1. WAP to Remove Duplicates from a String (Take Any String Example with Duplicate Characters):

```
import java.util.LinkedHashSet;
public class RemoveDuplicates {
    public static void main(String[] args) {
        String str = "programming";
        LinkedHashSet<Character> set = new LinkedHashSet<>();
        for (char c : str.toCharArray()) {
            set.add(c);
        }
        StringBuilder result = new StringBuilder();
        for (char c : set) {
            result.append(c);
        }
        System.out.println("String after removing duplicates: " + result.toString());
    }
}
```

2. WAP to Print Duplicate Characters from the String:

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

public class PrintDuplicates {
    public static void main(String[] args) {
        String str = "programming";
        Map<Character, Integer> charCountMap = new HashMap<>();
         for (char c : str.toCharArray()) {
```

```
            charCountMap.put(c, charCountMap.getOrDefault(c, 0) + 1);
        }
        System.out.println("Duplicate characters:");
        for (Map.Entry<Character, Integer> entry : charCountMap.entrySet()) {
            if (entry.getValue() > 1) {
                System.out.println(entry.getKey() + ": " + entry.getValue());
            }
        }
    }
}
```

3. WAP to Check if “2552” is Palindrome or Not:

```
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "2552";
        String reversed = new StringBuilder(str).reverse().toString();
        if (str.equals(reversed)) {
            System.out.println(str + " is a palindrome.");
        } else {
            System.out.println(str + " is not a palindrome.");
        }
    }
}
```

4. WAP to Count the Number of Consonants, Vowels, Special Characters in a String:

```
public class CountCharacters {
```

```
    public static void main(String[] args) {
        String str = "Hello, World!";
        int vowels = 0, consonants = 0, specialChars = 0;

        for (char c : str.toCharArray()) {
            if (Character.isLetter(c)) {
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ||
                    c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U') {
                    vowels++;
                } else {
                    consonants++;
                }
            } else if (!Character.isWhitespace(c)) {
                specialChars++;
            }
        }

        System.out.println("Number of vowels: " + vowels);
        System.out.println("Number of consonants: " + consonants);
        System.out.println("Number of special characters: " + specialChars);
    }
}
```

5. WAP to Implement Anagram Checking (Least Inbuilt Methods Being Used):

```
import java.util.Arrays;
```

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

        if (str1.length() != str2.length()) {
            System.out.println("Not anagrams.");
            return;
        }

        char[] arr1 = str1.toCharArray();
        char[] arr2 = str2.toCharArray();
         Arrays.sort(arr1);
        Arrays.sort(arr2);
        boolean isAnagram = Arrays.equals(arr1, arr2);
        System.out.println("Are the strings anagrams? " + isAnagram);
    }
}
```

6. WAP to Implement Pangram Checking (Least Inbuilt Methods Being Used):

```
public class PangramCheck {
    public static void main(String[] args) {
        String str = "The quick brown fox jumps over the lazy dog";
        boolean[] alphabet = new boolean[26];
        int index;
```

```
        str = str.toLowerCase();

        for (char c : str.toCharArray()) {
            if (Character.isLetter(c)) {
                index = c - 'a';
                alphabet[index] = true;
            }
        }

        boolean isPangram = true;
        for (boolean b : alphabet) {
            if (!b) {
                isPangram = false;
                break;
            }
        }

        System.out.println("Is the string a pangram? " + isPangram);
    }
}
```

7. WAP to Find if String Contains All Unique Characters:

```
public class UniqueCharacters {
    public static void main(String[] args) {
        String str = "abcdefg";
        boolean[] charSet = new boolean[256];
```

```
        for (char c : str.toCharArray()) {
            if (charSet[c]) {
                System.out.println("The string does not contain all unique characters.");
                return;
            }
            charSet[c] = true;
        }

        System.out.println("The string contains all unique characters.");
    }
}
```

8. WAP to Find the Maximum Occurring Character in a String:

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

public class MaxOccurringCharacter {
    public static void main(String[] args) {
        String str = "character";
        Map<Character, Integer> charCountMap = new HashMap<>();

        for (char c : str.toCharArray()) {
            charCountMap.put(c, charCountMap.getOrDefault(c, 0) + 1);
        }
```

```
        char maxChar = ' ';
        int maxCount = 0;

        for (Map.Entry<Character, Integer> entry : charCountMap.entrySet()) {
            if (entry.getValue() > maxCount) {
                maxCount = entry.getValue();
                maxChar = entry.getKey();
            }
        }

        System.out.println("Maximum occurring character: " + maxChar);
    }
}
```