## 

## 

## 

## 

## 

## 

## LAB 06

## STRING, STRING BUFFER

EXERCISE 1

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:

import java.util.\*;

public class StringMergeSort {

public static String mergeAndSort(String input1, String input2) {

// Step 1: Concatenate both strings

String concatenated = input1 + input2;

// Step 2: Remove duplicate characters and whitespaces

Set<Character> uniqueChars = new HashSet<>();

for (char ch : concatenated.toCharArray()) {

if (ch != ' ') {

uniqueChars.add(ch);

}

}

// Step 3: Sort the characters in descending order

List<Character> sortedList = new ArrayList<>(uniqueChars);

Collections.sort(sortedList, Collections.reverseOrder());

// Step 4: Create the final string

StringBuilder result = new StringBuilder();

for (char ch : sortedList) {

result.append(ch);

}

// Step 5: If the result is empty, return "null", else return the result

return result.length() > 0 ? result.toString() : "null";

}

public static void main(String[] args) {

// Using Scanner to take input from the user

Scanner scanner = new Scanner(System.in);

//nter the first string: ");

String input1 = scanner.nextLine();

// Input 2

String input2 = scanner.nextLine();

// Calling the mergeAndSort method and printing the result

String result = mergeAndSort(input1, input2);

System.out.println(result);

// Closing the scanner

scanner.close();

}

}

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

EXERCISE 2

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:

import java.util.Scanner;

public class WordProcessor {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input the string of words

String input = sc.nextLine();

// Input the 2-digit number

int number = sc.nextInt();

// Split the string into words

String[] words = input.split(" ");

// Extract the two positions from the 2-digit number

int pos1 = number / 10; // first digit (ten's place)

int pos2 = number % 10; // second digit (unit's place)

// Adjust positions for 0-based index

pos1--;

pos2--;

// Process both words

String result1 = processWord(words[pos1]);

String result2 = processWord(words[pos2]);

// Combine the results and print

String result = result1 + " " + result2;

System.out.println(result);

}

// Function to process each word

private static String processWord(String word) {

int len = word.length();

int mid = len / 2;

// Handle odd-length words correctly by including the middle letter in both parts

String middleToBegin;

String middleToEnd;

if (len % 2 == 0) {

// Even length

middleToBegin = new StringBuilder(word.substring(0, mid)).reverse().toString();

middleToEnd = word.substring(mid);

} else {

// Odd length

middleToBegin = new StringBuilder(word.substring(0, mid + 1)).reverse().toString();

middleToEnd = word.substring(mid);

}

// Combine both parts

return middleToBegin + middleToEnd;

}

}

|  | **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 |  |

EXERCISE 3

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:

import java.util.Scanner;

public class StringManipulation {

// Method to find the alphabet based on the given logic

public static char findChar(char ch1, char ch2) {

if (ch1 == ch2) {

// Case 1: If both characters are the same

return ch1;

} else {

// Case 2: If both characters are different

int max = Math.max(ch1 - 'a' + 1, ch2 - 'a' + 1);

int min = Math.min(ch1 - 'a' + 1, ch2 - 'a' + 1);

int pos = max - min;

return (char) ('a' + pos - 1); // Position starts at 1, so adjust by -1

}

}

// Method to process the input string and generate the output

public static String processString(String input) {

// Split the input string by ":"

String[] pairs = input.split(":");

StringBuilder result = new StringBuilder();

// Process each pair of characters

for (String pair : pairs) {

char ch1 = pair.charAt(0);

char ch2 = pair.charAt(1);

// Add the corresponding character to the result

result.append(findChar(ch1, ch2));

}

// Convert result to uppercase as specified

return result.toString().toUpperCase();

}

public static void main(String[] args) {

// Using Scanner to take input from the user

Scanner scanner = new Scanner(System.in);

// Input from use

//System.out.print("Enter the string (e.g., ww:ii:pp:rr:oo): ");

String input = scanner.nextLine();

// Process the string and get the result

String result = processString(input);

// Print the result

System.out.println( result);

// Close the scanner

scanner.close();

}

}

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | ww:ii:pp:rr:oo | WIPRO | WIPRO |  |
|  | zx:za:ee | BYE | BYE |  |