1. Remove Duplicates from a String:

java

public class RemoveDuplicates {

public static String removeDuplicates(String input) {

StringBuilder result = new StringBuilder();

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

char currentChar = input.charAt(i);

if (result.indexOf(String.valueOf(currentChar)) == -1) {

result.append(currentChar);

}

}

return result.toString();

}

public static void main(String[] args) {

String input = "programming";

String output = removeDuplicates(input);

System.out.println(output); // Output: "progamin"

}

}

2. Print Duplicates Characters from a String:

java

public class PrintDuplicates {

public static void printDuplicates(String input) {

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

char currentChar = input.charAt(i);

if (input.lastIndexOf(currentChar) > i) {

System.out.print(currentChar + " ");

}

}

}

public static void main(String[] args) {

String input = "programming";

System.out.print("Duplicates: ");

printDuplicates(input); // Output: "Duplicates: g r m"

}

}

3. Check if "2552" is a palindrome:

java

public class PalindromeCheck {

public static boolean isPalindrome(String input) {

String reversed = new StringBuilder(input).reverse().toString();

return input.equals(reversed);

}

public static void main(String[] args) {

String input = "2552";

boolean isPalindromic = isPalindrome(input);

System.out.println("Is Palindrome: " + isPalindromic); // Output: "Is Palindrome: true"

}

}

4. Count Consonants, Vowels, and Special Characters in a String:

java

public class CharacterCount {

public static void countCharacters(String input) {

int vowels = 0;

int consonants = 0;

int specialChars = 0;

input = input.toLowerCase();

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

char ch = input.charAt(i);

if (Character.isLetter(ch)) {

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

vowels++;

} else {

consonants++;

}

} else {

specialChars++;

}

}

System.out.println("Vowels: " + vowels);

System.out.println("Consonants: " + consonants);

System.out.println("Special Characters: " + specialChars);

}

public static void main(String[] args) {

String input = "Hello, World! 123";

countCharacters(input);

}

}

5. Anagram Checking (without using built-in methods):

java

public class AnagramCheck {

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

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

return false;

}

int[] count = new int[256];

for (char c : str1.toCharArray()) {

count[c]++;

}

for (char c : str2.toCharArray()) {

count[c]--;

}

for (int value : count) {

if (value != 0) {

return false;

}

}

return true;

}

public static void main(String[] args) {

String str1 = "listen";

String str2 = "silent";

boolean areAnagrams = areAnagrams(str1, str2);

System.out.println("Are Anagrams: " + areAnagrams); // Output: "Are Anagrams: true"

}

}

6. Pangram Checking (without using built-in methods):

java

public class PangramCheck {

public static boolean isPangram(String input) {

input = input.toLowerCase();

boolean[] alphabet = new boolean[26];

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

char ch = input.charAt(i);

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

alphabet[ch - 'a'] = true;

}

}

for (boolean letter : alphabet) {

if (!letter) {

return false;

}

}

return true;

}

public static void main(String[] args) {

String input = "The quick brown fox jumps over the lazy dog";

boolean isPangram = isPangram(input);

System.out.println("Is Pangram: " + isPangram); // Output: "Is Pangram: true"

}

}

7. Check if String contains all unique characters (without using built-in methods):

java

public class UniqueCharactersCheck {

public static boolean hasUniqueCharacters(String input) {

int length = input.length();

for (int i = 0; i < length - 1; i++) {

for (int j = i + 1; j < length; j++) {

if (input.charAt(i) == input.charAt(j)) {

return false;

}

}

}

return true;

}

public static void main(String[] args) {

String input = "abcdefg";

boolean hasUnique = hasUniqueCharacters(input);

System.out.println("Has Unique Characters: " + hasUnique); // Output: "Has Unique Characters: true"

}

}

8. Find the Maximum Occurring Character in a String:

java

public class MaxOccurringCharacter {

public static char findMaxOccurringCharacter(String input) {

int[] charCount = new int[256];

int maxCount = 0;

char maxChar = 0;

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

char ch = input.charAt(i);

charCount[ch]++;

if (charCount[ch] > maxCount) {

maxCount = charCount[ch];

maxChar = ch;

}

}

return maxChar;

}

public static void main(String[] args) {

String input = "programming";

char maxChar = findMaxOccurringCharacter(input);

System.out.println("Maximum Occurring Character: " + maxChar); // Output: "Maximum Occurring Character: g"

}

}