Status	Finished
Started	Monday, 18 November 2024, 8:04 PM
Completed	Monday, 18 November 2024, 8:23 PM
Duration	18 mins 56 secs

Question **1**

Correct

Marked out of 5.00

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

Get the ASCII values of all the extracted alphabets.

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

Return that single digit as output.

Note:

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

Example 1:

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

input2: {'b', 'c'}

output: 8

Explanation:

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

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

98 + 99 = 197

1 + 9 + 7 = 17

1 + 7 = 8

For example:

Input	Result
a b c b c	8

Answer: (penalty regime: 0 %)

```
import java.util.HashSet;
import java.util.Set;
public class CommonCharacterSum {
public static void main(String[] args) {

// Test input
 char[] input1 = {'a', 'b', 'c'};
 char[] input2 = {'b', 'c'};
 // Call the method to get the single digit sum
 int result = getSingleDigitSum(input1, input2);
 // Output the result
 System.out.println(result);
    }

 // Add elements to the sets
 public static int getSingleDigitSum(char[] input1, char[] input2) {

 Set<Character> set1 = new HashSet<>();
 Set<Character> set2 = new HashSet<>();// Create sets to store unique characters from each array
 for (char c : input1) set1.add(c);
 for (char c : input2) set2.add(c);
set1.retainAll(set2);
```

```
23  int sum1 = 0;
24  for (char c : set1) {
25     sum1+=(int)c;
26
27     }
28
29  while (sum1 >= 10) {
30  sum1 = sumDigits(sum1);
31  }// Sum digits until a single digit is obtained
32  return sum1;
33  }
34  // Helper method to sum the digits of a number
35  public static int sumDigits(int num) {
36  int sum = 0;
37  while (num > 0) {
38 sum += num % 10;
39  num /= 10;
40     }
41  return sum;
42  }
43  }
44
```

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

Passed all tests! ✓

Question **2**

Correct

Marked out of 5.00

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

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

Each alphabet is represented by a sequence of 0s.

This is as mentioned below:

Z : 0

Y : 00

X : 000

W : 0000

V : 00000

U : 000000

T : 0000000

and so on upto A having 26 0's (00000000000000000000000000).

The sequence of 0's in the encoded form are separated by a single 1 which helps to distinguish between 2 letters.

Example 1:

input1: 010010001

The decoded string (original word) will be: ZYX

Example 2:

input1: 00001000000000000000001000000000001000000000100000000000001

The decoded string (original word) will be: WIPRO

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

For example:

Input	Result
010010001	ZYX
00001000000000000000001000000000001000000000100000000000001	WIPRO

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;

public class DecodeString {
    public static String decode(String input) {
        // Split the input string by '1' to separate encoded letters
        String[] encodedLetters = input.split("1");
        StringBuilder decodedWord = new StringBuilder();

        for (String letter : encodedLetters) {
            // Count the number of '0's in each part
            int countZeros = letter.length();

            if (countZeros > 0) {
                // Map the count of '0's to the corresponding alphabet
                char decodedChar = (char) ('Z' - countZeros + 1);
                decodedWord.append(decodedChar);
            }
        }

        return decodedWord.toString();
```

```
        return decodedWord.toString();
    }

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

        // Prompt user to enter the encoded string

        String input = scanner.nextLine();

        // Decode the string and display the result
        String decodedString = decode(input);
        System.out.println(decodedString);

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

	Input	Expected	Got	
✓	010010001	ZYX	ZYX	✓
✓	00001000000000000000001000000000001000000000100000000000001	WIPRO	WIPRO	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of 5.00

Write a function that takes an input String (sentence) and generates a new String (modified sentence) by reversing the words in the original String, maintaining the words position.

In addition, the function should be able to control the reversing of the case (upper or lowercase) based on a case_option parameter, as follows:

If case_option = 0, normal reversal of words i.e., if the original sentence is "Wipro TechNologies BangaLore", the new reversed sentence should be "orpiW seigoloNhceT eroLagnaB".

If case_option = 1, reversal of words with retaining position's case i.e., if the original sentence is "Wipro TechNologies BangaLore", the new reversed sentence should be "Orpiw 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 %)

```
import java.util.Scanner;

public class Main {

    public static String reverseEachWordInString(String str, int caseOption) {
        StringBuilder revString = new StringBuilder();
        String[] words = str.split(" "); // Split the string into words

        for (String word : words) {
            StringBuilder reverseWord = new StringBuilder(word).reverse(); // Reverse the word
```

```
            // Handle commas at the end of the word
            if (reverseWord.length() > 0 && reverseWord.charAt(reverseWord.length() - 1) == ',') {
                char lastChar = reverseWord.charAt(reverseWord.length() - 1);
                reverseWord.deleteCharAt(reverseWord.length() - 1);
                reverseWord.reverse();
                reverseWord.append(lastChar);
            }

            // Handle case transformation
            if (caseOption == 1) { // Convert to "First Letter Capitalized"
                for (int i = 0; i < word.length(); i++) {
                    if (Character.isUpperCase(word.charAt(i))) {
                        reverseWord.setCharAt(i, Character.toUpperCase(reverseWord.charAt(i)));
                    } else if (Character.isLowerCase(word.charAt(i))) {
                        reverseWord.setCharAt(i, Character.toLowerCase(reverseWord.charAt(i)));
                    }
                }
            }

            // Append the processed word to the result
            revString.append(reverseWord).append(" ");
        }

        return revString.toString().trim(); // Return the final reversed string
    }

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

        // Read input string

        String strGiven = scanner.nextLine();

        // Read the case option (0 or 1)

        int caseOption = scanner.nextInt();

        // Get the reversed and processed string
        String result = reverseEachWordInString(strGiven, caseOption);

        // Output the result
```

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

Passed all tests! ✓

◄ Lab-12-MCQ

Jump to...

Identify possible words ►

Status	Finished
Started	Monday, 18 November 2024, 7:34 PM
Completed	Monday, 18 November 2024, 8:04 PM
Duration	29 mins 30 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   class prog {
4    public static void main(String[] args) {
5      Scanner sc= new Scanner(System.in);
6      int n = sc.nextInt();
7      // Create a HashSet object called numbers
8      HashSet<Integer> numbers=new HashSet<>();
9
10     // Add values to the set
11     for(int i=0;i<n;i++)
12     numbers.add(sc.nextInt());
13
14   int skey=sc.nextInt();
15
16     // Show which numbers between 1 and 10 are in the set
17
18       if(numbers.contains(skey))                        {
19         System.out.println( skey+ " was found in the set.");
20       } else {
```
```

```
21            System.out.println(skey +   was not found in the set.   );
22          }
23        }
24      }
25
```

	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 %)

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

class prog
{
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        int n1=s.nextInt();
        Set<String> set1=new HashSet<>();
        Set<String> set2=new HashSet<>();
        s.nextLine();
        for(int i=0;i<n1;i++)
        {
            set1.add(s.nextLine());
        }int n2=s.nextInt();
        s.nextLine();
        for(int i=0;i<n2;i++)
        {
            set2.add(s.nextLine());
        }
        set1.retainAll(set2);
        for(String elements:set1)
        {
            System.out.println(elements);
        }
    }
```

```
29 }
```

	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

```
import java.util.HashMap;
import java.util.Map.Entry;
import java.util.Set;
import java.util.Scanner;
 class prog
{
    public static void main(String[] args)
    {
        //Creating HashMap with default initial capacity and load factor
        HashMap<String, Integer> map = new HashMap<String, Integer>();

        String name;
        int num;
        Scanner sc= new Scanner(System.in);
        int n=sc.nextInt();
         for(int i =0;i<n;i++)
         {
             name=sc.next();
             num= sc.nextInt();
             map.put(name,num);
         }

        //Printing key-value pairs

        Set<Entry<String, Integer>> entrySet = map.entrySet();

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

        HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();

        //Inserting key-value pairs to anotherMap using put() method

        anotherMap.put("SIX", 6);

        anotherMap.put("SEVEN", 7);

        //Inserting key-value pairs of map to anotherMap using putAll() method

        anotherMap.putAll(map);  // code here

        //Printing key-value pairs of anotherMap

        entrySet = anotherMap.entrySet();

```

```
50          for (Entry<String, Integer> entry : entrySet)
51          {
52              System.out.println(entry.getKey()+" : "+entry.getValue());
```

	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 ►

Status	Finished
Started	Wednesday, 6 November 2024, 6:33 AM
Completed	Wednesday, 6 November 2024, 7:31 AM
Duration	58 mins 4 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 %)

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

public class Main{
    public static void main(String[] args){
        Scanner scanner =new
Scanner(System.in);
        int n=scanner.nextInt();
        ArrayList<Integer>arrayList=new
ArrayList<>();
         for (int i=0;i<n;i++){
arrayList.add(scanner.nextInt());
         }
         int first=arrayList.get(0);
         int last=
arrayList.get(arrayList.size()-1);
     System.out.println("ArrayList: " + arrayList);
     System.out.println("First : " + first + ", Last : "+ last);
    }
}
```

	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

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


public class Prog {

public static void main(String[] args)
{
 Scanner sc= new Scanner(System.in);
 int n = sc.nextInt();

 ArrayList<Integer> list = new ArrayList<>();

 for(int i = 0; i<n;i++)
 list.add(sc.nextInt());

 // printing initial value ArrayList
 System.out.println("ArrayList: " + list);
 list.set(1,100);


//Replacing the element at index 1 with 100


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

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

	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]	✓

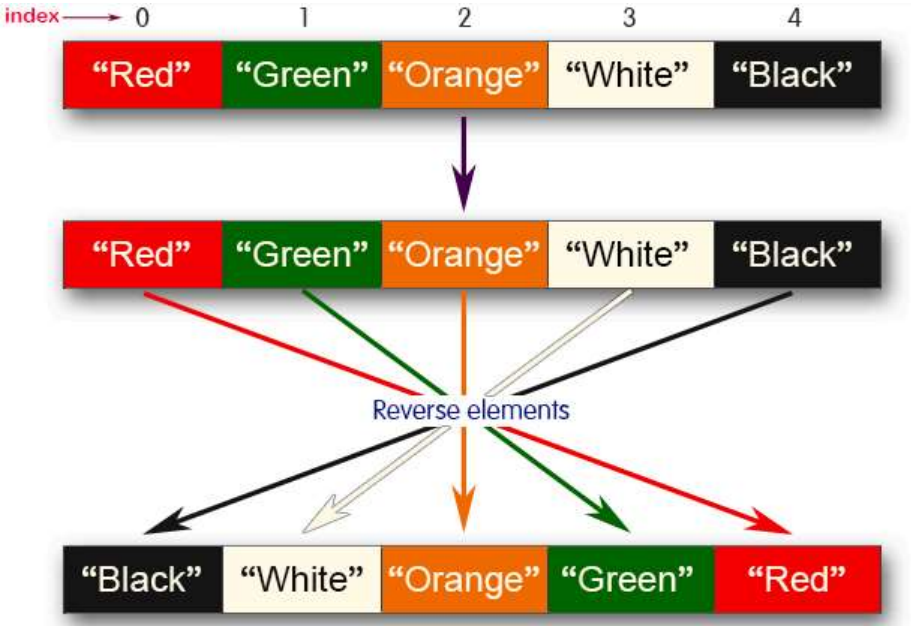
Passed all tests! ✓

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 %)

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
public class ReverseList{
    public static void main(String[] args){
        Scanner scanner = new
Scanner(System.in);
        int n=Integer.parseInt(scanner.nextLine().trim());
        List<String>list=new ArrayList<>();
        for(int i=0;i<n;i++){
            list.add(scanner.nextLine().trim());
        }
        System.out.println("List before reversing : ");
        System.out.println(list);
        Collections.reverse(list);
        System.out.println("List after reversing : ");
        System.out.println(list);
    }
}
```

	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 ►

Status	Finished
Started	Monday, 21 October 2024, 7:28 PM
Completed	Monday, 21 October 2024, 8:35 PM
Duration	1 hour 6 mins

Question **1**

Incorrect

Marked out of 5.00

Write a Java program to handle ArithmeticException and ArrayIndexOutOfBoundsException.

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

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

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

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

Input:

5

10 0 20 30 40

Output:

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

Input:

3
10 20 30

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

For example:

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

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;
public class ArithmeticAndArrayIndexOutOfBoundsExceptionHandling{
    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);
        try{
            int size=scanner.nextInt();
            int[]array=new int[size];
            for(int i=0;i<size;i++)
            {
                array[i]=scanner.nextInt();
            }
            int result=array[0]/array[1];
                 System.out.println("java.lang.ArithmeticException: / by zero");
        }catch(ArithmeticException e){
            System.out.println(e);
        }
        try{

            }
            catch(ArrayIndexOutOfBoundsException e){
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3
        }finally{
            System.out.println("I am always executed");
        }
    }
}
```

Syntax Error(s)

```
ArithmeticAndArrayIndexOutOfBoundsExceptionHandling.java:21: error: ')' expected
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3 " );
                                                                                  ^
ArithmeticAndArrayIndexOutOfBoundsExceptionHandling.java:21: error: ';' expected
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3 " );
                                                                                    ^
ArithmeticAndArrayIndexOutOfBoundsExceptionHandling.java:21: error: not a statement
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3 " );
                                                                                     ^
ArithmeticAndArrayIndexOutOfBoundsExceptionHandling.java:21: error: ';' expected
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3 " );
                                                                                          ^
ArithmeticAndArrayIndexOutOfBoundsExceptionHandling.java:21: error: '(' expected
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3 " );
                                                                                            ^
ArithmeticAndArrayIndexOutOfBoundsExceptionHandling.java:21: error: not a statement
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3 " );
                                                                                              ^
ArithmeticAndArrayIndexOutOfBoundsExceptionHandling.java:21: error: ';' expected
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3 " );
                                                                                                   ^
ArithmeticAndArrayIndexOutOfBoundsExceptionHandling.java:21: error: unclosed string literal
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: Index "  "" Out of bounds for length 3 " );
                                                                                                     ^
8 errors
```

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

```
class prog {
  public static void main(String[] args) {
    int n = 82;
    try{
         trynumber(n);
    }catch(Exception e){
        System.out.println(e.getMessage());
    }
    n = 37;
    try{
        trynumber(n);
    }catch(Exception e){
        System.out.println(e.getMessage());
    }
  }
    // call the trynumber(n);

  public static void trynumber(int n) throws Exception{
    try {
        //call the checkEvenNumber()
                checkEvenNumber(n);
      System.out.println(n + " is even.");
    } catch (Exception e) {
      System.out.println("" + e.getMessage());
    }
  }

  public static void checkEvenNumber(int number) throws Exception{
    if (number % 2 != 0) {
                              throw new Exception("Error: " + number + " is odd.");
    }
  }
}
```

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

Passed all tests! ✓

Question **3**

Correct

Marked out of 5.00

In the following program, an array of integer data is to be initialized.
During the initialization, if a user enters a value other than an integer, it will throw an InputMismatchException exception.
On the occurrence of such an exception, your program should print "You entered bad data."
If there is no such exception it will print the total sum of the array.

/* Define try-catch block to save user input in the array "name"
If there is an exception then catch the exception otherwise print the total sum of the array. */

Sample Input:

3
5 2 1

Sample Output:

8

Sample Input:

2

1 g

Sample Output:

```
You entered bad data.
```

For example:

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

Answer: (penalty regime: 0 %)

Reset answer

```
import java.util.Scanner;
import java.util.InputMismatchException;
 class prog {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int length = sc.nextInt();
    // create an array to save user input
    int[] name = new int[length];
    int sum=0;//save the total sum of the array.

    /* Define try-catch block to save user input in the array "name"
   If there is an exception then catch the exception otherwise print
   the total sum of the array. */
     try
      {

          for(int i=0;i<length;i++){
              name[i]=sc.nextInt();
              sum+=name[i];
          }
          System.out.println(sum);
         }catch(InputMismatchException e ){
             System.out.println("You entered bad data.");
      }
  }
}
```

```
27
```

	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 ►

Status	Finished
Started	Sunday, 6 October 2024, 2:56 PM
Completed	Sunday, 6 October 2024, 3:00 PM
Duration	4 mins 6 secs

Question **1**

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 %)

```
import java.util.Scanner;

public class VowelStringExtractor {

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

        // Iterate through the array of strings
        for (String s : stringArray) {
            // Check if the string is not empty and if both the first and last characters are vowels
```

```
            if (s.length() > 0 && vowels.indexOf(s.charAt(0)) != -1 && vowels.indexOf(s.charAt(s.length() - 1)) !=
                result.append(s); // Append matching string to the result
            }
        }

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

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

        // Input for the number of strings

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

        // Input for the strings in one line

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

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

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

	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! ✓

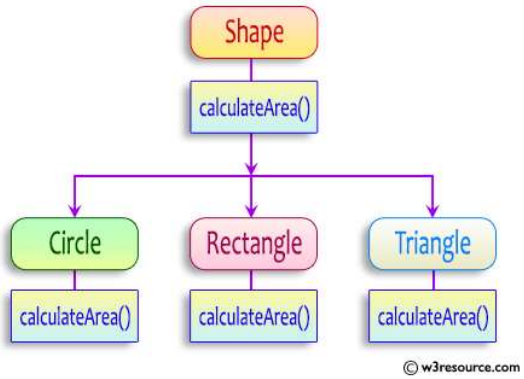
Question **2**

Correct

Marked out of 5.00

Create a base class Shape with a method called calculateArea(). Create three subclasses: Circle, Rectangle, and Triangle. Override the calculateArea() method in each subclass to calculate and return the shape's area.

In the given exercise, here is a simple diagram illustrating polymorphism implementation:




```
 abstract class Shape {
   public abstract double calculateArea() ;
   }
}
```

System.out.printf("Area of a Triangle :%.2f%n",((0.5)*base*height)); // use this statement

sample Input :

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

5 // length of the rectangle

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

4 // base of the triangle

3 // height of the triangle

OUTPUT:

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

For example:

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

Answer: (penalty regime: 0 %)

```
1 ▾ import java.util.Scanner;
2
3   // Abstract class Shape
4 ▾ abstract class Shape {
5       public abstract double calculateArea();
```

```
}

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

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

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

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

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

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

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

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

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

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

Passed all tests! ✓

Question **3**

Correct

Marked out of 5.00

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

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

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

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

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

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

```
25
26
```

	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! ✓

◄ Lab-08-MCQ

Jump to...

FindStringCode ►

Status	Finished
Started	Sunday, 6 October 2024, 10:45 AM
Completed	Sunday, 6 October 2024, 11:39 AM
Duration	54 mins 3 secs

Question **1**

Correct

Marked out of 5.00

RBI issues all national banks to collect interest on all customer loans.

Create an RBI interface with a variable String parentBank="RBI" and abstract method rateOfInterest().

RBI interface has two more methods default and static method.

default void policyNote() {

System.out.println("RBI has a new Policy issued in 2023.");

}

static void regulations(){

System.out.println("RBI has updated new regulations on 2024.");

}

Create two subclasses SBI and Karur which implements the RBI interface.

Provide the necessary code for the abstract method in two sub-classes.

Sample Input/Output:

RBI has a new Policy issued in 2023
RBI has updated new regulations in 2024.
SBI rate of interest: 7.6 per annum.
Karur rate of interest: 7.4 per annum.

For example:

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

Answer: (penalty regime: 0 %)

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

    // Abstract method
    double rateOfInterest();

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

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

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

// Karur class implementing RBI interface
class Karur implements RBI {
    // Implementing the abstract method
```

```
    public double rateOfInterest() {
        return 7.4;
    }
}

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

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

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

	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 **2**

Correct

Marked out of 5.00

create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

```
interface Playable {
    void play();
}

class Football implements Playable {
    String name;
    public Football(String name){
        this.name=name;
    }
  public void play() {
    System.out.println(name+" is Playing football");
   }
}
```

Similarly, create Volleyball and Basketball classes.

Sample output:

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

For example:

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

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;
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");
    }

}
class Volleyball implements Playable
{
    String name;
    public Volleyball(String name)
    {
        this.name=name;
    }
```

```
    public void play()
    {
        System.out.println(name+" is Playing volleyball");
    }
}
class Basketball implements Playable
{String name;
public Basketball(String name)
{
    this.name=name;
}
public void play()
{
    System.out.println(name+" is Playing basketball");
}

}
public class Main
{
    public static void main(String args[])
    {
        Scanner s= new Scanner(System.in);
        String name1=s.nextLine();
        String name2=s.nextLine();
        String name3=s.nextLine();
        Football s1=new Football(name1);
        Volleyball s2=new Volleyball(name2);
```

	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 **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

```
import java.util.Scanner;
 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);

}
class College implements Football {
    String homeTeam;
    String visitingTeam;

    public void setHomeTeam(String name){
        this.homeTeam=name;

   }
public void setVisitingTeam(String name){
  this.visitingTeam=name;
}
public void homeTeamScored(int points){
    System.out.println(homeTeam+" "+points+" scored");
}
public void visitingTeamScored(int points){
  System.out.println(visitingTeam+" "+points+" scored");
```

```
}
public void winningTeam(int p1, int p2){
    if(p1>p2)
     System.out.println(homeTeam+" is the winner!");
    else if(p1<p2)
     System.out.println(visitingTeam+" is the winner!");
    else
    System.out.println("It's a tie match.");
}
}
public class Main{
    public static void main(String[] args){
        String hname;
        Scanner sc= new Scanner(System.in);
       hname=sc.nextLine();
        String  vteam=sc.nextLine();
        int htpoints=sc.nextInt();
        int vtpoints=sc.nextInt();
   College s= new College();
    s.setHomeTeam(hname);
    s.setVisitingTeam(vteam);
    s.homeTeamScored(htpoints);
    s.visitingTeamScored(vtpoints);
    s.winningTeam(htpoints,vtpoints);

```

	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 ►

Status	Finished
Started	Sunday, 6 October 2024, 9:54 AM
Completed	Sunday, 6 October 2024, 10:42 AM
Duration	48 mins 18 secs

Question **1**

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){
```

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

```

	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! ✓

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 %)

```
import java.util.*;
class diff{
    char different(char a, char b){
        if ((int)a != (int)b)
            return (char)((int)'a' + ((int)a-(int)b) - 1);
        return a;
        }
}
public class Main{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        diff z = new diff();
        String q = scan.nextLine();
        StringBuffer ans = new StringBuffer();
        StringBuffer temp = new StringBuffer();
        for(int i = 0;i < q.length();i++){
            if(q.charAt(i) == ':'){
                temp.append(" ");
            }
            else{
                temp.append(Character.toString(q.charAt(i)));
            }
        }
        String h = temp.toString();
        for(int i = 0;i < temp.length();i++){
            if(i%3 == 0){
                ans.append(Character.toString(z.different(h.charAt(i),h.charAt(i+1))));
            }
        }
        System.out.print(ans.toString().toUpperCase());
    }
}
```

	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

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 %)

```
import java.util.*;


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

```
                ab.append(Character.toString(b.charAt(i)));
            }
        }
        char[] d = ab.toString().toCharArray();
        Arrays.sort(d);
        for(int i = d.length - 1;i >= 1;i--){
            if(d[i] != d[i-1])
            System.out.print(d[i]);
        }
        System.out.print(d[0]);
        }

    }
}

```

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

Passed all tests! ✓

◄ Lab-06-MCQ

Jump to...

Return second word in Uppercase ►

Status	Finished
Started	Saturday, 5 October 2024, 12:28 PM
Completed	Saturday, 5 October 2024, 12:38 PM
Duration	9 mins 58 secs

Question **1**

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 %)

```
class Mobile{
    public void basicMobile()
    {
        System.out.println("Basic Mobile is Manufactured");
    }
}
class CameraMobile extends Mobile{
    public void newFeature()
    {
        System.out.println("Camera Mobile is Manufactured");
    }
}
class AndroidMobile extends CameraMobile{
    public void androidMobile(){
        System.out.println("Android Mobile is Manufactured");
        System.out.println("Camera Mobile with 5MG px");
        System.out.println("Touch Screen Mobile is Manufactured");
    }
}

public class Main{
    public static void main(String[] args){
        AndroidMobile am = new AndroidMobile();
        am.basicMobile();
        am.newFeature();
        am.androidMobile();
    }
}
```

	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 **2**

Correct

Marked out of 5.00

create a class called College with attribute String name, constructor to initialize the name attribute , a method called Admitted(). Create a subclass called CSE that extends Student class, with department attribute , Course() method to sub class. Print the details of the Student.

College:

String collegeName;

public College() { }

public admitted() { }

Student:

String studentName;

String department;

public Student(String collegeName, String studentName,String depart) { }

public toString()

Expected Output:

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

For example:

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

Answer: (penalty regime: 0 %)

Reset answer

```
class College
{
protected   String collegeName;

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

    }

public void admitted() {
    System.out.println("A student admitted in "+collegeName);
}
}
class Student extends College{

String studentName;
String department;

public Student(String collegeName, String studentName,String depart) {
   // initialize the instance variables
   super(collegeName);
   this.studentName=studentName;
   this.department=depart;

}
```

```
28 public void details(){
29     System.out.println("CollegeName : "+collegeName);
30     System.out.println("StudentName : "+studentName);
31     System.out.println("Department : "+department);
32 }
33 }
34 public class Main {
35 public static void main (String[] args) {
36         Student  s1 = new Student("REC","Venkatesh","CSE");
37         s1.admitted();                                  // invoke the admitted() method
38         s1.details();
39 }
40 }
```

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

Passed all tests! ✓

Question **3**

Correct

Marked out of 5.00

Create a class 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

```
class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber, double balance){
        this.accountNumber=accountNumber;
        this.balance=balance;
    }

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

    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance");
        }
    }

    // Method to get the current balance
    public double getBalance() {
        // Return the current balance
        return balance;

    }
}

 class SavingsAccount extends BankAccount {
    // Constructor to initialize account number and balance
    public SavingsAccount(String accountNumber, double balance) {
        // Call the parent class constructor
        super(accountNumber,balance);

    }

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

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

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

Passed all tests! ✓

◄ Lab-05-MCQ

Jump to...

Is Palindrome Number? ►

Status	Finished
Started	Monday, 23 September 2024, 4:03 AM
Completed	Monday, 23 September 2024, 5:02 AM
Duration	59 mins 4 secs

Question **1**

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 %)

```
class prog{
    private String manufacturer;
    private String operatingSystem;
    private String color;
    private int cost;
    public prog(String manufacturer,String operatingSystem,String color,int cost){
        this.manufacturer=manufacturer;
        this.operatingSystem=operatingSystem;
        this.color=color;
        this.cost=cost;
    }
    public String getManufacturer(){
        return  manufacturer;
    }
    public void setManufacturer(String manufacturer){
        this.manufacturer=manufacturer;

    }
    public String getOperatingSystem(){
        return operatingSystem;
    }


    public void setOperatingSystem(String operatingSystem){
        this.operatingSystem=operatingSystem;
    }

    public String getcolor(){
        return color;
    }
    public void setColor(String color){
        this.color=color;
```

```
    }
    public void setcost(int cost){
        this.cost=cost;

    }

    @Override
    public String toString(){
        return "manufacturer = "+ manufacturer + "\n" +
        "operating_system = " + operatingSystem + "\n" +
        "color = " + color + "\n" +
        "cost = " + cost;
    }

    public static void main(String[] args) {
        prog mobile = new prog("Redmi", "Andriod", "Blue", 34000);
        System.out.println(mobile);
    }
}
```

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

Passed all tests! ✓

Question **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 %)

```
public class Student {
    private String name;
    private int rollNumber;

    public Student() {
        name = null;
        rollNumber = 0;
    }

    public Student(String name) {
        this.name = name;
        rollNumber = 0;
    }

    public Student(String name, int rollNumber) {
        this.name = name;
        this.rollNumber = rollNumber;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getRollNumber() {
        return rollNumber;
    }

```

```
    public void setRollNumber(int rollNumber) {
        this.rollNumber = rollNumber;
    }

    public static void main(String[] args) {
        Student student1 = new Student();
        System.out.println("No-arg constructor is invoked");
        Student student2 = new Student("Rajalakshmi");
        System.out.println("1 arg constructor is invoked");
        Student student3 = new Student("Lakshmi", 101);
        System.out.println("2 arg constructor is invoked");

        System.out.println("Name =" + student1.getName() + " , Roll no = " + student1.getRollNumber());

        System.out.println("Name =" + student2.getName() + " , Roll no = " + student2.getRollNumber());

        System.out.println("Name =" + student3.getName() + " , Roll no = " + student3.getRollNumber());
    }
}
```

	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 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

```
import java.io.*;
import java.util.Scanner;

class Circle
{
    private double radius;
    public Circle(double radius){
        // set the instance variable radius
        this.radius = radius;


    }
    public void setRadius(double radius){
        // set the radius
        this.radius = radius;


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

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

    }
    public double calculateCircumference()    {
        // complete the statement
       return  2 * Math.PI * radius;
    }
}
class prog{
    public static void main(String[] args)  {
        int r;
        Scanner sc= new Scanner(System.in);
        r=sc.nextInt();
        Circle c= new Circle(r);
        System.out.println("Area = "+String.format("%.2f", c.calculateArea()));
```

```
40          System.out.println( Circumference =  +String.format( %.2f , c.calculateCircumference()));
41
42
43      }
44  }
45
```

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

Passed all tests! ✓

◄ Lab-04-MCQ

Jump to...

Number of Primes in a specified range ►

Status	Finished
Started	Sunday, 22 September 2024, 11:30 PM
Completed	Monday, 23 September 2024, 12:18 AM
Duration	48 mins 13 secs

Question **1**

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 x 9), (-4 x 9), (3 x 9), (0 x 9)} = {-72, -36, -27, 0}

So, the expected output is the resultant array {-72, -36, -27, 0}.

Example 2:

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

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

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

Explanation:

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

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

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

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

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

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

Example 3:

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

input2 = {-9, 9}

Expected Output = {-162, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

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

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

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

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

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

For example:

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

Answer: (penalty regime: 0 %)

```
import java.util.Arrays;
import java.util.Scanner;
import java.util.stream.Collectors;
public class Test{
    public static int[] transformArray(int[] inputArray) {
        int maxNum = Arrays.stream(inputArray).max().getAsInt();

        for (int i = 0; i < inputArray.length; i++) {
            inputArray[i] = inputArray[i] - maxNum;
        }

        for (int i =0; i < inputArray.length; i++){
            inputArray[i] = inputArray[i] * maxNum;
        }

        return inputArray;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] inputArray = new int[n];
        for (int i=0; i<n; i++){
            inputArray[i] = scanner.nextInt();
        }
        int[] result = transformArray(inputArray);

        /*for (int i=0; i<result.length; i++){
            System.out.println(" "+ result[i] + " ");
        }
        scanner.close();*/

        System.out.println(Arrays.stream(result).mapToObj(String::valueOf).collect(Collectors.joining(" ")));
    }

}
```

	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 **2**

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
```

```java
public class LongestPositiveSequenceSum {
    public static int findLongestPositiveSequenceSum(int[] arr) {
        int maxSum = -1;

        int currentSum = 0;

        int maxLength = 0;

        int currentLength = 0;

        for(int i=0; i<arr.length; i++) {
            if (arr[i] >=0) {
                currentSum += arr[i];
                currentLength++;
            } else {
                if (currentLength > maxLength) {
                    maxSum = currentSum;
                    maxLength = currentLength;
                } else if (currentLength == maxLength) {
                    maxSum += currentSum;
                }
                currentSum = 0;
                currentLength = 0;
            }
        }

        if(currentLength > maxLength) {
            maxSum = currentSum;
        } else if (currentLength == maxLength) {
            maxSum += currentSum;
        }
        return maxSum == -1 ? -1 : maxSum;
    }

    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for(int i=0; i<n; i++){
            arr[i] = scanner.nextInt();
        }

        int result = findLongestPositiveSequenceSum(arr);
        System.out.println(result);
        scanner.close();
    }
}
```

	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! ✓

Question **3**

Correct

Marked out of 5.00

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

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

This is explained below:

Example 1:

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

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

Step 1:

Starting from the 0^{th} index of the array pick up digits as per below:

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

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

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

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

4^{th} 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 %)

```java
import java.util.Scanner;

public class sumOfSpecificDigits {
    public static int extractDigit(int number, int position) {

        String numStr = Integer.toString(number);

        if (numStr.length() <= position) {
            return 0;
        }

        return Character.getNumericValue(numStr.charAt(numStr.length()-1-position));
    }

    public static int calculateSumOfSquares(int[] numbers) {
        int sum = 0;

        for (int i=0; i < numbers.length; i++) {
            int extractedDigit = extractDigit(numbers[i], i);
            sum += extractedDigit * extractedDigit;
        }
        return sum;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] numbers = new int[n];
        for(int i=0; i<n; i++) {
            numbers[i] = scanner.nextInt();
        }

        int result = calculateSumOfSquares(numbers);

        System.out.println(result);

        scanner.close();
    }
}
```

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

Passed all tests! ✓

◄ Lab-03-MCQ

Jump to...

Simple Encoded Array ►

Status	Finished
Started	Sunday, 22 September 2024, 10:15 PM
Completed	Sunday, 22 September 2024, 11:29 PM
Duration	1 hour 13 mins

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 %)

```
import java.util.Scanner;

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

        int day = scanner.nextInt();
        boolean isOnVacation = scanner.nextBoolean();

        int wakeUpTime;

        if (isOnVacation) {
            if (day>1 && day<=5) {
                wakeUpTime = 7;
            } else {
```

```
                wakeUpTime = 9;
            }
        } else {
            if (day>1 && day<=5){
                wakeUpTime = 5;
            } else {
                wakeUpTime = 6;
            }
        }
        System.out.println(wakeUpTime + ":00");
    }
}
```

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

Logic to print number in words in Java programming.

Example

Input

1234

Output

One Two Three Four

Input:

16

Output:

one six

For example:

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

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;

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

        String number = scanner.nextLine();

        String[] units = {"Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"};

        for (int i=0; i< number.length(); i++){
            char ch = number.charAt(i);
            int digit  = Character.getNumericValue(ch);
            System.out.print(units[digit] + " ");
        }
    }
}
```

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

Passed all tests! ✓

Question **3**

Correct

Marked out of 5.00

Consider the following sequence:

1st term: 1

2nd term: 1 2 1

3rd term: 1 2 1 3 1 2 1

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

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

Example Input:

1

Output:

1

Example Input:

4

Output:

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

For example:

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

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;

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

        int n = scanner.nextInt();

        String term = "1";

        for(int i=2; i<=n; i++){
            term += " "+ i + " " +term;
        }
        System.out.println(term);

    }
}
```

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

Passed all tests! ✓

◄ Lab-02-MCQ

Jump to...

Lab-03-MCQ ►

Status	Finished
Started	Sunday, 22 September 2024, 3:52 PM
Completed	Sunday, 22 September 2024, 4:11 PM
Duration	19 mins 22 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 %)

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

	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 %)

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

	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 %)

```
import java.util.Scanner;
public class prog
{
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        int n1=s.nextInt();
        int n2=s.nextInt();
        System.out.println(Math.abs(n1%10)+Math.abs(n2%10));
        }
}
```

	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? ►

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 bonafide work of **"KRITHIKA .B, KOMMANA SAI LEELA YUKTHA"** who carried out theproject 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

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

3.2Dashboard

```
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_imort_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 minize_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
289         } catch (Exception e) {e.printStackTrace();}
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
    TIMESTAMP(date) ASC LIMIT 6";
321
322         String sql = "SELECT date, SUM(total) FROM customer_receipt GROUP BY date ORDER BY
    TIMESTAMP(date) ASC LIMIT 6;";
323
324         conn = databaseHandler.connectDb();
325
326         try {
327
328             XYChart.Series chart = new XYChart.Series();
329
330             prepare = conn.prepareStatement(sql);
331             resultSet = prepare.executeQuery();
332
333             while(resultSet.next()){
334                 chart.getData().add(new XYChart.Data(resultSet.getString(1), resultSet.getInt(2)));
335             }
336
337             home_incomeChart.getData().add(chart);
338
339         } catch (Exception e) {e.printStackTrace();}
340
341     }
342
343     public void homeOrdersChart(){
344
345         home_orderChart.getData().clear();
346
347         String sql = "SELECT date, COUNT(id) FROM customer_receipt GROUP BY date ORDER BY
    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.
    The SUM(total) adds up all income values for each date, so we get the total income on that day.
351         FROM customer_receipt: This tells the database to look in the customer_receipt table.
352         GROUP BY date: It groups the data by date, so it only shows one total amount per date.
353         ORDER BY TIMESTAMP(date) ASC: Orders the data by date from oldest to newest.
354         LIMIT 6: Only shows the last 6 days (or dates) of data.
355         */
356
357         conn = databaseHandler.connectDb();
358
359         try {
360
361             XYChart.Series chart = new XYChart.Series<>(); // This creates an empty series, which is
    like a container that will hold all the data points for the chart.
362
363             prepare = conn.prepareStatement(sql);
364             resultSet = prepare.executeQuery();
365
366             while(resultSet.next()){
367                 chart.getData().add(new XYChart.Data(resultSet.getString(1), resultSet.getInt(2)));
368
369                 /*
370                 resultSet.next(): Moves to the next row of data.
371                 resultSet.getString(1): Gets the date from the first column.
372                 resultSet.getInt(2): Gets the total income for that date from the second column.
373                 */
374
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,
    date) "
386                 + "VALUES(?,?,?,?,?,?,?,?)";
387
388         conn = databaseHandler.connectDb();
389
390         try {
391
392             Alert alert;
393
394             if (addProduct_productId_txt.getText().isEmpty()
395                     || addProduct_productType_combo.getSelectionModel().getSelectedItem() == null
396                     || addProduct_brand_txt.getText().isEmpty()
397                     || addProduct_productName_txt.getText().isEmpty()
398                     || addProduct_price_txt.getText().isEmpty()
399                     || addProduct_status_combo.getSelectionModel().getSelectedItem() == null
400                     || GetData.path == "") {
401
402                 alert = new Alert(Alert.AlertType.ERROR);
403                 alert.setTitle("Error Message");
404                 alert.setHeaderText(null);
405                 alert.setContentText("Please fill all blank fields");
406                 alert.showAndWait();
407
```

```
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 
464         } catch (Exception e) {
```

```
            e.printStackTrace();
        }

    }

    private String[] listType = {"Snacks", "Drinks", "Dessert", "Gadgets", "Personal Product",
"Others"};

    public void addProductsListTypes() {

        List<String> listT = new ArrayList<>();

        for (String data : listType) {
            listT.add(data);
        }

        ObservableList listData = FXCollections.observableArrayList(listT);
        addProduct_productType_combo.setItems(listData);

    }

    private String[] listStatus = {"Available", "Not Available"};

    public void addProductsListStatus() {
        List<String> listS = new ArrayList<>();

        for (String data : listStatus) {
            listS.add(data);
        }

        ObservableList listData = FXCollections.observableArrayList(listS);
        addProduct_status_combo.setItems(listData);
    }

    public void addproductsSelectedItemsToInputFields() {

        ProductsModel productsData = addProduct_tableView.getSelectionModel().getSelectedItem(); //
TableView<ProductsModel> addProduct_tableView; nisa return wenneth ProductsModel object ekak
        int num = addProduct_tableView.getSelectionModel().getSelectedIndex();

        if ((num - 1) < -1) {
            return;
        } // 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.

        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
//        addProduct_productType_combo.getSelectionModel().clearSelection();
        addProduct_brand_txt.setText(productsData.getBrand());
        addProduct_productName_txt.setText(productsData.getProductName());
        addProduct_price_txt.setText(String.valueOf(productsData.getPrice()));
//        addProduct_status_combo.getSelectionModel().clearSelection();

        String uri = "file:" + productsData.getImage();

        image = new Image(uri, 115, 128, false, true);
        addProduct_imgView.setImage(image);

        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
```

```
    the path is empty, by clicking a row in the table, the path of the image can be set to the path of
    GetData.java.
520     }
521 
522     public void addProductsUpdate() {
523 
524         String uri = GetData.path;
525         uri = uri.replace("\\", "\\\\");
526 
527         Date date = new Date();
528         java.sql.Date sqlDate = new java.sql.Date(date.getTime());
529 
530         String sql = "UPDATE product SET type = '"
531                 + addProduct_productType_combo.getSelectionModel().getSelectedItem()
532                 + "', brand = '" + addProduct_brand_txt.getText()
533                 + "', product_name = '" + addProduct_productName_txt.getText()
534                 + "', price = '" + addProduct_price_txt.getText()
535                 + "', status = '" + addProduct_status_combo.getSelectionModel().getSelectedItem()
536                 + "', image = '" + uri
537                 + "', date = '" + sqlDate
538                 + "' WHERE product_id = '" + addProduct_productId_txt.getText() + "'";
539 
540         conn = databaseHandler.connectDb();
541 
542         try {
543 
544             Alert alert;
545 
546             if (addProduct_productId_txt.getText().isEmpty()
547                     || addProduct_productType_combo.getSelectionModel().getSelectedItem() == null
548                     || addProduct_brand_txt.getText().isEmpty()
549                     || addProduct_productName_txt.getText().isEmpty()
550                     || addProduct_price_txt.getText().isEmpty()
551                     || addProduct_status_combo.getSelectionModel().getSelectedItem() == null
552                     || GetData.path == "") {
553 
554                 alert = new Alert(Alert.AlertType.ERROR);
555                 alert.setTitle("Error Message");
556                 alert.setHeaderText(null);
557                 alert.setContentText("Please fill all blank fields");
558                 alert.showAndWait();
559 
560             } else {
561 
562                 alert = new Alert(Alert.AlertType.CONFIRMATION);
563                 alert.setTitle("Confirmation Message");
564                 alert.setHeaderText(null);
565                 alert.setContentText("Are you sure you want to UPDATE Product ID: " +
    addProduct_productId_txt.getText() + " ?");
566 
567                 Optional<ButtonType> option = alert.showAndWait();
568 
569                 if (option.get().equals(ButtonType.OK)) {
570 
571                     statement = conn.createStatement();
572                     statement.executeUpdate(sql);
573 
574                     alert = new Alert(Alert.AlertType.INFORMATION);
575                     alert.setTitle("Information Message");
576                     alert.setHeaderText(null);
577                     alert.setContentText("Successfully Updated");
578                     alert.showAndWait();
579 
```

```
                    // TO BECOME UPDATED YOUR TABLEVIEW
                    addProductsShowListData();

                    // CLEAR THE FIELDS
                    addProductsReset();
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void addProductsDelete() {

        String sql = "DELETE FROM product WHERE product_id = '" + addProduct_productId_txt.getText() + "'";
        conn = databaseHandler.connectDb();

        try {

            Alert alert;

            if (addProduct_productId_txt.getText().isEmpty()
                    || addProduct_productType_combo.getSelectionModel().getSelectedItem() == null
                    || addProduct_brand_txt.getText().isEmpty()
                    || addProduct_productName_txt.getText().isEmpty()
                    || addProduct_price_txt.getText().isEmpty()
                    || addProduct_status_combo.getSelectionModel().getSelectedItem() == null
                    || GetData.path == "") {

                alert = new Alert(Alert.AlertType.ERROR);
                alert.setTitle("Error Message");
                alert.setHeaderText(null);
                alert.setContentText("Please fill all blank fields");
                alert.showAndWait();

            } else {

                alert = new Alert(Alert.AlertType.CONFIRMATION);
                alert.setTitle("Confirmation Message");
                alert.setHeaderText(null);
                alert.setContentText("Are you sure you want to DELETE Product ID: " + addProduct_productId_txt.getText() + " ?");

                Optional<ButtonType> option = alert.showAndWait();

                if (option.get().equals(ButtonType.OK)) {

                    statement = conn.createStatement();
                    statement.executeUpdate(sql);

                    alert = new Alert(Alert.AlertType.INFORMATION);
                    alert.setTitle("Information Message");
                    alert.setHeaderText(null);
                    alert.setContentText("Successfully Deleted");
                    alert.showAndWait();

                    // TO BECOME UPDATED YOUR TABLEVIEW
                    addProductsShowListData();

                    // CLEAR THE FIELDS
                    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         This is helpful to ensure that the user completes the current action (choosing an image) before
    interacting with the rest of the application.
677 
678         ---Why Specify the Parent Window?--
679         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:
680 
681         Blocking interaction with the main window:
682 
683         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.
684         Dialog behavior and position:
685 
686         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.
687         Maintaining application flow:
688 
```

```
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
750             prepare = conn.prepareStatement(sql);
751             resultSet = prepare.executeQuery();
752             ProductsModel proData;
753
754             while (resultSet.next()) {
755
756                 proData = new ProductsModel(
757                         resultSet.getInt("product_id"),
758                         resultSet.getString("type"),
759                         resultSet.getString("brand"),
760                         resultSet.getString("product_name"),
761                         resultSet.getDouble("price"),
762                         resultSet.getString("status"),
763                         resultSet.getString("image"),
764                         resultSet.getDate("date")
765                 );
766
767                 productList.add(proData);
768             }
769
770         } catch (Exception e) {
771             e.printStackTrace();
772         }
773
774         return productList;
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 colomns 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,
    date) "
811                 + "VALUES(?, ?, ?, ?, ?, ?, ?)";
812 
813         conn = databaseHandler.connectDb();
814 
815         try {
816 
817             String checkData = "SELECT * FROM product WHERE product_name = '"
818                     + orders_productName_combo.getSelectionModel().getSelectedItem() + "'";
819 
820             double priceData = 0;
821 
822             statement = conn.createStatement();
823             resultSet = statement.executeQuery(checkData);
824 
825             if (resultSet.next()) {
826                 priceData = resultSet.getDouble("price"); // if there are more prices more than 1 then
    what happen?????
827 //                ---------------
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 
934         } catch (Exception e) {
935             e.printStackTrace();
936         }
937 
938     }
939 
940     public void ordersReceipt() throws JRException{
941 
942         try {
943 
944 //            customerId();
945 
946             HashMap hash = new HashMap();
947             hash.put("inventoryP", customer_Id);
948 
949             if(totalP == 0){
950 
951                 Alert alert = new Alert(Alert.AlertType.ERROR);
952                 alert.setTitle("Error Message");
953                 alert.setHeaderText(null);
954                 alert.setContentText("Invalid :3"); // kalin order eka pay karala iwara nam den totalP = 0 ema nisa receipt ekak print kranna denne aye order ekak daddi
955                 alert.showAndWait();
956 
957             }else{
958 
959                 JasperDesign jDesign = JRXmlLoader.load("C:\\Users\\krith\\Downloads\\YT_InventoryManagementSystem_JavaFx-main\\YT_InventoryManagementSystem_JavaFx-main\\report.jrxml");
960                 JasperReport jReport = JasperCompileManager.compileReport(jDesign);
961                 JasperPrint jPrint = JasperFillManager.fillReport(jReport, hash, conn);
962 
963                 JasperViewer.viewReport(jPrint, false);
964 
965 
966             }
967 
968         } catch (Exception e) {e.printStackTrace();}
969 
970     }
971 
972     public void ordersReset(){
973 
974         customerId();
975 
976         String sql = "DELETE FROM customer WHERE customer_id = '"+ customer_Id +"'";
977 
978         conn = databaseHandler.connectDb();
979 
980         try {
981             if(!orders_tableView.getItems().isEmpty()){
982 
983                 Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
984                 alert.setTitle("Confirmation Message");
985                 alert.setHeaderText(null);
986                 alert.setContentText("Are you sure you want to reset?");
987                 Optional<ButtonType> option = alert.showAndWait();
```

```
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
1052        private double totalP;
1053
1054        public void ordersDisplayTotal() {
1055
1056            customerId();
1057
1058            String sql = "SELECT SUM(price) FROM customer WHERE customer_id = '" + customer_Id + "'
    ";   // mehidee cus id of customer table is 1, and cus id of customer_receipt is 0, customerId() ekedi
    customer table eke customer id eka select krala gannwa, eka thamay methana thiyenne '"+customer_Id+"'
    lesa.
1059
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
    Product", "Others"};
1080
1081        public void OrdersListTypes() {
1082
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
    brand name combo we want call again this mehtod, because we want to show product name combo list
1097
1098            // Clear the product name combo box when a new type is selected
1099            orders_productName_combo.getSelectionModel().clearSelection();
1100            orders_productName_combo.getItems().clear();
1101
1102            // Check if brand selection exists before executing the query
1103            if (orders_productType_combo.getSelectionModel().getSelectedItem() == null) {
1104                return; // Exit the method if no brand is selected
1105            }
1106
1107            String sql = "SELECT * FROM product WHERE type = '"
1108                    + 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
    select a number between 0 and 20. Here's a detailed explanation of each part of this method:
1187
1188            1. Spinner Initialization:
1189
1190
1191            spinner = new SpinnerValueFactory.IntegerSpinnerValueFactory(0, 20, 0);
1192
1193            SpinnerValueFactory.IntegerSpinnerValueFactory(0, 20, 0) creates an integer-based spinner
    with a minimum value of 0, a maximum value of 20, and an initial/default value of 0.
1194            The SpinnerValueFactory helps manage the range and value of the spinner for quantity
    selection.
1195
1196
1197            2. Assigning the Spinner to the orders_quantity_spinner:
1198
1199            orders_quantity_spinner.setValueFactory(spinner);
1200
1201            Here, orders_quantity_spinner is a spinner UI element (likely created in the FXML file or
    elsewhere in your JavaFX UI code).
1202            This line assigns the spinner factory as the value factory for orders_quantity_spinner.
    This means that the spinner's value range (0 to 20) and starting value (0) are now applied to
    orders_quantity_spinner.
1203
1204
1205            3. Placement in Initialization or Button Click:
1206
1207            ordersSpinner() is called in the initialize() method (often used in JavaFX controllers to
    initialize UI elements when the scene is loaded) or on a button click.
1208            When called in initialize(), it sets up the spinner with the specified range and default
    value as soon as the interface is loaded, making it ready for user interaction.
1209
1210             */
1211        }
1212
1213        private int qty;
1214
1215        public void ordersShowSpinnerValue() {
1216            qty = orders_quantity_spinner.getValue();
1217        }
1218
1219        public ObservableList<CustomerData> ordersListData() {
1220            customerId(); // pay button eke explain ekata meka yodaganna
1221            ObservableList<CustomerData> listData = FXCollections.observableArrayList();
1222            String sql = "SELECT * FROM customer WHERE customer_id = '" + customer_Id + "' ";
1223
1224            conn = databaseHandler.connectDb();
1225
1226            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
    and increments customer_Id by 1.
1310                else if(checkId == customer_Id) checks if the last customer_id from customer_receipt
    matches customer_Id. If they are the same, it increments customer_Id by 1 to avoid duplication.
1311
1312                 */
1313            } catch (Exception e) {
1314                e.printStackTrace();
1315            }
1316        }
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,
    #969635);");
1327                addProducts_btn.setStyle("-fx-background-color:transparent;");
1328                orders_btn.setStyle("-fx-background-color:transparent;");
1329
1330                homeDisplayTotalOrders();
1331                homeTotalIncome();
1332                homeAvailableProducts();
1333                homeIncomeChart();
1334                homeOrdersChart();
1335
1336            } else if (event.getSource() == addProducts_btn) {
1337
1338                home_form.setVisible(false);
1339                addProducts_form.setVisible(true);
1340                orders_form.setVisible(false);
1341
1342                home_btn.setStyle("-fx-background-color:transparent;");
1343                addProducts_btn.setStyle("-fx-background-color:linear-gradient(to bottom right,
    #269e70, #969635);");
1344                orders_btn.setStyle("-fx-background-color:transparent;");
1345
1346                addProductsShowListData();
1347                addProductsListTypes();
1348                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
1410                    root.setOnMouseReleased((MouseEvent event) -> {
1411                        stage.setOpacity(1);
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 displayUsername(){
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            displayUsername();
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;
    }
```

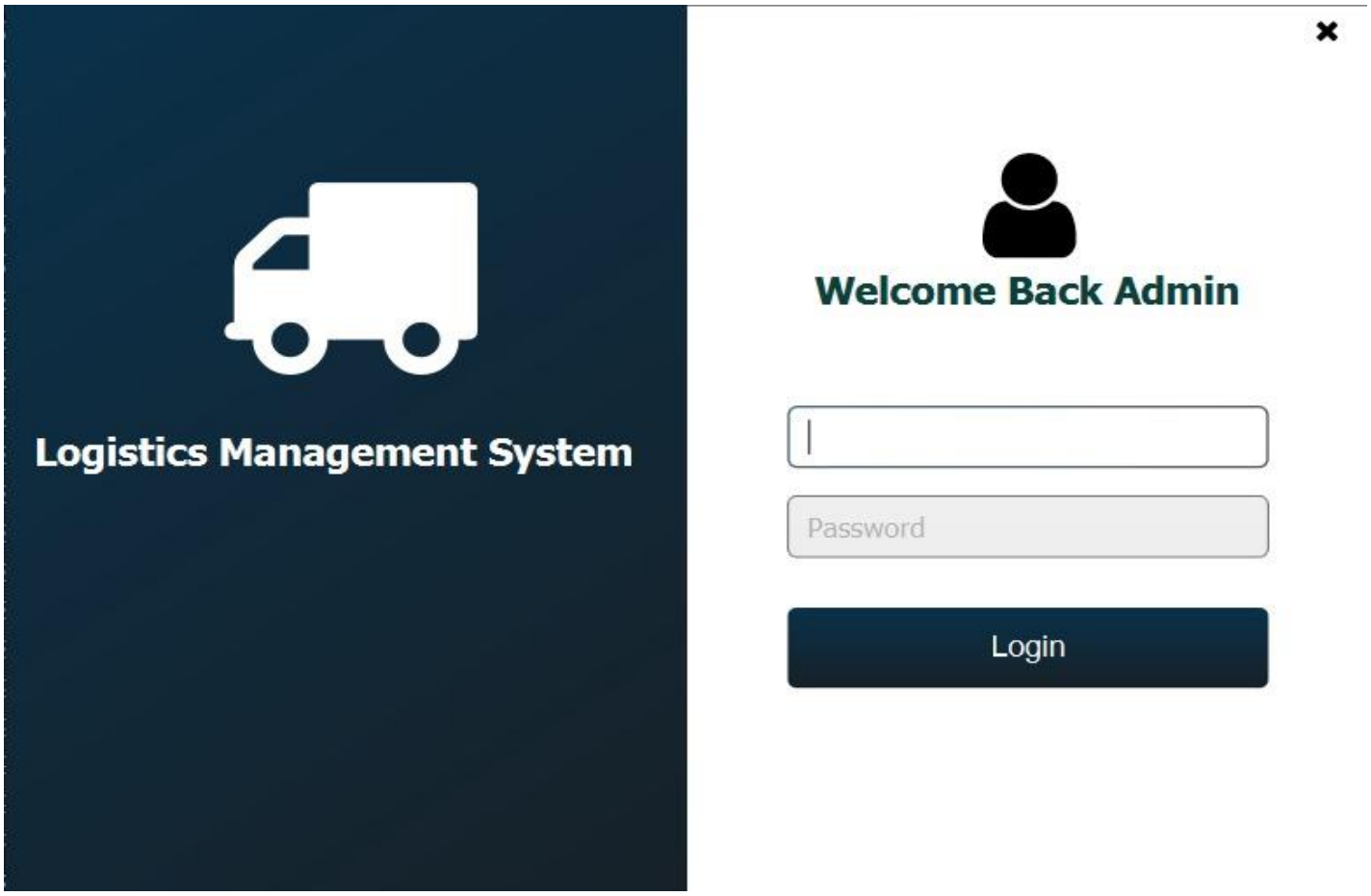
```
    public Double getPrice() {
        return price;
    }

    public String getStatus() {
        return status;
    }

    public String getImage() {
        return image;
    }

    public Date getDate() {
        return date;
    }


}
```

3.7 Module info :

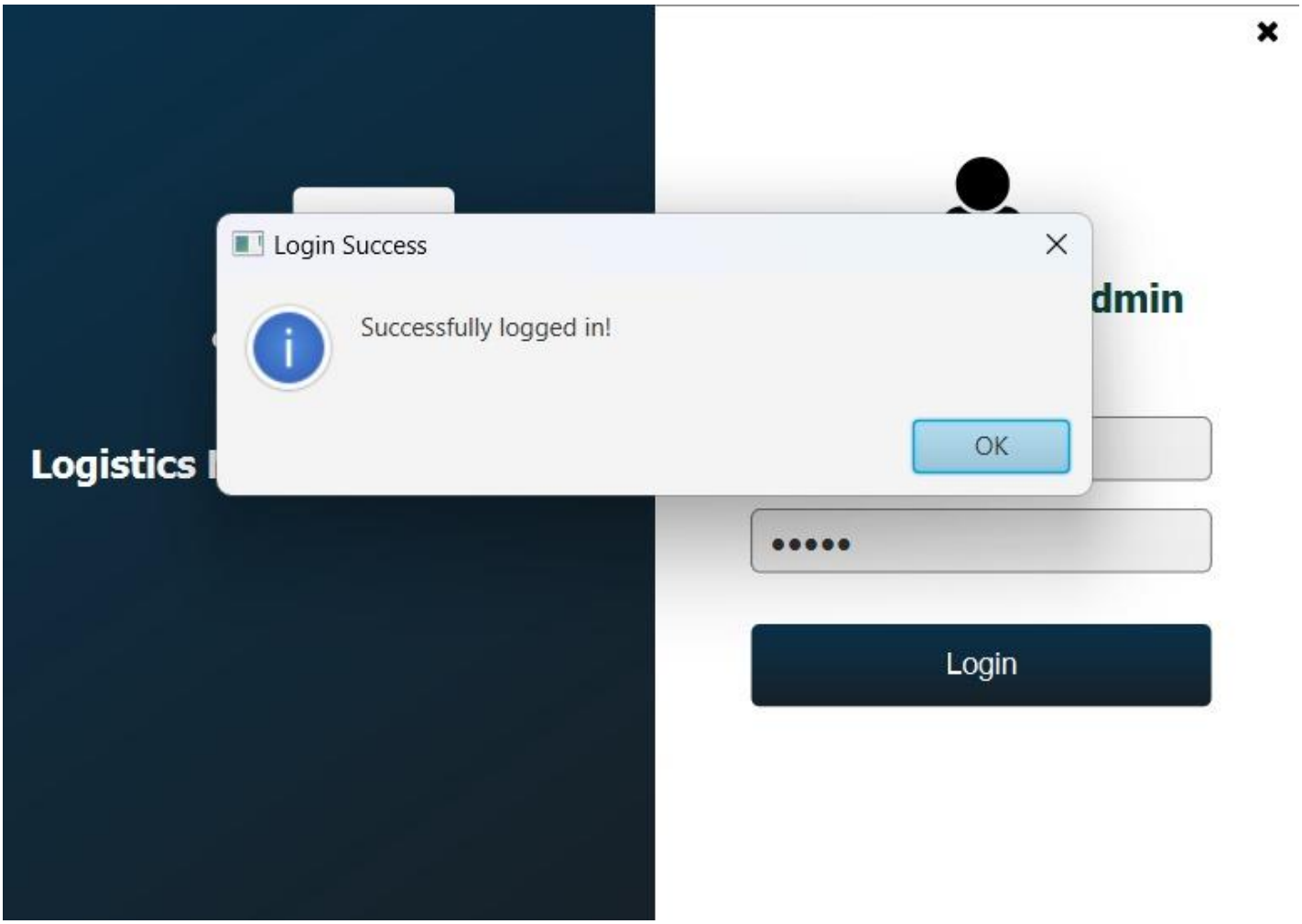
```
module com.mycompany.yt_inventorymanagementsystem_javafx {
    requires javafx.controls;
    requires javafx.fxml;
    requires java.sql;
    requires de.jensd.fx.glyphs.fontawesome;
    requires java.base;
    requires jasperreports;

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

SNAPSHOTS:

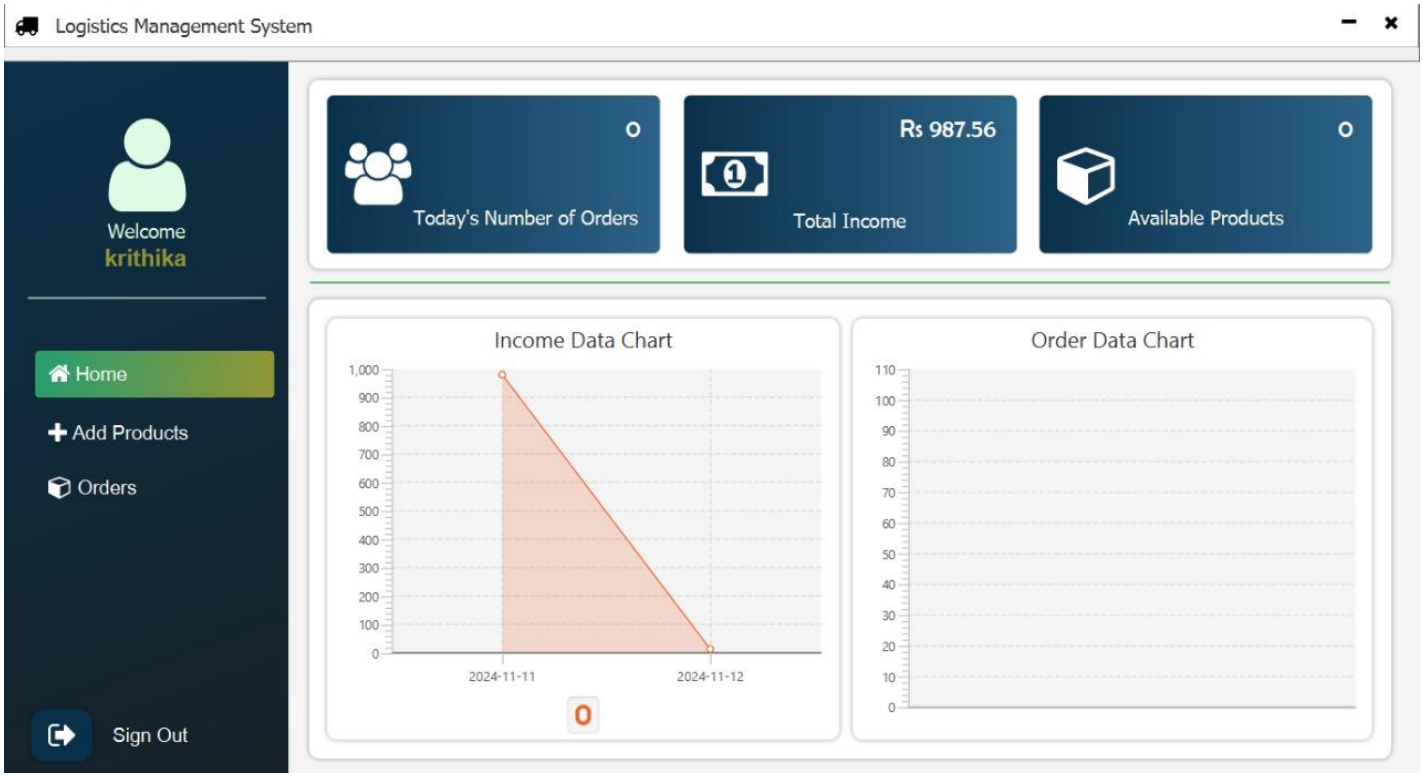
4.1 login page



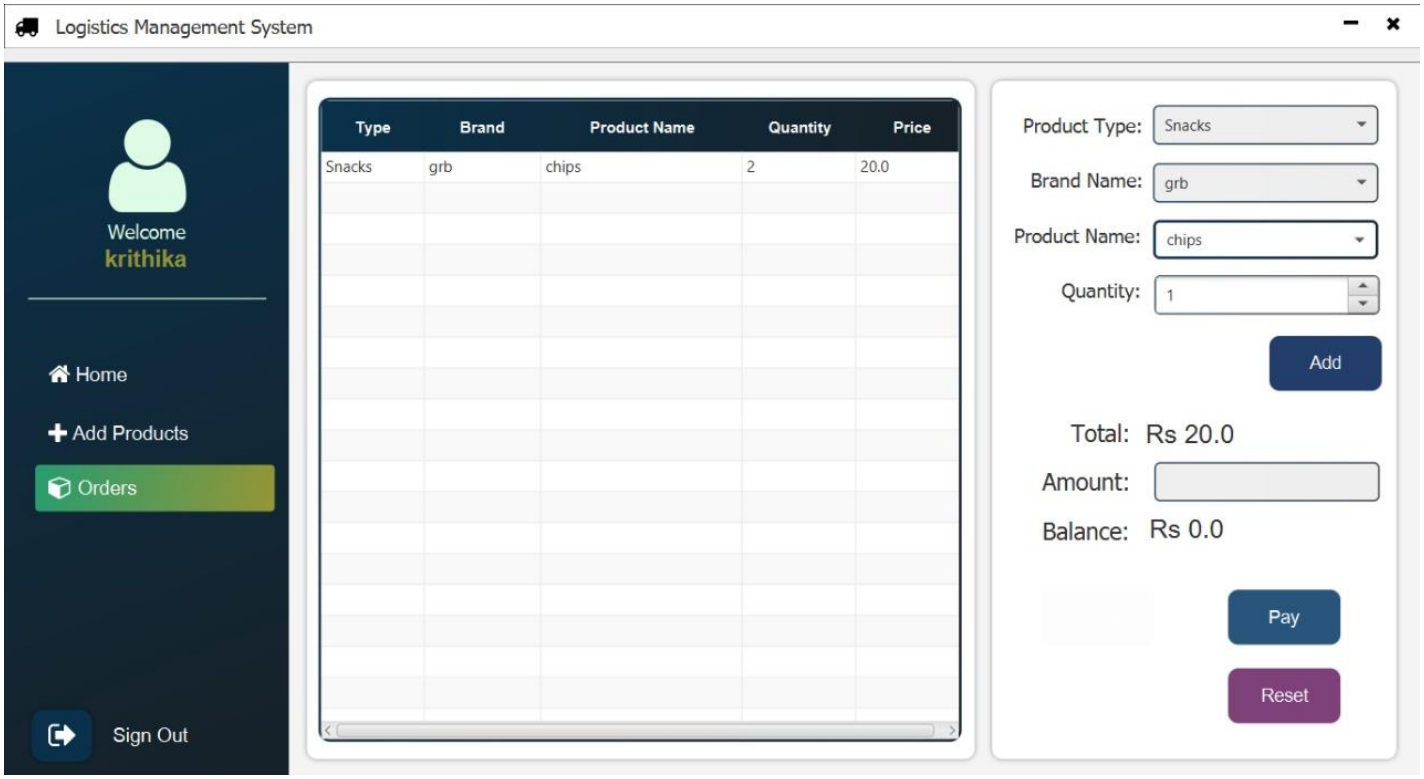
4.2 login successfull


Login Success
Successfully logged in!
OK
dmin
Logistics
Login

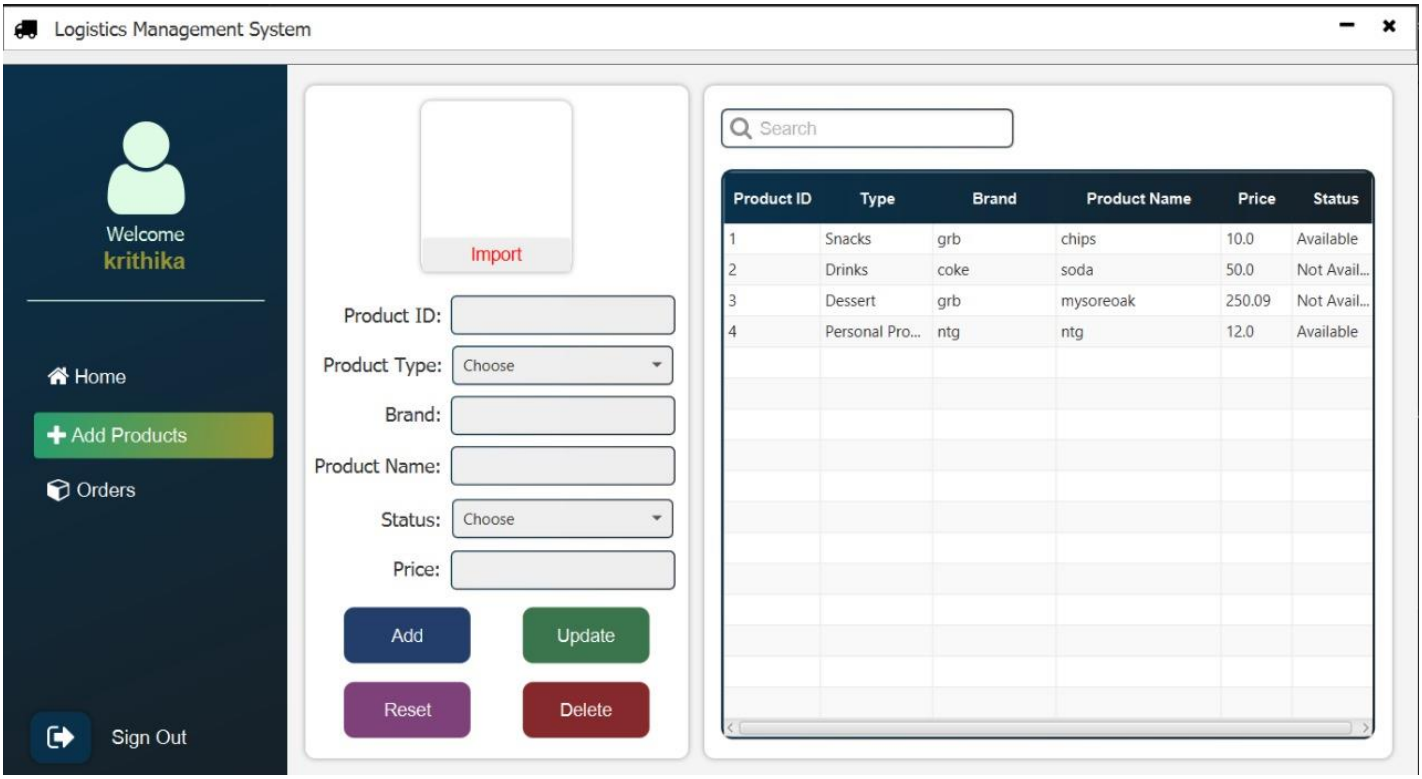
4.3 home page


Logistics Management System
Welcome
krithika
Home
Add Products
Orders
Sign Out
0
Today's Number of Orders
Rs 987.56
Total Income
0
Available Products
Income Data Chart
1,000
900
800
700
600
500
400
300
200
100
0
2024-11-11
2024-11-12
0
Order Data Chart
110
100
90
80
70
60
50
40
30
20
10
0

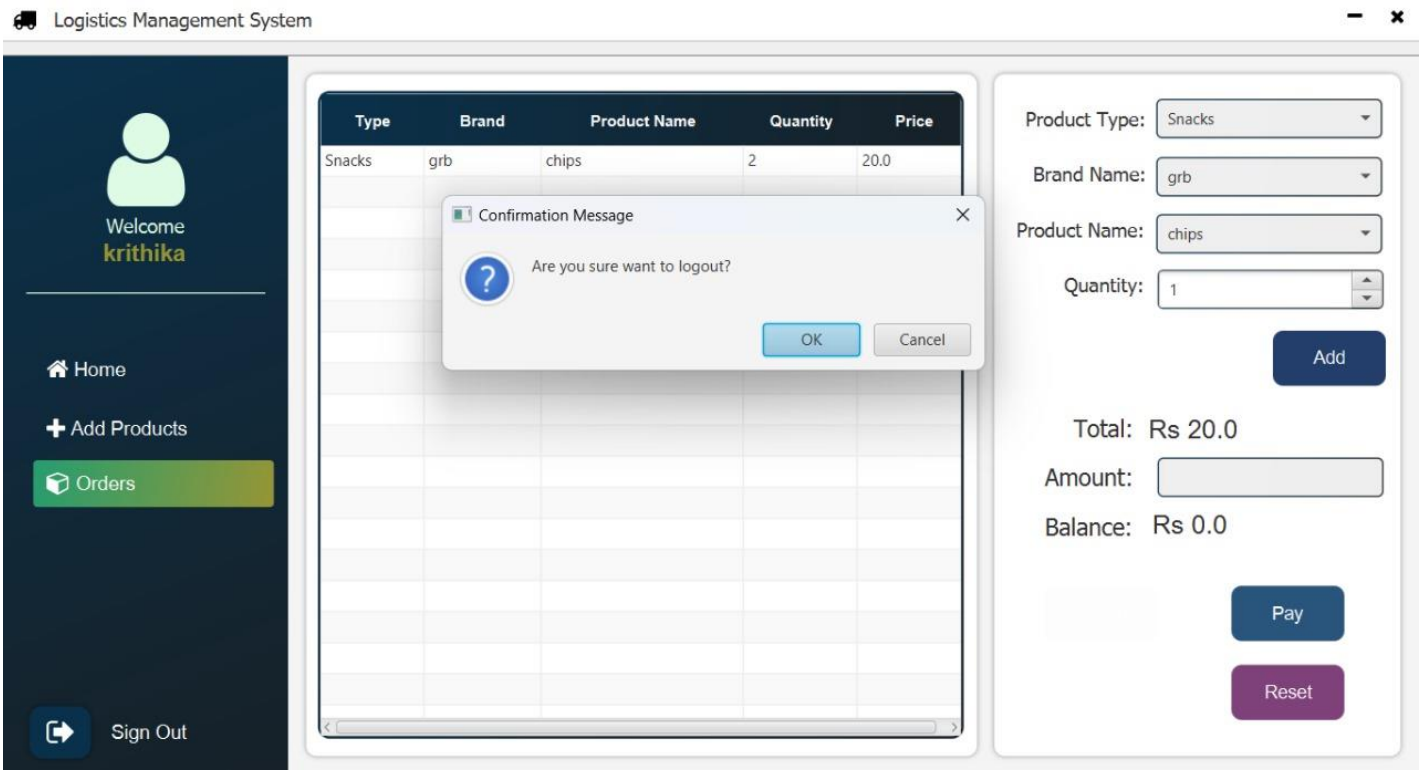
4.4 entering orders



4.5 adding products



4.6 logout successful



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