PRACTICE PROBLEM 1: String Creation and Manipulation
Task: Create a program that demonstrates different ways to create strings and basic manipulation.

```
public class StringManipulation{
public static void main(String[] args) {
// TODO: Create the same string "Java Programming" using 3
different methods:
// 1. String literal
// 2. new String() constructor
// 3. Character array
// TODO: Compare the strings using == and .equals()
// Print the results and explain the difference
// TODO: Create a string with escape sequences that displays:
// Programming Quote:
// "Code is poetry" - Unknown
// Path: C:\Java\Projects
}
```

OUTPUT

```
public class StringManipulation {
    public static void main(String[] args) {
        // 1. String literal
        String strLiteral = "Java Programming";

        // 2. new String() constructor
        String strConstructor = new String("Java Programming");

        // 3. Character array
        char[] charArray = {'J', 'a', 'v', 'a', ' ', 'P', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g'};
        String strFromArray = new String(charArray);

        // Compare the strings using == and .equals()
        System.out.println("Comparing String literal with new String() constructor:");
        // The == operator compares memory addresses
        System.out.println("strLiteral == strConstructor: " + (strLiteral == strConstructor));
        // The .equals() method compares the actual content
        System.out.println("strLiteral.equals(strConstructor): " + strLiteral.equals(strConstructor));

        System.out.println("\nExplanation:");
        System.out.println("The == operator checks if two string variables refer to the exact same
object in memory.");
        System.out.println("Since 'strLiteral' is created in the string pool and 'strConstructor' is a
new object on the heap, they have different memory addresses, so == returns false.");
```

```
        System.out.println("The .equals() method checks if the content (the sequence of
characters) of the two strings is the same. Since both strings contain \"Java Programming\",
.equals() returns true.");

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

PRACTICE PROBLEM 2:
String Input and Processing
Task: Create a program that takes user input and processes it using various string methods.

OUTPUT

```
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("Please enter your full name (first and last): ");
        String fullName = scanner.nextLine();

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

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

        // Process the input
        // 1. Extract first and last name separately
        int spaceIndex = fullName.indexOf(" ");
        String firstName = fullName.substring(0, spaceIndex);
        String lastName = fullName.substring(spaceIndex + 1);

        // 2. Count total characters in the sentence (excluding spaces)
        int charCount = sentence.replace(" ", "").length();
```

```
        // 3. Convert programming language to uppercase
        String languageUpperCase = language.toUpperCase();

        // 4. Display a formatted summary
        System.out.println("\n--- Input Summary ---");
        System.out.println("First Name: " + firstName);
        System.out.println("Last Name: " + lastName);
        System.out.println("Favorite Language (in all caps): " + languageUpperCase);
        System.out.println("Your sentence has " + charCount + " characters (excluding spaces).");

        scanner.close();
    }
}
```

PRACTICE PROBLEM 3:
String Arrays and Methods
Task: Create a program that manages a list of student names using string arrays and methods.
OUTPUT

```
public class StringArrays {
    // Finds and returns the longest name in a string array
    public static String findLongestName(String[] names) {
        String longestName = "";
        for (String name : names) {
            if (name.length() > longestName.length()) {
                longestName = name;
            }
        }
        return longestName;
    }

    // Counts how many names start with a given letter (case-insensitive)
    public static int countNamesStartingWith(String[] names, char letter) {
        int count = 0;
        // Convert the letter to a string for easier comparison
        String letterStr = String.valueOf(letter).toLowerCase();
        for (String name : names) {
            if (name.toLowerCase().startsWith(letterStr)) {
                count++;
            }
        }
        return count;
    }
```

```
    // Formats all names to "Last, First" format
    public static String[] formatNames(String[] names) {
        String[] formattedNames = new String[names.length];
        for (int i = 0; i < names.length; i++) {
            String name = names[i];
            int spaceIndex = name.indexOf(" ");
            String firstName = name.substring(0, spaceIndex);
            String lastName = name.substring(spaceIndex + 1);
            formattedNames[i] = lastName + ", " + firstName;
        }
        return formattedNames;
    }

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

        // Test all your methods and display results
        System.out.println("Original list of names: " + java.util.Arrays.toString(students));

        // Test findLongestName()
        String longest = findLongestName(students);
        System.out.println("The longest name is: " + longest);

        // Test countNamesStartingWith()
        char targetLetter = 'A';
        int count = countNamesStartingWith(students, targetLetter);
        System.out.println("Number of names starting with '" + targetLetter + "': " + count);

        // Test formatNames()
        String[] formattedList = formatNames(students);
        System.out.println("Formatted names (Last, First): " +
java.util.Arrays.toString(formattedList));
    }
}
```

PRACTICE PROBLEM 4:

Complete String Application (10 minutes)

Task: Create a simple text processor that combines all concepts learned.

OUTPUT

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

```
public class TextProcessor {
    // Method to clean and validate input
    public static String cleanInput(String input) {
        // Remove leading/trailing spaces and multiple spaces
        String cleaned = input.trim().replaceAll("\\s+", " ");
        // Convert to sentence case (first letter capitalized)
        if (cleaned.length() > 0) {
            cleaned = cleaned.substring(0, 1).toUpperCase() + cleaned.substring(1).toLowerCase();
        }
        return cleaned;
    }

    // Method to analyze text
    public static void analyzeText(String text) {
        // Count words, sentences, characters
        String[] words = text.split("\\s+");
        int wordCount = words.length;
        int charCount = text.replace(" ", "").length();
        // A simple way to count sentences (assuming they end with a common punctuation mark)
        int sentenceCount = 0;
        for (char c : text.toCharArray()) {
            if (c == '.' || c == '?' || c == '!') {
                sentenceCount++;
            }
        }
        if (sentenceCount == 0 && text.length() > 0) {
            sentenceCount = 1; // Assume at least one sentence
        }

        // Find longest word
        String longestWord = "";
        for (String word : words) {
            String cleanWord = word.replaceAll("[^a-zA-Z]", "");
            if (cleanWord.length() > longestWord.length()) {
                longestWord = cleanWord;
            }
        }

        // Find most common character
        Map<Character, Integer> charMap = new HashMap<>();
        char mostCommonChar = ' ';
        int maxCount = 0;
        for (char c : text.toLowerCase().toCharArray()) {
```

```
            if (Character.isLetter(c)) {
                charMap.put(c, charMap.getOrDefault(c, 0) + 1);
                if (charMap.get(c) > maxCount) {
                    maxCount = charMap.get(c);
                    mostCommonChar = c;
                }
            }
        }

        // Display statistics
        System.out.println("\n--- Text Analysis ---");
        System.out.println("Word Count: " + wordCount);
        System.out.println("Character Count: " + charCount);
        System.out.println("Sentence Count: " + sentenceCount);
        System.out.println("Longest Word: " + longestWord);
        System.out.println("Most Common Character: '" + mostCommonChar + "' (appears " +
maxCount + " times)");
    }

    // Method to create word array and sort alphabetically
    public static String[] getWordsSorted(String text) {
        // Split text into words, remove punctuation, sort
        String[] words = text.split("\\s+");
        for (int i = 0; i < words.length; i++) {
            words[i] = words[i].replaceAll("[^a-zA-Z]", "").toLowerCase();
        }
        Arrays.sort(words);
        return words;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("=== TEXT PROCESSOR ===");
        System.out.println("Enter a paragraph of text below:");

        // Ask user for a paragraph of text
        String userText = scanner.nextLine();

        // Cleans and validates the input
        String cleanedText = cleanInput(userText);
        System.out.println("\nCleaned Text: " + cleanedText);

        // Analyzes the text (word count, character count, etc.)
        analyzeText(cleanedText);
```

```
        // Shows the words in alphabetical order
        String[] sortedWords = getWordsSorted(cleanedText);
        System.out.println("\nWords in alphabetical order: " + Arrays.toString(sortedWords));

        // Allows user to search for specific words
        System.out.print("\nEnter a word to search for: ");
        String searchWord = scanner.nextLine().toLowerCase();
        int foundCount = 0;
        for (String word : sortedWords) {
            if (word.equals(searchWord)) {
                foundCount++;
            }
        }
        System.out.println("The word '" + searchWord + "' was found " + foundCount + " time(s).");

        scanner.close();
    }
}
```