PRACTICE PROBLEM 1:
String Creation and Manipulation

```
public class StringManipulation {
    public static void main(String[] args) {
        // 1. Create the same string "Java Programming" using 3 different methods:

        // String literal
        String s1 = "Java Programming";
        System.out.println("String Literal (s1): " + s1);

        // new String() constructor
        String s2 = new String("Java Programming");
        System.out.println("new String() (s2): " + s2);

        // Character array
        char[] charArray = {'J', 'a', 'v', 'a', ' ', 'P', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g'};
        String s3 = new String(charArray);
        System.out.println("Character Array (s3): " + s3);

        // 2. Compare the strings using == and .equals()
        System.out.println("\n--- String Comparison ---");
        System.out.println("s1 == s2: " + (s1 == s2));
        System.out.println("s1.equals(s2): " + s1.equals(s2));
        System.out.println("s1 == s3: " + (s1 == s3));
        System.out.println("s1.equals(s3): " + s1.equals(s3));
        System.out.println("s2 == s3: " + (s2 == s3));
        System.out.println("s2.equals(s3): " + s2.equals(s3));

        // Explanation of == vs. .equals()
        System.out.println("\nExplanation:");
        System.out.println("The `==` operator compares the memory addresses (references) of
the string objects.");
        System.out.println("In this case, s1, s2, and s3 are distinct objects in memory, even if
they have the same content.");
        System.out.println("The `.equals()` method compares the actual content of the strings
for equality.");
        System.out.println("Since s1, s2, and s3 all hold the same sequence of characters
(\"Java Programming\"), .equals() returns true for their comparisons.");

        // 3. Create a string with escape sequences
        String message = "Programming Quote:\n\"Code is poetry\" - Unknown\nPath:
C:\\Java\\Projects";
        System.out.println("\n--- String with Escape Sequences ---");
        System.out.println(message);
    }
}
```

PRACTICE PROBLEM 2:
String Input and Processing

```
import java.util.Scanner;

public class StringMethods {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Ask user for their full name (first and last name)
        System.out.print("Enter your full name (first and last name): ");
        String fullName = scanner.nextLine(); //

        // Ask user for their favorite programming language
        System.out.print("Enter your favorite programming language: ");
        String language = scanner.nextLine(); //

        // Ask user for a sentence about their programming experience
        System.out.print("Enter a sentence about your programming experience: ");
        String experienceSentence = scanner.nextLine(); //

        // Process the input:

        // 1. Extract first and last name separately
        String firstName = "";
        String lastName = "";
        int spaceIndex = fullName.indexOf(' '); // Find the index of the first space

        if (spaceIndex != -1) { // If a space exists
            firstName = fullName.substring(0, spaceIndex); // Extract the first name
            lastName = fullName.substring(spaceIndex + 1); // Extract the last name
        } else {
            // Handle cases where only one name is provided (e.g., if the user didn't enter a
space)
            firstName = fullName; // Treat the entire input as the first name
            lastName = "N/A"; // Indicate that the last name is not available
        }

        // 2. Count total characters in the sentence (excluding spaces)
        int charCount = 0;
        for (char c : experienceSentence.toCharArray()) { // Convert the string to a character
array
            if (!Character.isWhitespace(c)) { // Check if the character is not a whitespace
                charCount++;
            }
        }
        // Alternatively, using replaceAll() and length() methods:
```

```
        // int charCount = experienceSentence.replaceAll("\\s", "").length();


        // 3. Convert programming language to uppercase
        String upperCaseLanguage = language.toUpperCase(); // Convert the string to
uppercase

        // 4. Display a formatted summary using printf() for controlled output formatting
        System.out.printf("\n--- Summary of Your Input ---%n");
        System.out.printf("Full Name: %s %n", fullName);
        System.out.printf("First Name: %s, Last Name: %s %n", firstName, lastName);
        System.out.printf("Favorite Programming Language: %S %n", upperCaseLanguage); //
Use %S to print in uppercase
        System.out.printf("Characters in Your Experience Sentence (excluding spaces): %d
%n", charCount);

        scanner.close();
    }
}
```

PRACTICE PROBLEM 3:
String Arrays and Methods

```
import java.util.Arrays; // Import the Arrays class for array utility methods.

public class StringArrays {

    // Method to find the longest name in a string array
    public static String findLongestName(String[] names) {
        if (names == null || names.length == 0) { // Handle empty or null arrays
            return null;
        }

        String longestName = names[0]; // Assume the first name is the longest initially
        for (int i = 1; i < names.length; i++) { // Iterate through the rest of the array
            if (names[i].length() > longestName.length()) { // If current name is longer than the
assumed longest
                longestName = names[i]; // Update the longest name
            }
        }
        return longestName; // Return the longest name found
    }

    // Method to count how many names start with a given letter (case-insensitive)
    public static int countNamesStartingWith(String[] names, char letter) {
        if (names == null) { // Handle null arrays
            return 0;
        }

        int count = 0;
        char lowerCaseLetter = Character.toLowerCase(letter); // Convert the given letter to
lowercase for case-insensitive comparison

        for (String name : names) { // Iterate through the names in the array
            if (name != null && name.length() > 0 && Character.toLowerCase(name.charAt(0))
== lowerCaseLetter) { // Check if the name starts with the given letter (case-insensitive)
                count++;
            }
        }
        return count;
    }

    // Method to format all names to "Last, First" format
    public static String[] formatNames(String[] names) {
        if (names == null) { // Handle null arrays
            return null;
        }

        String[] formattedNames = new String[names.length]; // Create a new array to store
formatted names

        for (int i = 0; i < names.length; i++) {
            String name = names[i];
            int spaceIndex = name.indexOf(' '); // Find the index of the first space

            if (spaceIndex != -1) { // If a space exists (indicating a first and last name)
                String firstName = name.substring(0, spaceIndex); // Extract the first name
                String lastName = name.substring(spaceIndex + 1); // Extract the last name
                formattedNames[i] = lastName + ", " + firstName; // Format to "Last, First"
            } else {
                formattedNames[i] = name; // If no space (e.g., single name), keep it as is
            }
        }
        return formattedNames;
    }

    public static void main(String[] args) {
        String[] students = {"John Smith", "Alice Johnson", "Bob Brown", "Carol Davis", "David
Wilson"};

        // Test findLongestName method
        String longestName = findLongestName(students); // Call the findLongestName method
        System.out.println("Longest name: " + longestName); // Display the result

        // Test countNamesStartingWith method
        int countJ = countNamesStartingWith(students, 'J'); // Call the countNamesStartingWith
method
        System.out.println("Names starting with 'J' (case-insensitive): " + countJ); // Display the
result

        // Test formatNames method
        String[] formattedStudents = formatNames(students); // Call the formatNames method
        System.out.println("\nFormatted names:");
        for (String name : formattedStudents) { // Iterate and print the formatted names
            System.out.println(name);
        }
    }
}
```

PRACTICE PROBLEM 4:
Complete String Application

```
import java.util.*;

public class TextProcessor {

    // Method to clean and validate input
    public static String cleanInput(String input) {
        // Remove extra spaces and convert to proper case
        input = input.trim().replaceAll("\\s+", " ");
        return input;
    }

    // Method to analyze text
    public static void analyzeText(String text) {
        // Character count
        int charCount = text.length();

        // Word count
        String[] words = text.split("\\s+");
        int wordCount = words.length;

        // Sentence count (basic - split by '.', '!', '?')
        String[] sentences = text.split("[.!?]");
        int sentenceCount = sentences.length;

        // Find longest word
        String longestWord = "";
        for (String w : words) {
            if (w.length() > longestWord.length()) {
                longestWord = w;
            }
        }

        // Find most common character
        HashMap<Character, Integer> freq = new HashMap<>();
        for (char c : text.toCharArray()) {
            if (Character.isLetter(c)) {
                c = Character.toLowerCase(c);
                freq.put(c, freq.getOrDefault(c, 0) + 1);
            }
        }
        char mostCommon = ' ';
        int maxFreq = 0;
        for (Map.Entry<Character, Integer> entry : freq.entrySet()) {
            if (entry.getValue() > maxFreq) {
                maxFreq = entry.getValue();
                mostCommon = entry.getKey();
            }
        }

        // Display statistics
        System.out.println("Characters: " + charCount);
        System.out.println("Words: " + wordCount);
        System.out.println("Sentences: " + sentenceCount);
        System.out.println("Longest word: " + longestWord);
        System.out.println("Most common character: " + mostCommon + " (" + maxFreq + "
times)");
    }

    // Method to create word array and sort alphabetically
    public static String[] getWordsSorted(String text) {
        text = text.replaceAll("[^a-zA-Z ]", ""); // remove punctuation
        String[] words = text.split("\\s+");
        Arrays.sort(words, String.CASE_INSENSITIVE_ORDER);
        return words;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("=== TEXT PROCESSOR ===");
        System.out.print("Enter a paragraph: ");
        String input = scanner.nextLine();

        // Step 1: Clean input
        String cleaned = cleanInput(input);

        // Step 2: Analyze text
        System.out.println("\n--- TEXT ANALYSIS ---");
        analyzeText(cleaned);

        // Step 3: Show words in alphabetical order
        System.out.println("\n--- SORTED WORDS ---");
        String[] sortedWords = getWordsSorted(cleaned);
        for (String w : sortedWords) {
            System.out.print(w + " ");
        }
        System.out.println();

        // Step 4: Allow user to search for specific word
        System.out.print("\nEnter a word to search: ");
        String searchWord = scanner.nextLine();
        boolean found = Arrays.asList(sortedWords).contains(searchWord);
        if (found) {
            System.out.println("Word FOUND in text!");
        } else {
            System.out.println("Word NOT FOUND in text.");
        }

        scanner.close();
    }
}
```