WEEK-06

### Question 1

### Given 2 strings inputl & 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:

### 

### Program:

### import java.util.LinkedHashSet;

### import java.util.Scanner;

### import java.util.Set;

### public class StringProcessor {

### public static String processStrings(String input1, String input2) {

### // Concatenate both strings

### String combined = input1 + input2;

### // Remove spaces and create a set to remove duplicates

### Set<Character> charSet = new LinkedHashSet<>();

### for (char c : combined.toCharArray()) {

### if (c != ' ') {

### charSet.add(c);

### }

### }

### // If the set is empty, return "null"

### if (charSet.isEmpty()) {

### return "null";

### }

### // Convert set to an array and sort it in descending order

### Character[] uniqueChars = charSet.toArray(new Character[0]);

### java.util.Arrays.sort(uniqueChars, java.util.Collections.reverseOrder());

### // Build the result string

### StringBuilder result = new StringBuilder();

### for (char c : uniqueChars) {

### result.append(c);

### }

### return result.toString();

### }

### public static void main(String[] args) {

### Scanner scanner = new Scanner(System.in);

### 

### // Prompt for the first input

### String input1 = scanner.nextLine();

### 

### // Prompt for the second input

### String input2 = scanner.nextLine();

### // Process and display the result

### String result = processStrings(input1, input2);

### System.out.println(result);

### 

### scanner.close();

### }

### }

| Test | Input | Expected | Got |
| --- | --- | --- | --- |
| 1 | appleorange | rponlgea | rponlgea |
| 2 | fruitsare good | utsroigfeda | utsroigfeda |
| 3 |  | null | null |

### Question 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:

### 

### Program:

### import java.util.Scanner;

### public class MiddleExtractor {

### // Method to process the input string and the 2-digit number

### public static String processInput(String input1, int input2) {

### String[] words = input1.split(" "); // Split input string into words

### int firstWordIndex = input2 / 10 - 1; // Get the index of the first word (0-based)

### int secondWordIndex = input2 % 10 - 1; // Get the index of the second word (0-based)

### // Validate indices

### if (firstWordIndex < 0 || firstWordIndex >= words.length || secondWordIndex < 0 || secondWordIndex >= words.length) {

### return "Invalid word index in input2";

### }

### // Process both words

### String processedFirstWord = processWord(words[firstWordIndex]);

### String processedSecondWord = processWord(words[secondWordIndex]);

### // Return the processed words separated by a space

### return processedFirstWord + " " + processedSecondWord;

### }

### // Method to process a single word

### private static String processWord(String word) {

### int length = word.length();

### int middleIndex = length / 2; // Find the middle index

### // Extract Middle-to-Begin part

### String middleToBegin;

### if (length % 2 == 0) {

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

### } else {

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

### }

### // Extract Middle-to-End part

### String middleToEnd = word.substring(middleIndex); // From middle to end

### 

### // Combine both parts

### return middleToBegin + middleToEnd;

### }

### // Main method

### public static void main(String[] args) {

### Scanner scanner = new Scanner(System.in);

### 

### // Get user input

### String input1 = scanner.nextLine();

### 

### int input2 = scanner.nextInt();

### // Process and display the result

### String result = processInput(input1, input2);

### System.out.println(result);

### 

### scanner.close();

### }

### }

| Input | Expected | Got |  |
| --- | --- | --- | --- |
|  | Today is a Nice Day41 | iNce doTday | iNce doTday |
|  | Fruits like Mango and Apple are common but Grapes are rare39 | naMngo arGpes | naMngo arGpes |

### Question 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:

### 

### Program:

### import java.util.Scanner;

### public class WordProcessor {

### 

### public static String processInput(String input1) {

### String[] words = input1.split(":"); // Split the input string by colon

### StringBuilder output = new StringBuilder();

### 

### for (String word : words) { // Corrected for loop syntax

### // Check if the word has at least 2 characters

### if (word.length() < 2) {

### continue; // Skip words that are too short

### }

### 

### char firstChar = word.charAt(0);

### char secondChar = word.charAt(1);

### 

### // Check if the first and second characters are the same

### if (firstChar == secondChar) {

### output.append(Character.toUpperCase(firstChar));

### } else {

### // Calculate the positions of the characters

### int pos1 = firstChar - 'a' + 1;

### int pos2 = secondChar - 'a' + 1;

### int diff = Math.abs(pos1 - pos2); // Calculate the absolute difference

### 

### // Ensure diff is within bounds

### char resultChar = (char) ('a' + (diff - 1)); // Find resulting character

### output.append(Character.toUpperCase(resultChar));

### }

### }

### 

### return output.toString(); // Return the processed string

### }

### public static void main(String[] args) {

### Scanner scan = new Scanner(System.in);

### String a = scan.nextLine(); // Get user input

### System.out.println(processInput(a)); // Process and print result

### scan.close(); // Close the scanner

### }

### }

### 