn) {
1321
1322             home_form.setVisible(true);
1323             addProducts_form.setVisible(false);
1324             orders_form.setVisible(false);
1325
1326             home_btn.setStyle("-fx-background-color:linear-gradient(to bottom right, #269e70,
1327 #969635);");
1328             addProducts_btn.setStyle("-fx-background-color:transparent;");
1329             orders_btn.setStyle("-fx-background-color:transparent;");
1330
1331             homeDisplayTotalOrders();
1332             homeTotalIncome();
1333             homeAvailableProducts();
1334             homeIncomeChart();
1335             homeOrdersChart();
1336
1337         } else if (event.getSource() == addProducts_btn) {
1338
1339             home_form.setVisible(false);
1340             addProducts_form.setVisible(true);
1341             orders_form.setVisible(false);
1342
1343             home_btn.setStyle("-fx-background-color:transparent;");
1344             addProducts_btn.setStyle("-fx-background-color:linear-gradient(to bottom right,
1345 #269e70, #969635);");
1346             orders_btn.setStyle("-fx-background-color:transparent;");
1347
1348             addProductsShowListData();
1349             addProductsListTypes();
1350             addProductsListStatus();

```

```

1349         addProductsSearch();
1350
1351     } else if (event.getSource() == orders_btn) {
1352
1353         home_form.setVisible(false);
1354         addProducts_form.setVisible(false);
1355         orders_form.setVisible(true);
1356
1357         home_btn.setStyle("-fx-background-color:transparent;");
1358         addProducts_btn.setStyle("-fx-background-color:transparent;");
1359         orders_btn.setStyle("-fx-background-color:linear-gradient(to bottom right, #269e70,
#969635);");
1360
1361         orders_productType_combo.getSelectionModel().clearSelection();
1362         orders_brandName_combo.getSelectionModel().clearSelection();
1363         orders_productName_combo.getSelectionModel().clearSelection();
1364
1365         ordersShowListData();
1366         OrdersListTypes();
1367         ordersListBrands();
1368         ordersListProductName();
1369         ordersSpinner();
1370
1371     }
1372 }
1373
1374 public void defaultNav(){
1375     home_btn.setStyle("-fx-background-color:linear-gradient(to bottom right, #269e70,
#969635);"); // software eka run karala home page eka penne, ema nisa home_btn eka click unama watena
pata watila tyena one muladima
1376 }
1377
1378 private double x = 0;
1379 private double y = 0;
1380
1381 public void logout() {
1382     try {
1383         Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
1384         alert.setTitle("Confirmation Message");
1385         alert.setHeaderText(null);
1386         alert.setContentText("Are you sure want to logout?");
1387         Optional<ButtonType> option = alert.showAndWait();
1388
1389         if (option.get().equals(ButtonType.OK)) {
1390
1391             main_form.getScene().getWindow().hide();
1392
1393             // LINK YOUR LOGIN FORM
1394             Parent root = FXMLLoader.load(getClass().getResource("/fxml/Login.fxml"));
1395             Stage stage = new Stage();
1396             Scene scene = new Scene(root);
1397
1398             root.setOnMousePressed((MouseEvent event) -> {
1399                 x = event.getSceneX();
1400                 y = event.getSceneY();
1401             });
1402
1403             root.setOnMouseDragged((MouseEvent event) -> {
1404                 stage.setX(event.getScreenX() - x);
1405                 stage.setY(event.getScreenY() - y);
1406
1407                 stage.setOpacity(0.8);
1408             });

```

```

1409             root.setOnMouseReleased((MouseEvent event) -> {
1410                 stage.setOpacity(1);
1411             });
1412         });
1413
1414         stage.initStyle(StageStyle.TRANSPARENT);
1415
1416         stage.setScene(scene);
1417         stage.show();
1418     } else {
1419         return;
1420     }
1421
1422     } catch (Exception e) {
1423         e.printStackTrace();
1424     }
1425 }
1426
1427 public void setUsername(){
1428     username_lbl.setText(GetData.username);
1429 }
1430
1431 public void minimize() {
1432     Stage stage = (Stage) main_form.getScene().getWindow();
1433     stage.setIconified(true);
1434 }
1435
1436 public void close() {
1437     System.exit(0);
1438 }
1439
1440 @Override
1441 public void initialize(URL url, ResourceBundle rb) {
1442
1443     setUsername();
1444     defaultNav();
1445
1446     homeDisplayTotalOrders();
1447     homeTotalIncome();
1448     homeAvailableProducts();
1449     homeIncomeChart();
1450     homeOrdersChart();
1451
1452     // TO SHOW THE DATA ON TABLEVIEW
1453     addProductsShowListData();
1454     addProductsListTypes();
1455     addProductsListStatus();
1456
1457     ordersShowListData();
1458     OrdersListTypes();
1459     ordersListBrands();
1460     ordersListProductName();
1461     ordersSpinner();
1462 }
1463
1464 }
```

3.3App :

```

import java.io.IOException;
import javafx.scene.input.MouseEvent;
import javafx.stage.StageStyle;

/**
 * JavaFX App
 */
public class App extends Application {

    private static Scene scene;
    private static Parent root;
    private double x = 0;
    private double y = 0;

    @Override
    public void start(Stage stage) throws IOException {
        Parent root = FXMLLoader.load(getClass().getResource("/fxml/Login.fxml"));
        scene = new Scene(root, 623, 439); // Specify the fxml folder

        root.setOnMousePressed((MouseEvent event) -> {
            x = event.getSceneX();
            y = event.getSceneY();
        });

        root.setOnMouseDragged((MouseEvent event) -> {
            stage.setX(event.getScreenX() - x);
            stage.setY(event.getScreenY() - y);

            stage.setOpacity(0.8);
        });

        root.setOnMouseReleased((MouseEvent event) -> {
            stage.setOpacity(1);
        });

        stage.initStyle(StageStyle.TRANSPARENT);

        stage.setScene(scene);
        stage.show();
    }

    static void setRoot(String fxml) throws IOException {
        scene.setRoot(loadFXML("fxml/" + fxml)); // Specify the fxml folder
    }

    private static Parent loadFXML(String fxml) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(App.class.getResource("/" + fxml + ".fxml")); // Include leading slash for resources
        return fxmlLoader.load();
    }

    public static void main(String[] args) {
        launch();
    }
}

```

3.4 Customer data

```

package com.mycompany.yt_inventorymanagementsystem_javafx;

import java.util.Date;

/**

```

```
*  
* @author USER  
*/  
public class CustomerData {  
  
    private Integer customerId;  
    private String type;  
    private String brand;  
    private String productName;  
    private Integer quantity;  
    private Double price;  
    private Date date;  
  
    public CustomerData(Integer customerId, String type, String brand, String productName, Integer quantity, Double price, Date date) {  
        this.customerId = customerId;  
        this.type = type;  
        this.brand = brand;  
        this.productName = productName;  
        this.quantity = quantity;  
        this.price = price;  
        this.date = date;  
    }  
  
    public Integer getCustomerId() {  
        return customerId;  
    }  
  
    public String getType() {  
        return type;  
    }  
  
    public String getBrand() {  
        return brand;  
    }  
  
    public String getProductName() {  
        return productName;  
    }  
  
    public Integer getQuantity() {  
        return quantity;  
    }  
  
    public Double getPrice() {  
        return price;  
    }  
  
    public Date getDate() {  
        return date;  
    }  
}
```

3.5 Primary

```
4 package com.mycompany.yt_inventorymanagementsystem_javafx;
5
6 import java.io.IOException;
7 import javafx.fxml.FXML;
8
9 public class PrimaryController {
10
11     @FXML
12     private void switchToSecondary() throws IOException {
13         App.setRoot("secondary");
14     }
15 }
```

3.6 Products model

```
package com.mycompany.yt_inventorymanagementsystem_javafx;

import java.sql.Date;

/**
 *
 * @author USER
 */
public class ProductsModel {

    private Integer productId;
    private String type;
    private String brand;
    private String productName;
    private Double price;
    private String status;
    private String image;
    private Date date; // It represents only the date (year, month, and day) without any time information (hours, minutes, seconds, or milliseconds).

    public ProductsModel(Integer productId, String type, String brand, String productName, Double price, String status, String image, Date date) {
        this.productId = productId;
        this.type = type;
        this.brand = brand;
        this.productName = productName;
        this.price = price;
        this.status = status;
        this.image = image;
        this.date = date;
    }

    public Integer getProductId() {
        return productId;
    }

    public String getType() {
        return type;
    }

    public String getBrand() {
        return brand;
    }

    public String getProductName() {
        return productName;
    }
}
```

```

public Double getPrice() {
    return price;
}

public String getStatus() {
    return status;
}

public String getImage() {
    return image;
}

public Date getDate() {
    return date;
}

}

```

3.7 Module info :

```

module com.mycompany.yt_inventorymanagementsystem_javafx {
    requires javafx.controls;
    requires javafx.fxml;
    requires java.sql;
    requires de.jensd.fx.glyphs.fontawesome;
    requires java.base;
    requires jasperreports;

    opens com.mycompany.yt_inventorymanagementsystem_javafx to javafx.fxml;
    exports com.mycompany.yt_inventorymanagementsystem_javafx;
}

```

SNAPSHOTS:

4.1 login page

x



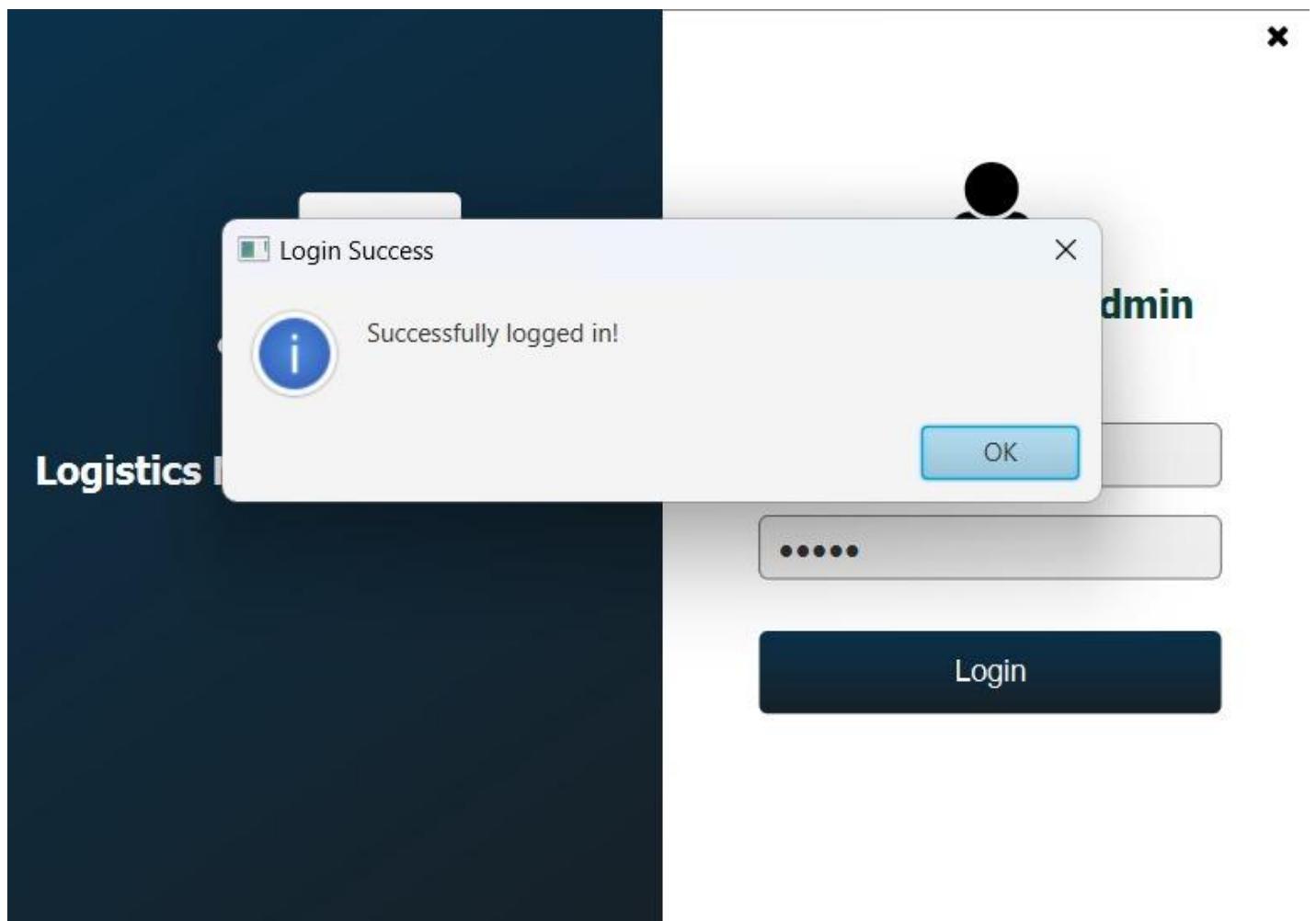
Logistics Management System



Welcome Back Admin

A text input field with a placeholder character 'I'.A password input field with the placeholder text 'Password'.

Login





Welcome
krithika

Home

+ Add Products

Orders



Sign Out



Today's Number of Orders

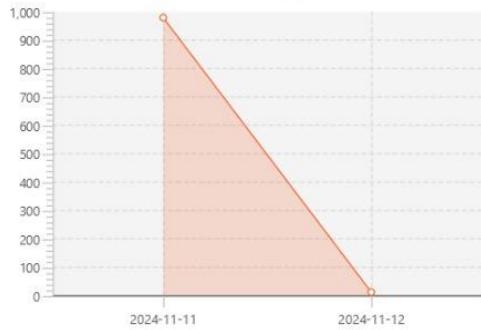


Rs 987.56



Available Products

Income Data Chart



Order Data Chart



4.4 entering orders

The screenshot shows the 'Logistics Management System' interface. On the left, a sidebar displays a user profile icon, the welcome message 'Welcome krithika', navigation links for 'Home', 'Add Products', and 'Orders' (which is highlighted in green), and a 'Sign Out' button. The main area contains a table with columns: Type, Brand, Product Name, Quantity, and Price. A single row is present with values: Snacks, grb, chips, 2, and 20.0. To the right of the table are input fields for 'Product Type' (Snacks), 'Brand Name' (grb), 'Product Name' (chips), and 'Quantity' (1). Below these is an 'Add' button. Further down, the total amount 'Total: Rs 20.0' is displayed, along with 'Amount:' and 'Balance: Rs 0.0' fields, and 'Pay' and 'Reset' buttons.

Type	Brand	Product Name	Quantity	Price
Snacks	grb	chips	2	20.0

Product Type: Snacks

Brand Name: grb

Product Name: chips

Quantity: 1

Add

Total: Rs 20.0

Amount:

Balance: Rs 0.0

Pay

Reset

4.5 adding products

The screenshot shows the 'Logistics Management System' interface. On the left sidebar, there is a user icon, a 'Welcome krithika' message, and navigation links for 'Home', '+ Add Products' (highlighted in green), 'Orders', and 'Sign Out'. The main content area has a title 'Import' with input fields for Product ID, Product Type (dropdown: Choose), Brand, Product Name, Status (dropdown: Choose), and Price. Below these are four buttons: 'Add' (blue), 'Update' (green), 'Reset' (purple), and 'Delete' (red). To the right is a search bar and a table listing products:

Product ID	Type	Brand	Product Name	Price	Status
1	Snacks	grb	chips	10.0	Available
2	Drinks	coke	soda	50.0	Not Avail...
3	Dessert	grb	mysoreoak	250.09	Not Avail...
4	Personal Pro...	ntg	ntg	12.0	Available

4.6 logout successful

The screenshot shows the 'Logistics Management System' interface after a successful logout. The left sidebar remains the same. The main content area displays a table with a single row: Type (Snacks), Brand (grb), Product Name (chips), Quantity (2), and Price (20.0). A confirmation dialog box is centered over the table, asking 'Are you sure want to logout?' with 'OK' and 'Cancel' buttons. To the right of the table, there is a form for adding a new product entry:

Product Type: Snacks
Brand Name: grb
Product Name: chips
Quantity: 1

Add

Total: Rs 20.0
Amount:
Balance: Rs 0.0

Pay
Reset

Conclusion

With the help of our project, businesses can efficiently manage their logistics operations by tracking inventory, coordinating with suppliers, and processing orders seamlessly. The system organizes all critical data in one centralized platform, accessible through their accounts, saving time and ensuring a more streamlined, professional approach to logistics management.

Reference

1. <https://www.javatpoint.com/java-tutorial>
2. <https://www.wikipedia.org/>
3. <https://www.w3schools.com/sql/>
4. <https://www.codecademy.com/learn/learn-sql>