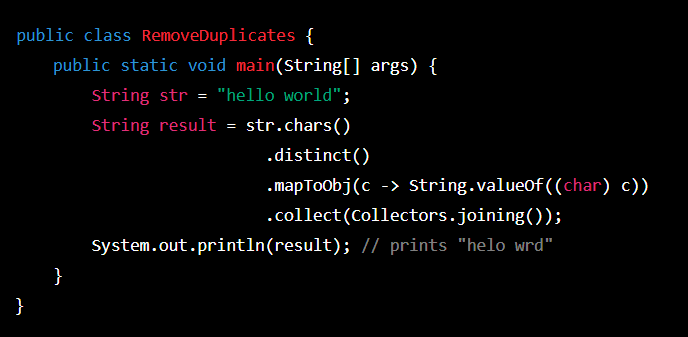
Strings in Java 3

Assignment

Question 1- WAP(write a program) to remove Duplicate for a string.(Take any String example with duplicates character)

Ans.

Certainly. Here’s a program that removes duplicates from a string in shorter format:



In this program, we first declare a ‘String’ variable ‘str’ that contains the input string with duplicates. We then call the ‘chars’ method on ‘str’ to obtain an ‘InterStream’ of the unicode code points for each character in the string. We then call the ‘distinct’ method on the ‘IntStreame’ to obtain a stream of unique code points.

We then use the ‘mapToObj’ method to map each code point to a ‘String’ representation of the corresponding character. Finally, we use the ‘collect’ method with ‘Collectors.joining’

To concatenate the resulting ‘string’ objects into a single spring.

The resulting string contains only the unique characters from the original string.

Question 2- WAP to print Duplicates characters from the String

Ans-

Here’s is program that prints the duplicate characters in a given string:

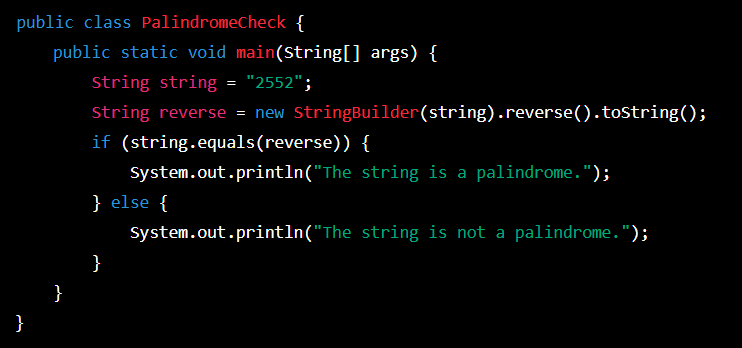


The program first creates a ‘HashMap’ to store the frequency of each character in the string. It then iterates through the string and updates the frequency count in the map. Finally, it prints out any characters that have a frequency greater than 1, which indicates that they are duplicates.

Question 3- Write a program to check if “2552” is palindrome or not.

Ans-

Here's a program:



This program uses the ‘StringBuilder’ class to reverse the original string and stores the reversed string in a new ‘reverse’ variable. It then compares the original spring and the reversed string using the ‘equals’ method. If they are equal, the program prints”The string is a palindrome”.

Question 4- Write a program to count the number of consonants, vowels, special characters in a string.

Ans-

Here is the program to count the number of consonants, vowels, and special characters in a given string:

import java.util.Scanner;

public class CountChars {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a string: ");

String str = scanner.nextLine();

int vowels = 0, consonants = 0, specialChars = 0;

str = str.toLowerCase();

for(int i = 0; i < str.length(); ++i) {

char ch = str.charAt(i);

if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

++vowels;

}

else if((ch >= 'a' && ch <= 'z')) {

++consonants;

}

else {

++specialChars;

}

}

System.out.println("Number of vowels: " + vowels);

System.out.println("Number of consonants: " + consonants);

System.out.println("Number of special characters: " + specialChars);

}

}

The program first prompts the user to enter a string . it then initializes variables to count the number of vowels, consonants, and special characters in the string.

The ‘for’ loop iterates through each character in the string. If the character is a vowel (a, e, i, o, or u), the ‘vowels’ variable is incremented. If the character is a consonant, the ‘consonants’ variable is incremented. If the character is neither a vowel nor a consonant (i.e., a special character), the ‘specialChars’ variable is incremented.

Finally, the program prints out the number of vowels, consonants, and special characters in the string.

Question 5- Write a program to implement Anagram Checking least inbuilt methods being used.

Ans-

Here’s a program to implement Anagram Checking least inbuilt methods.

import java.util.Arrays;

import java.util.Scanner;

public class AnagramChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Read in the two strings to check

System.out.println("Enter the first string:");

String str1 = scanner.nextLine();

System.out.println("Enter the second string:");

String str2 = scanner.nextLine();

// Check if the two strings are anagrams

boolean areAnagrams = checkIfAnagrams(str1, str2);

if (areAnagrams) {

System.out.println(str1 + " and " + str2 + " are anagrams.");

} else {

System.out.println(str1 + " and " + str2 + " are not anagrams.");

}

}

public static boolean checkIfAnagrams(String str1, String str2) {

// Remove all spaces and convert to lowercase

str1 = str1.replaceAll("\\s", "").toLowerCase();

str2 = str2.replaceAll("\\s", "").toLowerCase();

// If the strings are different lengths, they can't be anagrams

if (str1.length() != str2.length()) {

return false;

}

// Convert the strings to char arrays and sort them

char[] charArray1 = str1.toCharArray();

char[] charArray2 = str2.toCharArray();

Arrays.sort(charArray1);

Arrays.sort(charArray2);

// Compare the sorted char arrays to see if they match

return Arrays.equals(charArray1, charArray2);

}

}

The ‘checkIfAnagrams’ method takes in two strings, removes all spaces and converts them to lowercase. It then checks if the two strings are the same length. And if they are not, it returns false. If they are the same length, convert the strings to char arrays, sort them, and compare them to see if they are equal using the ‘Arrays.equals’ method.

The main method reads in two strings from the user, calls the ‘checkIfAnagrams’ method, and prints out the result.

Question 6- write a program to implement pangram Checking with least inbuilt methods being used.

Ans-

Here’s a program to implement pangram Checking without using inbuilt methods:

import java.util.Scanner;

public class PangramChecker {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter a string: ");

String str = input.nextLine();

if (isPangram(str)) {

System.out.println(str + " is a pangram.");

} else {

System.out.println(str + " is not a pangram.");

}

}

public static boolean isPangram(String str) {

str = str.toLowerCase();

boolean[] alphabet = new boolean[26];

int index;

for (int i = 0; i < str.length(); i++) {

if ('a' <= str.charAt(i) && str.charAt(i) <= 'z') {

index = str.charAt(i) - 'a';

alphabet[index] = true;

}

}

for (int i = 0; i < alphabet.length; i++) {

if (!alphabet[i]) {

return false;

}

}

return true;

}

}

The program prompts the user to enter a string, and then passes the string to the ‘isPangram’

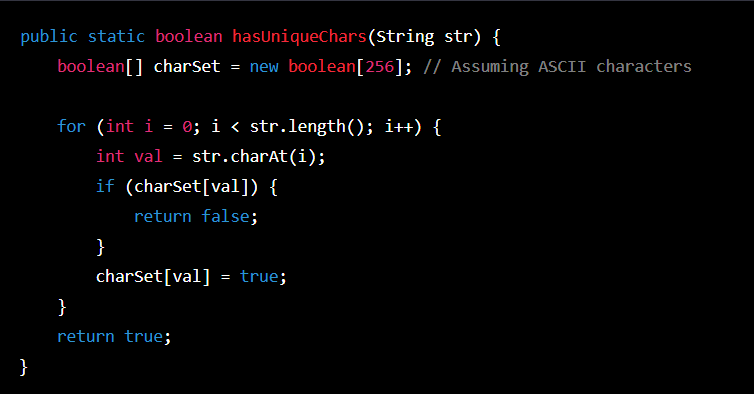
method.The method converts the string to lowercase, initializes a boolean array to keep track of the letters in the alphabet, and then loops through the string to see the corresponding value in the array to true for each letter. Finally, the method checks if all the values in the array are true, which indicates that all the letters in the alphabet are present in the string is a pangram and false otherwise.

Note! That this program doesn’t use many built-in methods other than ‘toLowerCase',’charAt’, and ‘length’.

Question 7- Write a program to find if a string contains all unique characters.

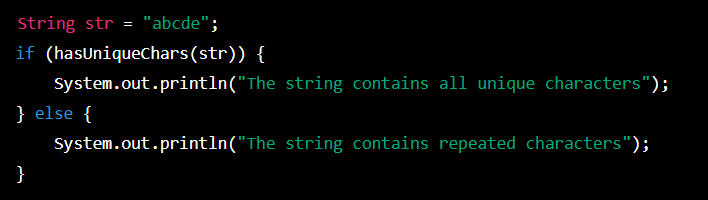
Ans-

Here’s a program that checks if a string contains all unique characters.



This program uses a boolean array of size 256 to keep track of the characters that have been seen. If then iterates through the string, checking if the character at each position has been seen before. If it has, the function returns false. If it reaches the end of the string without finding any repeating characters, it returns true.

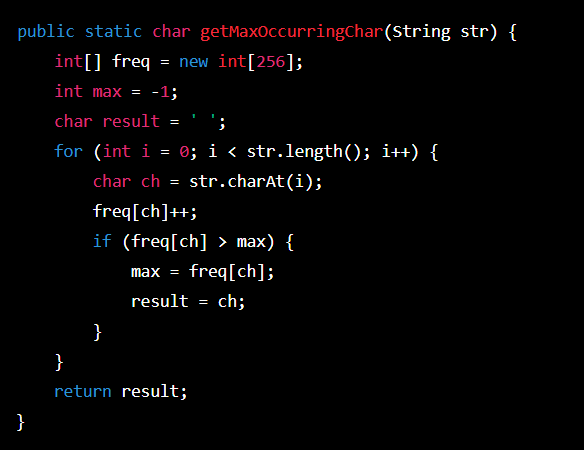
To use this function, simply call ‘hasUniqueChars’ with a string as the argument. For example:



Question 8- Write a program to find the maximum occurring character in a String

Ans-

Here’s a program to find the maximum occurring character in a String:



This program uses an integer array ‘freq’ to keep track of the frequency of each character in the input String ‘str’ . It then iterates through the string, updating the frequency count for each character. If the frequency count for a character is greater than the current maximum, it updates the maximum and stores the character as the result.

To use this program, simply call the ‘getMaxOccurringChar’ method with your String as an argument. For example:

String str = "hello world";

char maxChar = getMaxOccurringChar(str);

System.out.println("The maximum occurring character in '" + str + "' is '" + maxChar + "'");

This would output:

****