# Strings in Java

# Assignment

1. WAP(Write a Program) to remove Duplicates from a String.(Take any String example with duplicates character

   Ans1. import java.util.LinkedHashSet;

   import java.util.Set;

   public class RemoveDuplicatesFromString {

      public static void main(String[] args) {

         String input = "programming";

         String result = removeDuplicates(input);

         System.out.println("Original String: " + input);

```
    System.out.println("String after removing duplicates: " + result);
}

public static String removeDuplicates(String str) {
    // Use a LinkedHashSet to maintain the order and remove duplicates
    Set<Character> set = new LinkedHashSet<>();

    // Add each character of the string to the set
    for (char c : str.toCharArray()) {
        set.add(c);
    }

    // Build the string from the set
    StringBuilder sb = new StringBuilder();
    for (Character c : set) {
```

```
            sb.append(c);

        }

        return sb.toString();

    }

}
```

2. WAP to print Duplicates characters from the String

Ans. 2.

```
import java.util.HashMap;
import java.util.Map;

public class FindDuplicateCharacters {
    public static void main(String[] args)
{
        String input = "programming";
        System.out.println("Original
String: " + input);
        System.out.print("Duplicate
characters: ");
        printDuplicateCharacters(input);
    }

    public static void
printDuplicateCharacters(String str) {
        // Create a HashMap to store
```

```
character frequencies
        Map<Character, Integer>
charFrequencyMap = new
HashMap<>();


        // Convert the string to a character
array
        char[] charArray =
str.toCharArray();


        // Count the frequency of each
character
        for (char c : charArray) {
            if
(charFrequencyMap.containsKey(c)) {
                charFrequencyMap.put(c,
charFrequencyMap.get(c) + 1);
            } else {
                charFrequencyMap.put(c, 1);
            }
        }


        // Print characters that have a
frequency greater than 1
        for (Map.Entry<Character,
Integer> entry :
charFrequencyMap.entrySet()) {
            if (entry.getValue() > 1) {

System.out.print(entry.getKey() + " ");
            }
        }
```

```
        }
    }
```

3. WAP to check if “2552” is palindrome or not

Ans3.
```
public class PalindromeCheck {
    public static void main(String[] args) {
        String input = "2552";

        if (isPalindrome(input)) {
            System.out.println(input + " is a palindrome.");
        } else {
            System.out.println(input + " is not a palindrome.");
        }
    }

    public static boolean isPalindrome(String str) {
        // Compare the string with its reverse
        String reversed = new StringBuilder(str).reverse().toString();
        return str.equals(reversed);
    }
}
```

4. WAP to count the number of consonants, vowels, special characters in a String

Ans4.
```
public class CountCharacters {
```

```
   public static void main(String[] args)
{
      String input = "Hello World! 123
@#";
      int vowelCount = 0,
consonantCount = 0, specialCharCount
= 0;


      // Convert the string to lowercase
to simplify checks
      input = input.toLowerCase();


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


         // Check if the character is a
letter
         if (ch >= 'a' && ch <= 'z') {
            // Check if it is a vowel
            if (ch == 'a' || ch == 'e' || ch ==
'i' || ch == 'o' || ch == 'u') {
               vowelCount++;
            } else {
               consonantCount++;
            }
         }
         // Check if the character is a
special character or space
         else if (!Character.isDigit(ch)
&& !Character.isWhitespace(ch)) {
```

```
                specialCharCount++;
            }
        }

        System.out.println("Vowels: " +
vowelCount);
        System.out.println("Consonants: "
+ consonantCount);
        System.out.println("Special
Characters: " + specialCharCount);
    }
}
```

5. WAP to implement Anagram Checking least inbuilt methods being used

Ans5.
```
public class AnagramCheck {
    public static void main(String[] args)
{
        String str1 = "listen";
        String str2 = "silent";

        if (areAnagrams(str1, str2)) {
            System.out.println(str1 + " and "
+ str2 + " are anagrams.");
        } else {
            System.out.println(str1 + " and "
+ str2 + " are not anagrams.");
        }
    }
```

```
   public static boolean
areAnagrams(String str1, String str2) {
        // If lengths are not the same, they
cannot be anagrams
        if (str1.length() != str2.length()) {
            return false;
        }

        // Create an array to count the
frequency of each character
        int[] charCount = new int[26]; //
Assuming only lowercase letters

        // Convert strings to lowercase
(optional, but ensures case insensitivity)
        str1 = str1.toLowerCase();
        str2 = str2.toLowerCase();

        // Increment counts for characters
in str1
        for (int i = 0; i < str1.length(); i++)
{
            charCount[str1.charAt(i) -
'a']++;
        }

        // Decrement counts for characters
in str2
        for (int i = 0; i < str2.length(); i++)
{
            charCount[str2.charAt(i) - 'a']--;
```

```
        }

        // If all counts are zero, the strings
    are anagrams
        for (int count : charCount) {
            if (count != 0) {
                return false;
            }
        }

        return true;
    }
}
```

6. WAP to implement Pangram Checking with least inbuilt methods being used

Ans6.

```
public class PangramCheck {
    public static void main(String[] args) {
        String input = "The quick brown fox
jumps over the lazy dog";

        if (isPangram(input)) {
            System.out.println("The input is a
pangram.");
        } else {
            System.out.println("The input is not
a pangram.");
```

```
        }
    }

    public static boolean isPangram(String str) {
        // Create a boolean array to mark the presence of each letter
        boolean[] mark = new boolean[26]; // 26 letters in the alphabet
        int index;

        // Convert the string to lowercase to handle both uppercase and lowercase letters
        str = str.toLowerCase();

        // Iterate through the string and mark the presence of each letter
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);

            // Check if the character is a lowercase letter
            if (ch >= 'a' && ch <= 'z') {
                index = ch - 'a'; // Calculate the index (0 for 'a', 1 for 'b', ..., 25 for 'z')
                mark[index] = true; // Mark the letter as present
```

```
        }
    }

    // Check if all letters are marked
    for (boolean letterMarked : mark) {
        if (!letterMarked) {
            return false; // If any letter is not marked, it's not a pangram
        }
    }

    return true; // All letters are marked, so it's a pangram
  }
}
```

7. WAP to find if String contains all unique characters

   Ans7.

```
import java.util.HashSet;

public class UniqueCharacters {
    public static boolean hasUniqueChars(String str) {
        HashSet<Character> charSet = new HashSet<>();
```

```
        for (char ch : str.toCharArray()) {
            if (charSet.contains(ch)) {
                return false; // Duplicate character found
            }
            charSet.add(ch);
        }
        return true; // All characters are unique
    }

    public static void main(String[] args) {
        String test = "hello";  // Change input here
        System.out.println("Is Unique? " + hasUniqueChars(test));
    }
}
```

8. WAP to find the maximum occurring character in a String

Ans8.
```
public class MaxOccurringChar {
    public static char getMaxOccurringChar(String str) {
        int[] freq = new int[256];  // ASCII character frequency array
```

```
        // Count frequency of each character
        for (char ch : str.toCharArray()) {
            freq[ch]++;
        }

        // Find the character with maximum
frequency
        int maxCount = 0;
        char maxChar = ' ';
        for (char ch : str.toCharArray()) {
            if (freq[ch] > maxCount) {
                maxCount = freq[ch];
                maxChar = ch;
            }
        }
        return maxChar;
    }

    public static void main(String[] args) {
        String test = "hello world";
        System.out.println("Max Occurring
Character: " + getMaxOccurringChar(test));
    }
}
```