1. **Write a Program to remove Duplicates from String.**

Ans.

```
import java.util.HashSet;
import java.util.Set;

public class RemoveDuplicatesFromString {
    public static String removeDuplicates(String str) {
        Set<Character> set = new HashSet<>();
        StringBuilder sb = new StringBuilder();

        for (char ch : str.toCharArray()) {
            if (!set.contains(ch)) {
                set.add(ch);
                sb.append(ch);
            }
        }

        return sb.toString();
    }

    public static void main(String[] args) {
        String input = "Hello, World!";
        String result = removeDuplicates(input);
        System.out.println("Original String: " + input);
        System.out.println("String after removing duplicates: " + result);
    }
}
```

2. **Write a Program to Print duplicate characters from the string.**

Ans.

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

public class PrintDuplicateCharacters {
    public static void printDuplicateCharacters(String str) {
        // Create a HashMap to store character frequency
        Map<Character, Integer> charFrequencyMap = new HashMap<>();

        // Iterate through each character in the string
        for (char ch : str.toCharArray()) {
            // Increment the count if character already exists in the map
            if (charFrequencyMap.containsKey(ch)) {
                int count = charFrequencyMap.get(ch);
```

```
            charFrequencyMap.put(ch, count + 1);
        } else {
            // Add the character to the map if it doesn't exist
            charFrequencyMap.put(ch, 1);
        }
    }

    // Print the characters with count greater than 1
    System.out.println("Duplicate characters in the string: ");
    for (Map.Entry<Character, Integer> entry : charFrequencyMap.entrySet()) {
        if (entry.getValue() > 1) {
            System.out.println(entry.getKey());
        }
    }
}

public static void main(String[] args) {
    String input = "Hello, World!";
    System.out.println("Original String: " + input);
    printDuplicateCharacters(input);
}
}
```

3. **Write a Program to check if “2552” is palindrome or not.**

Ans.

```
public class PalindromeCheck {
    public static boolean isPalindrome(String str) {
        int start = 0;
        int end = str.length() - 1;

        while (start < end) {
            if (str.charAt(start) != str.charAt(end)) {
                return false;
            }
            start++;
            end--;
        }

        return true;
    }

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

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

4. **Write a Program to count the number of consonants, vowels, special characters in a string.**

Ans.

```
public class CharacterCount {
    public static void countCharacters(String str) {
        int vowelCount = 0;
        int consonantCount = 0;
        int specialCharCount = 0;

        str = str.toLowerCase();

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

            if (Character.isLetter(ch)) {
                if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                    vowelCount++;
                } else {
                    consonantCount++;
                }
            } else {
                specialCharCount++;
            }
        }

        System.out.println("Number of vowels: " + vowelCount);
        System.out.println("Number of consonants: " + consonantCount);
        System.out.println("Number of special characters: " + specialCharCount);
    }

    public static void main(String[] args) {
        String input = "Hello, World!";
        System.out.println("Original String: " + input);
        countCharacters(input);
```

```
    }
}
```

5. **Write a Program to implement anagram checking least inbuilt methods being used.**

Ans.

```
public class AnagramCheck {
    public static boolean areAnagrams(String str1, String str2) {
        if (str1.length() != str2.length()) {
            return false;
        }

        int[] charCount = new int[26];

        for (int i = 0; i < str1.length(); i++) {
            char ch1 = str1.charAt(i);
            char ch2 = str2.charAt(i);

            charCount[ch1 - 'a']++;
            charCount[ch2 - 'a']--;
        }

        for (int count : charCount) {
            if (count != 0) {
                return false;
            }
        }

        return true;
    }

    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.");
        }
    }
}
```

6. **Write a Program to implement pangram checking least inbuilt methods being used.**

Ans.

```
public class PangramCheck {
    public static boolean isPangram(String str) {
        // Create a boolean array to track occurrence of each letter
        boolean[] letters = new boolean[26];

        // Convert the string to lowercase
        str = str.toLowerCase();

        // Iterate through each character in the string
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            // Check if the character is an alphabet
            if (ch >= 'a' && ch <= 'z') {
                letters[ch - 'a'] = true;
            }
        }

        // Check if any letter is missing
        for (boolean letter : letters) {
            if (!letter) {
                return false;
            }
        }

        return true;
    }

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

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

7. **Write a Program to find if string contains all unique characters.**

Ans.

```
public class UniqueCharacterCheck {
    public static boolean hasUniqueCharacters(String str) {
        if (str.length() > 128) {
            return false; // Assuming ASCII character set
        }

        boolean[] charSet = new boolean[128];

        for (int i = 0; i < str.length(); i++) {
            int asciiValue = str.charAt(i);
            if (charSet[asciiValue]) {
                return false;
            }
            charSet[asciiValue] = true;
        }

        return true;
    }

    public static void main(String[] args) {
        String input = "abcdefg";
        if (hasUniqueCharacters(input)) {
            System.out.println("The string contains all unique characters.");
        } else {
            System.out.println("The string does not contain all unique characters.");
        }
    }
}
```

8. **Write a Program to find the maximum occurring characters in a string.**

Ans.

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

public class MaxOccurringCharacter {
    public static char findMaxOccurringCharacter(String str) {
        Map<Character, Integer> charCount = new HashMap<>();

        // Count the occurrence of each character in the string
        for (char ch : str.toCharArray()) {
            charCount.put(ch, charCount.getOrDefault(ch, 0) + 1);
        }
```

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

        // Find the character with the maximum occurrence
        for (Map.Entry<Character, Integer> entry : charCount.entrySet()) {
            char ch = entry.getKey();
            int count = entry.getValue();

            if (count > maxCount) {
                maxChar = ch;
                maxCount = count;
            }
        }

        return maxChar;
    }

    public static void main(String[] args) {
        String input = "Hello, World!";
        char maxChar = findMaxOccurringCharacter(input);
        System.out.println("Max occurring character: " + maxChar);
    }
}
```