# Week 2 – S2 – Practise Problem Solution

**Name: Ramesh Harisabapathi Chettiar**

**Roll Number: RA2411030010263**

**Course: Networking and Communications**

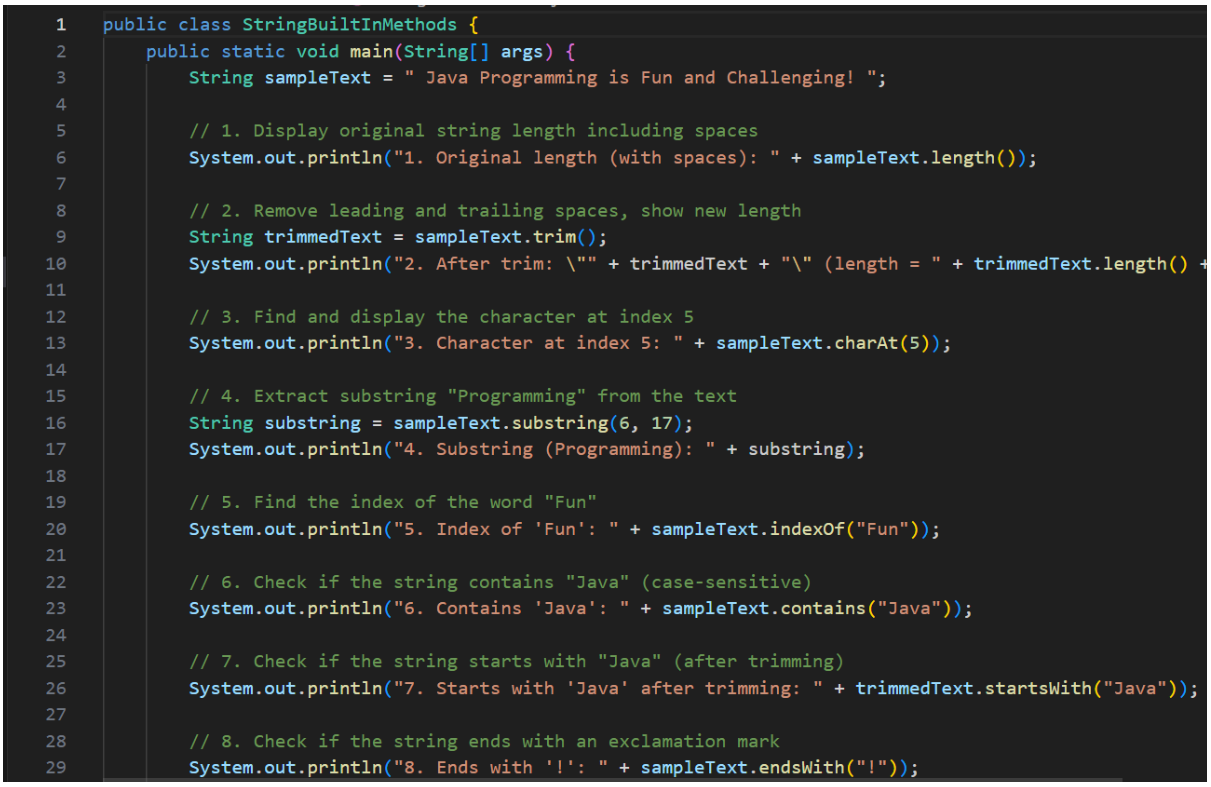
**Semester: 3**

**Date of Submission: 23/08/2025**

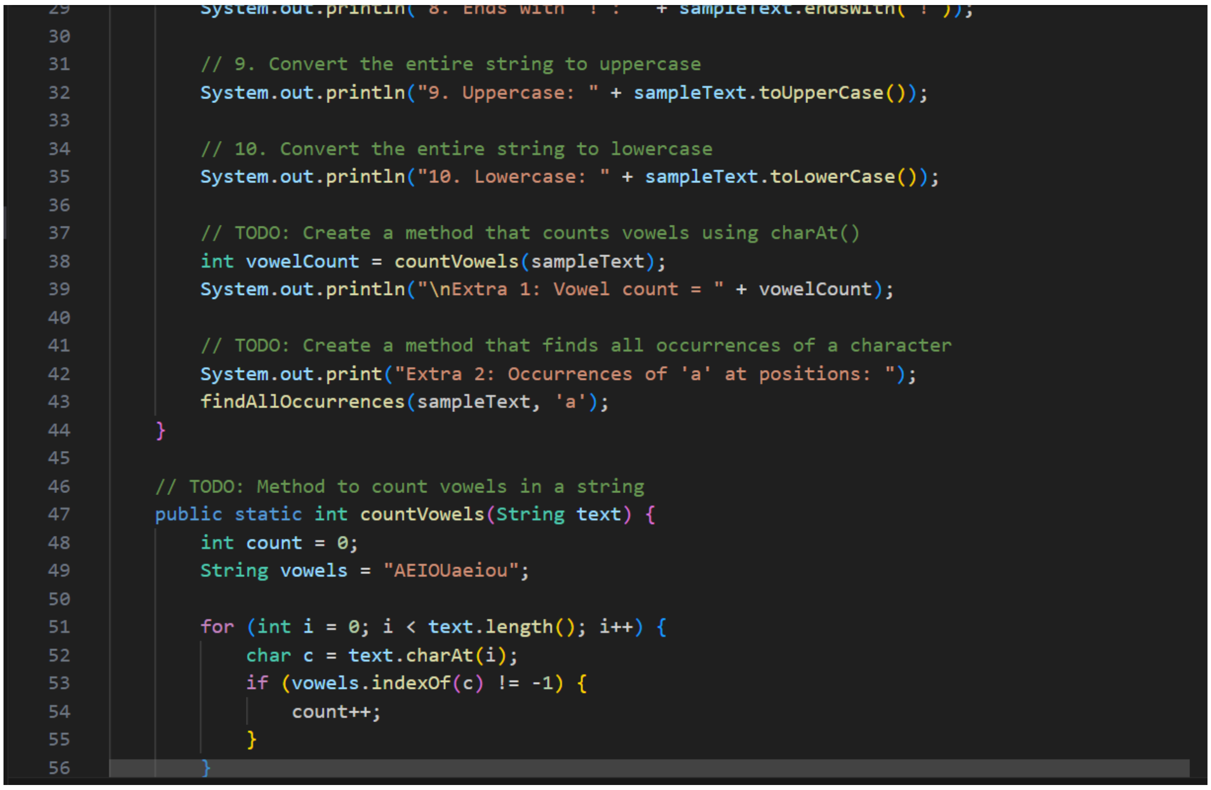
**PRACTICE PROBLEM 1: (Any 4)**

**Built-In String Methods - Basic Operations**

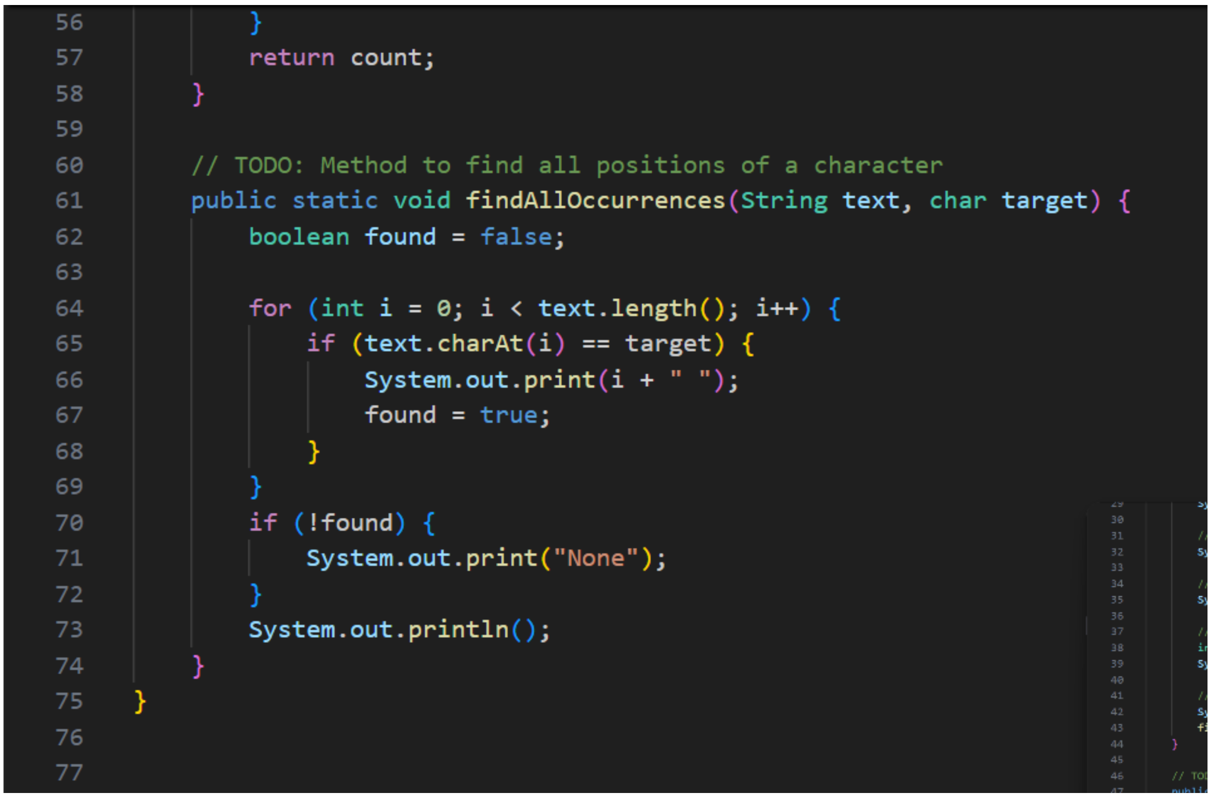
**Task: Create a program that demonstrates common String methods for text analysis and manipulation.**



```java
public class StringBuiltInMethods {
    public static void main(String[] args) {
        String sampleText = " Java Programming is Fun and Challenging! ";

        // 1. Display original string length including spaces
        System.out.println("1. Original length (with spaces): " + sampleText.length());

        // 2. Remove leading and trailing spaces, show new length
        String trimmedText = sampleText.trim();
        System.out.println("2. After trim: \"" + trimmedText + "\" (length = " + trimmedText.length() +

        // 3. Find and display the character at index 5
        System.out.println("3. Character at index 5: " + sampleText.charAt(5));

        // 4. Extract substring "Programming" from the text
        String substring = sampleText.substring(6, 17);
        System.out.println("4. Substring (Programming): " + substring);

        // 5. Find the index of the word "Fun"
        System.out.println("5. Index of 'Fun': " + sampleText.indexOf("Fun"));

        // 6. Check if the string contains "Java" (case-sensitive)
        System.out.println("6. Contains 'Java': " + sampleText.contains("Java"));

        // 7. Check if the string starts with "Java" (after trimming)
        System.out.println("7. Starts with 'Java' after trimming: " + trimmedText.startsWith("Java"));

        // 8. Check if the string ends with an exclamation mark
        System.out.println("8. Ends with '!': " + sampleText.endsWith("!"));
```

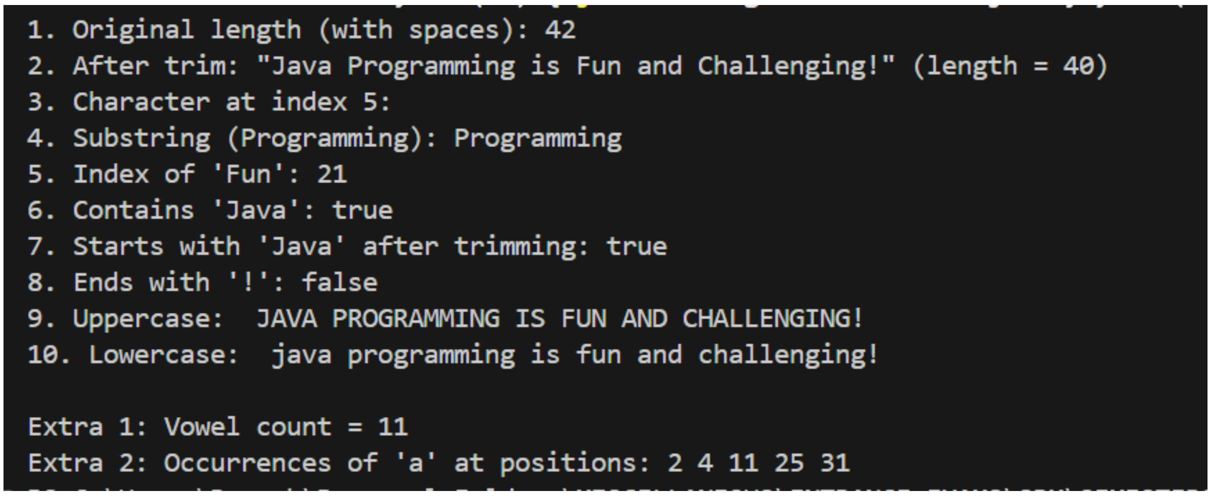


```java

        // 9. Convert the entire string to uppercase
        System.out.println("9. Uppercase: " + sampleText.toUpperCase());

        // 10. Convert the entire string to lowercase
        System.out.println("10. Lowercase: " + sampleText.toLowerCase());

        // TODO: Create a method that counts vowels using charAt()
        int vowelCount = countVowels(sampleText);
        System.out.println("\nExtra 1: Vowel count = " + vowelCount);

        // TODO: Create a method that finds all occurrences of a character
        System.out.print("Extra 2: Occurrences of 'a' at positions: ");
        findAllOccurrences(sampleText, 'a');
    }

    // TODO: Method to count vowels in a string
    public static int countVowels(String text) {
        int count = 0;
        String vowels = "AEIOUaeiou";

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (vowels.indexOf(c) != -1) {
                count++;
            }
        }
```



```java
        }
        return count;
    }

    // TODO: Method to find all positions of a character
    public static void findAllOccurrences(String text, char target) {
        boolean found = false;

        for (int i = 0; i < text.length(); i++) {
            if (text.charAt(i) == target) {
                System.out.print(i + " ");
                found = true;
            }
        }
        if (!found) {
            System.out.print("None");
        }
        System.out.println();
    }
}
```

**OUTPUT→**

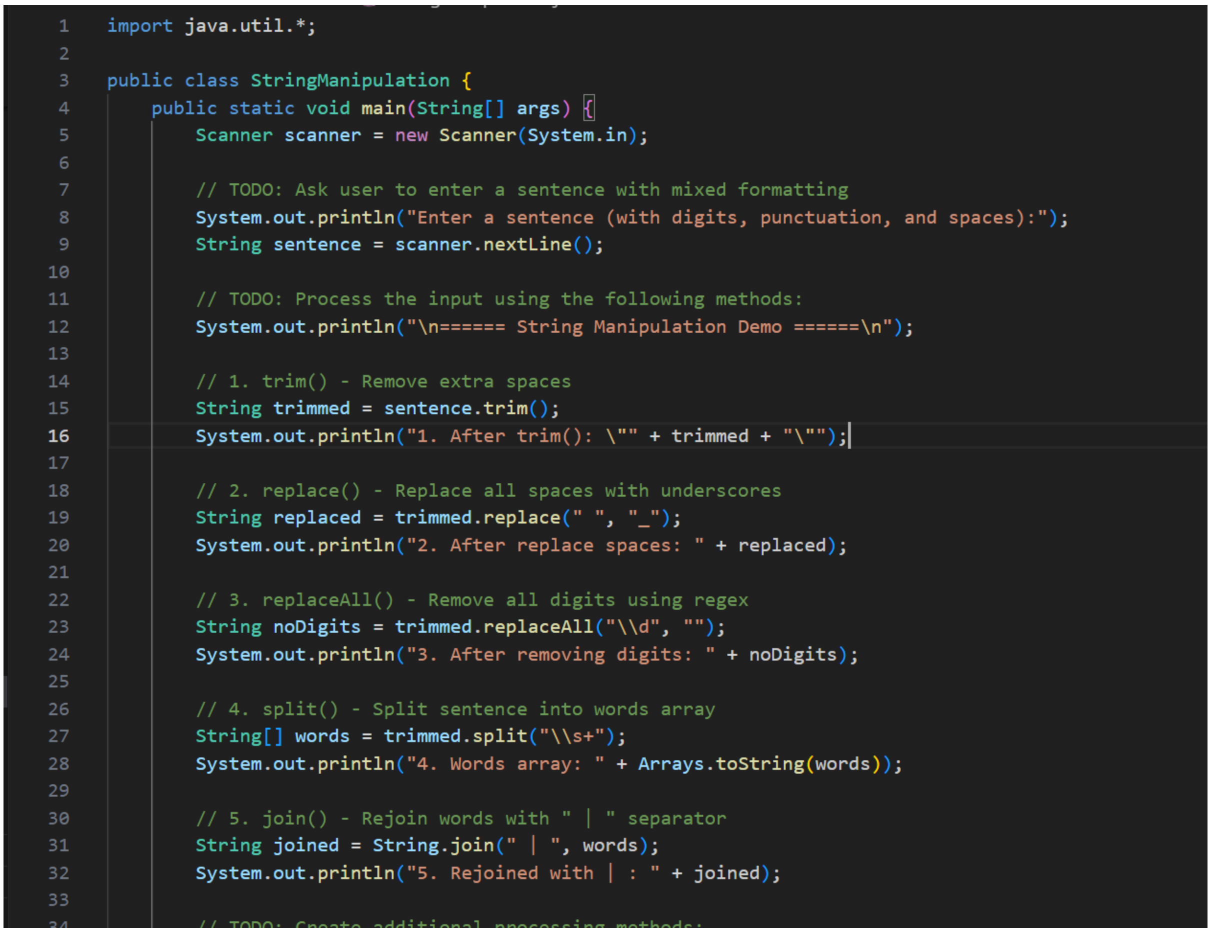


```
1. Original length (with spaces): 42
2. After trim: "Java Programming is Fun and Challenging!" (length = 40)
3. Character at index 5:
4. Substring (Programming): Programming
5. Index of 'Fun': 21
6. Contains 'Java': true
7. Starts with 'Java' after trimming: true
8. Ends with '!': false
9. Uppercase:  JAVA PROGRAMMING IS FUN AND CHALLENGING!
10. Lowercase:  java programming is fun and challenging!

Extra 1: Vowel count = 11
Extra 2: Occurrences of 'a' at positions: 2 4 11 25 31
```

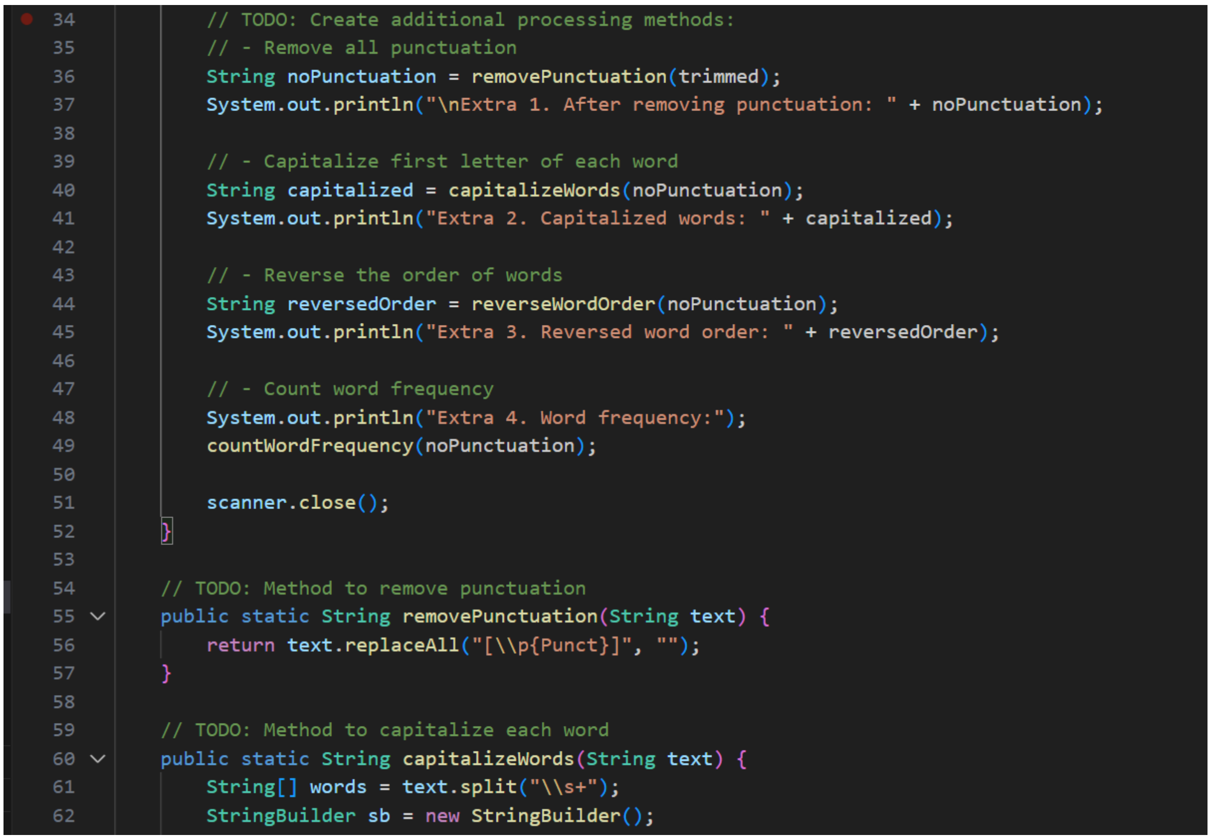
## 🛠 PRACTICE PROBLEM 2:

**String Manipulation Methods**

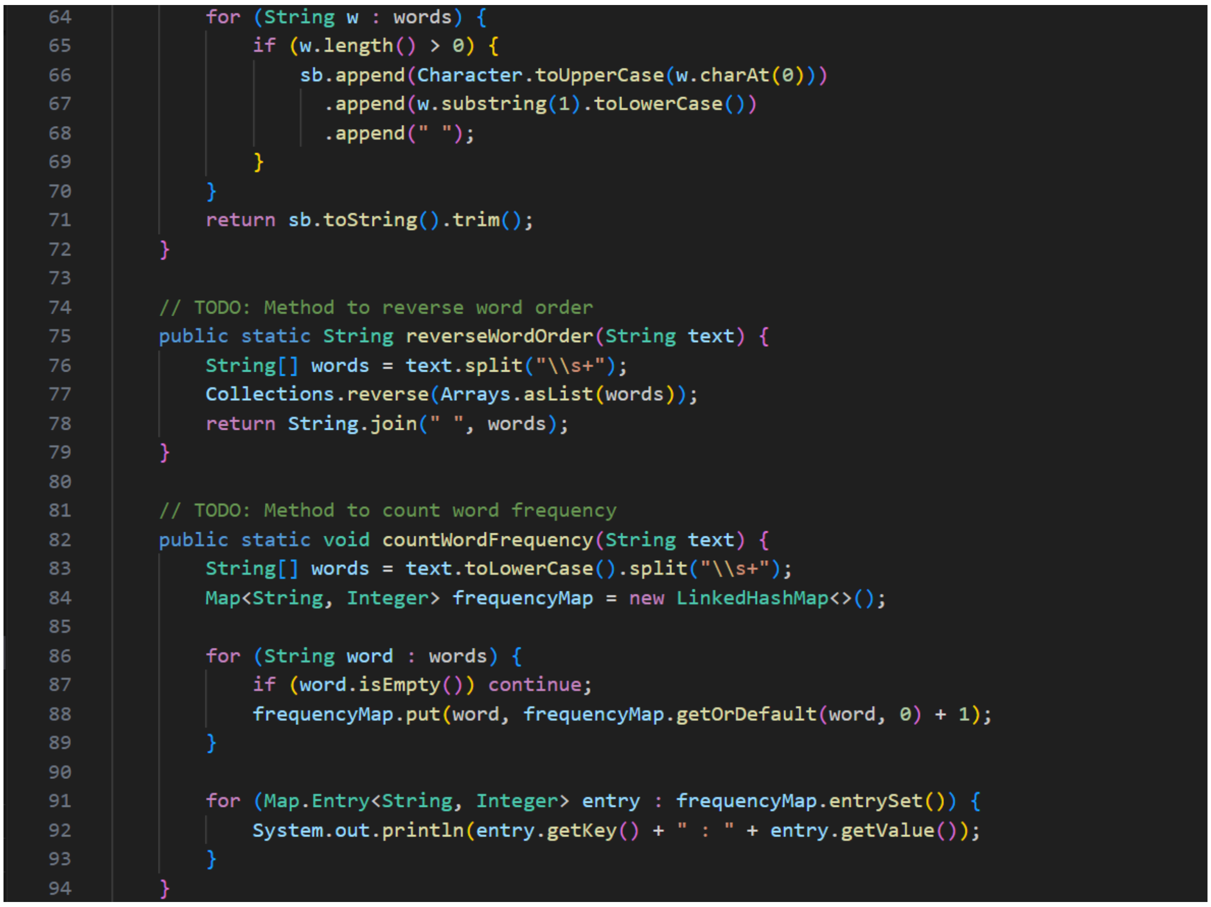
**Task: Create a text processing utility that uses various string manipulation methods.**



```java
import java.util.*;

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

        // TODO: Ask user to enter a sentence with mixed formatting
        System.out.println("Enter a sentence (with digits, punctuation, and spaces):");
        String sentence = scanner.nextLine();

        // TODO: Process the input using the following methods:
        System.out.println("\n====== String Manipulation Demo ======\n");

        // 1. trim() - Remove extra spaces
        String trimmed = sentence.trim();
        System.out.println("1. After trim(): \"" + trimmed + "\"");

        // 2. replace() - Replace all spaces with underscores
        String replaced = trimmed.replace(" ", "_");
        System.out.println("2. After replace spaces: " + replaced);

        // 3. replaceAll() - Remove all digits using regex
        String noDigits = trimmed.replaceAll("\\d", "");
        System.out.println("3. After removing digits: " + noDigits);

        // 4. split() - Split sentence into words array
        String[] words = trimmed.split("\\s+");
        System.out.println("4. Words array: " + Arrays.toString(words));

        // 5. join() - Rejoin words with " | " separator
        String joined = String.join(" | ", words);
        System.out.println("5. Rejoined with | : " + joined);

```

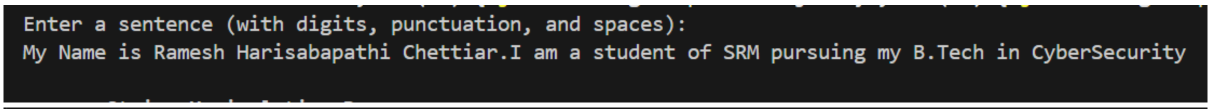


```java
        // TODO: Create additional processing methods:
        // - Remove all punctuation
        String noPunctuation = removePunctuation(trimmed);
        System.out.println("\nExtra 1. After removing punctuation: " + noPunctuation);

        // - Capitalize first letter of each word
        String capitalized = capitalizeWords(noPunctuation);
        System.out.println("Extra 2. Capitalized words: " + capitalized);

        // - Reverse the order of words
        String reversedOrder = reverseWordOrder(noPunctuation);
        System.out.println("Extra 3. Reversed word order: " + reversedOrder);

        // - Count word frequency
        System.out.println("Extra 4. Word frequency:");
        countWordFrequency(noPunctuation);

        scanner.close();
    }

    // TODO: Method to remove punctuation
    public static String removePunctuation(String text) {
        return text.replaceAll("[\\p{Punct}]", "");
    }

    // TODO: Method to capitalize each word
    public static String capitalizeWords(String text) {
        String[] words = text.split("\\s+");
        StringBuilder sb = new StringBuilder();
```

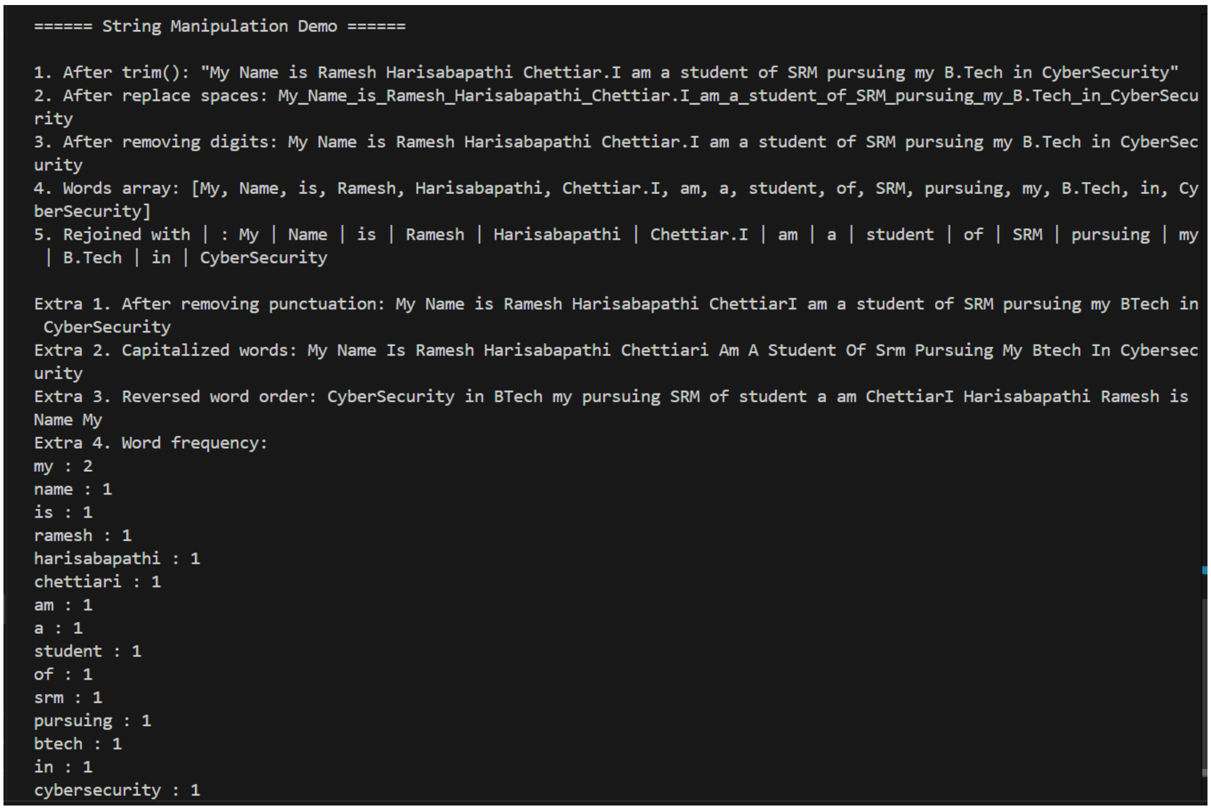


```java
        for (String w : words) {
            if (w.length() > 0) {
                sb.append(Character.toUpperCase(w.charAt(0)))
                  .append(w.substring(1).toLowerCase())
                  .append(" ");
            }
        }
        return sb.toString().trim();
    }

    // TODO: Method to reverse word order
    public static String reverseWordOrder(String text) {
        String[] words = text.split("\\s+");
        Collections.reverse(Arrays.asList(words));
        return String.join(" ", words);
    }

    // TODO: Method to count word frequency
    public static void countWordFrequency(String text) {
        String[] words = text.toLowerCase().split("\\s+");
        Map<String, Integer> frequencyMap = new LinkedHashMap<>();

        for (String word : words) {
            if (word.isEmpty()) continue;
            frequencyMap.put(word, frequencyMap.getOrDefault(word, 0) + 1);
        }

        for (Map.Entry<String, Integer> entry : frequencyMap.entrySet()) {
            System.out.println(entry.getKey() + " : " + entry.getValue());
        }
    }
```

**OUTPUT→**



```
Enter a sentence (with digits, punctuation, and spaces):
My Name is Ramesh Harisabapathi Chettiar.I am a student of SRM pursuing my B.Tech in CyberSecurity
```

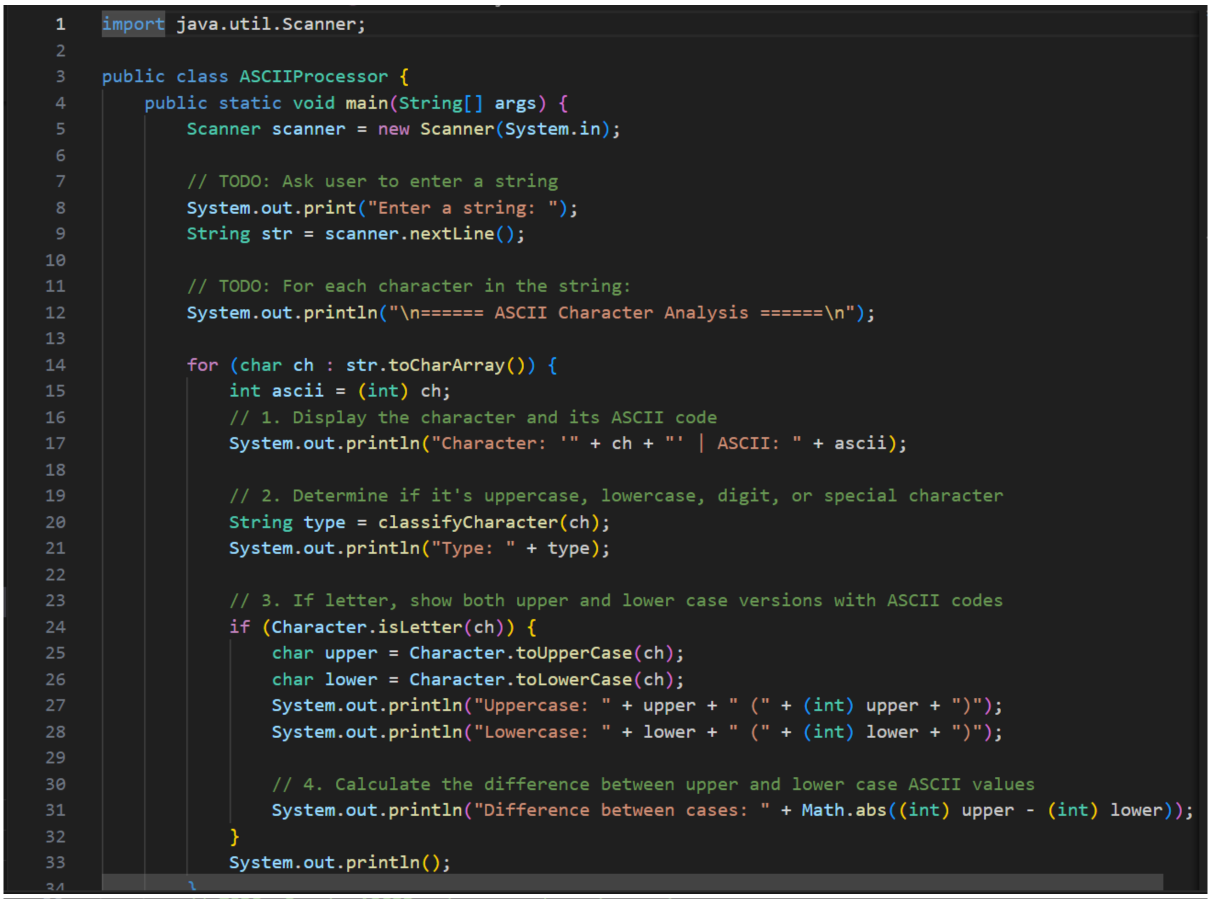


```
====== String Manipulation Demo ======

1. After trim(): "My Name is Ramesh Harisabapathi Chettiar.I am a student of SRM pursuing my B.Tech in CyberSecurity"
2. After replace spaces: My_Name_is_Ramesh_Harisabapathi_Chettiar.I_am_a_student_of_SRM_pursuing_my_B.Tech_in_CyberSecu
rity
3. After removing digits: My Name is Ramesh Harisabapathi Chettiar.I am a student of SRM pursuing my B.Tech in CyberSec
urity
4. Words array: [My, Name, is, Ramesh, Harisabapathi, Chettiar.I, am, a, student, of, SRM, pursuing, my, B.Tech, in, Cy
berSecurity]
5. Rejoined with | : My | Name | is | Ramesh | Harisabapathi | Chettiar.I | am | a | student | of | SRM | pursuing | my
 | B.Tech | in | CyberSecurity

Extra 1. After removing punctuation: My Name is Ramesh Harisabapathi ChettiarI am a student of SRM pursuing my BTech in
 CyberSecurity
Extra 2. Capitalized words: My Name Is Ramesh Harisabapathi Chettiari Am A Student Of Srm Pursuing My Btech In Cybersec
urity
Extra 3. Reversed word order: CyberSecurity in BTech my pursuing SRM of student a am ChettiarI Harisabapathi Ramesh is
Name My
Extra 4. Word frequency:
my : 2
name : 1
is : 1
ramesh : 1
harisabapathi : 1
chettiari : 1
am : 1
a : 1
student : 1
of : 1
srm : 1
pursuing : 1
btech : 1
in : 1
cybersecurity : 1
```

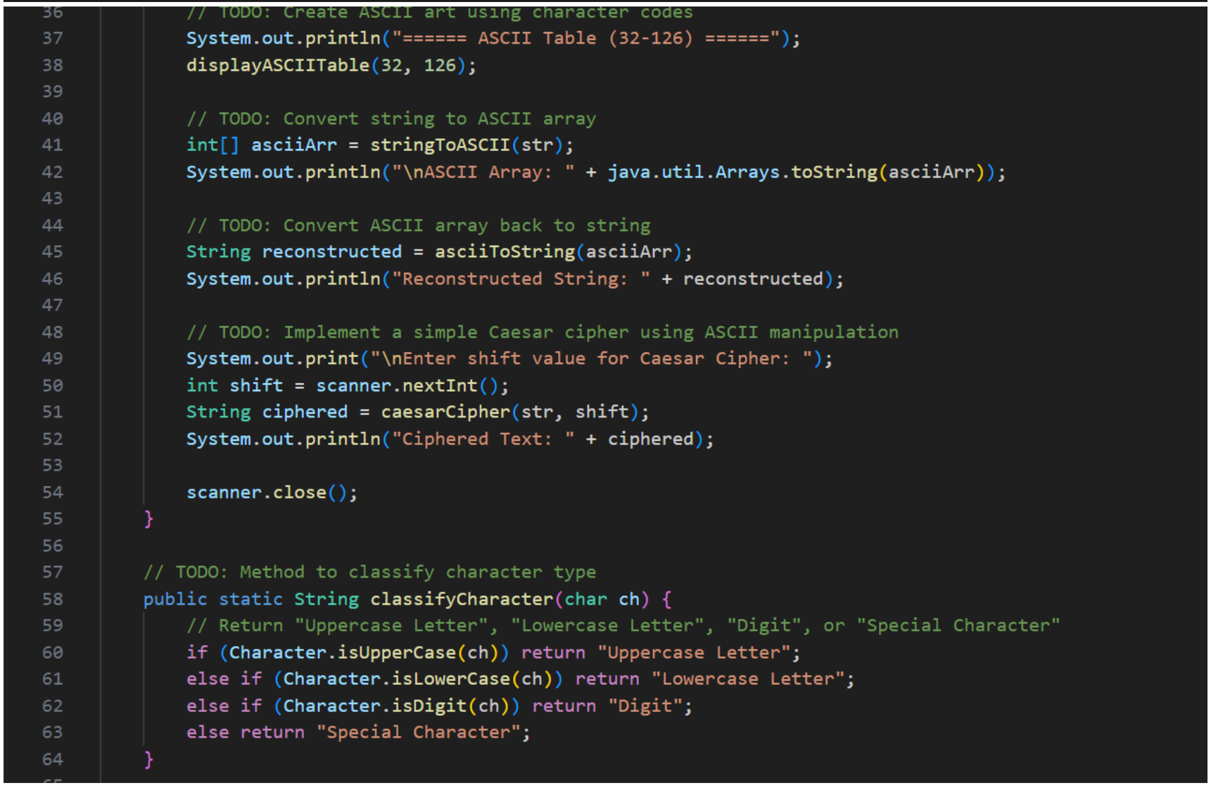
## 🛠 PRACTICE PROBLEM 3:

**ASCII Codes and Character Conversion**

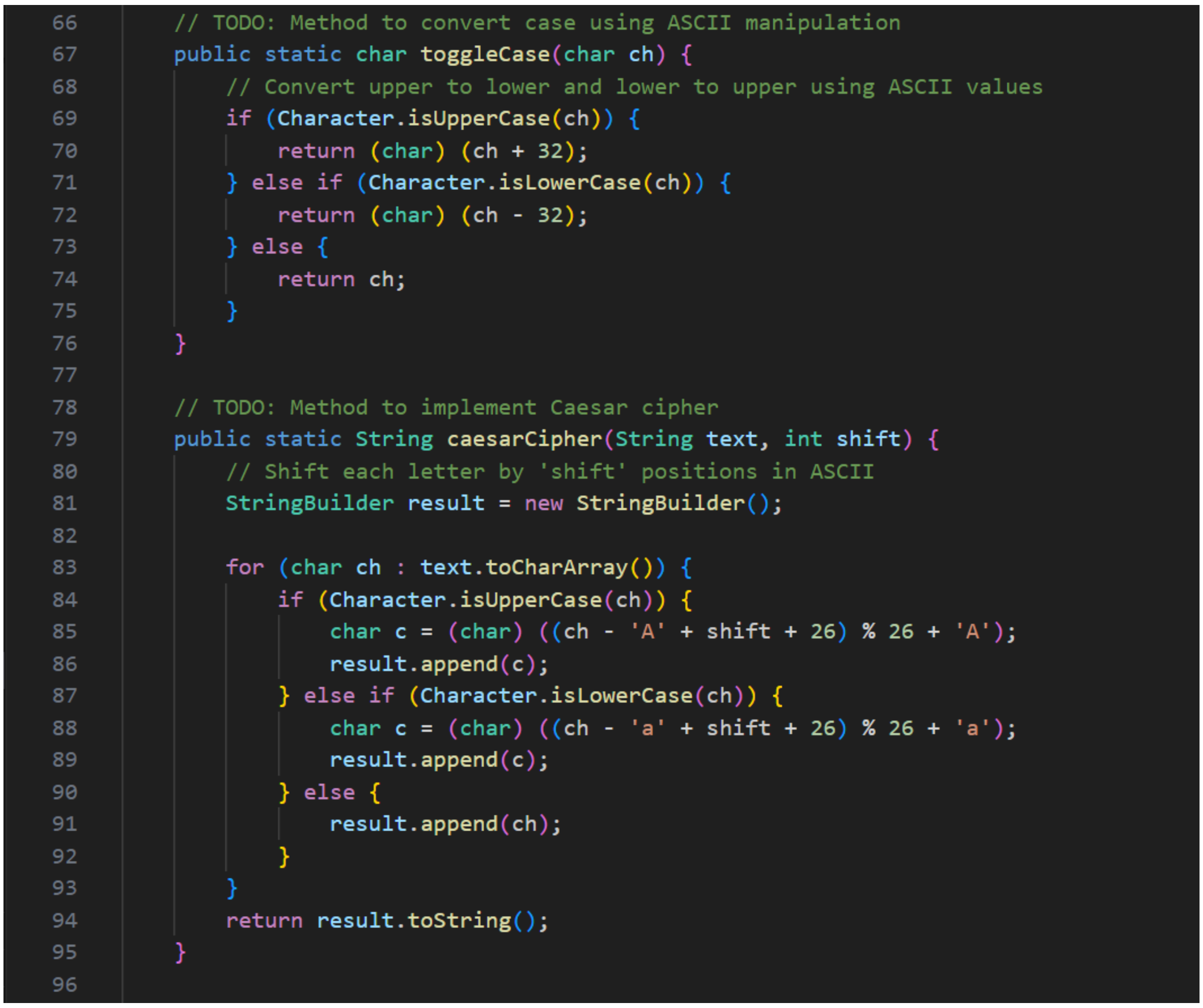
**Task: Create a program that demonstrates ASCII character manipulation and conversion.**

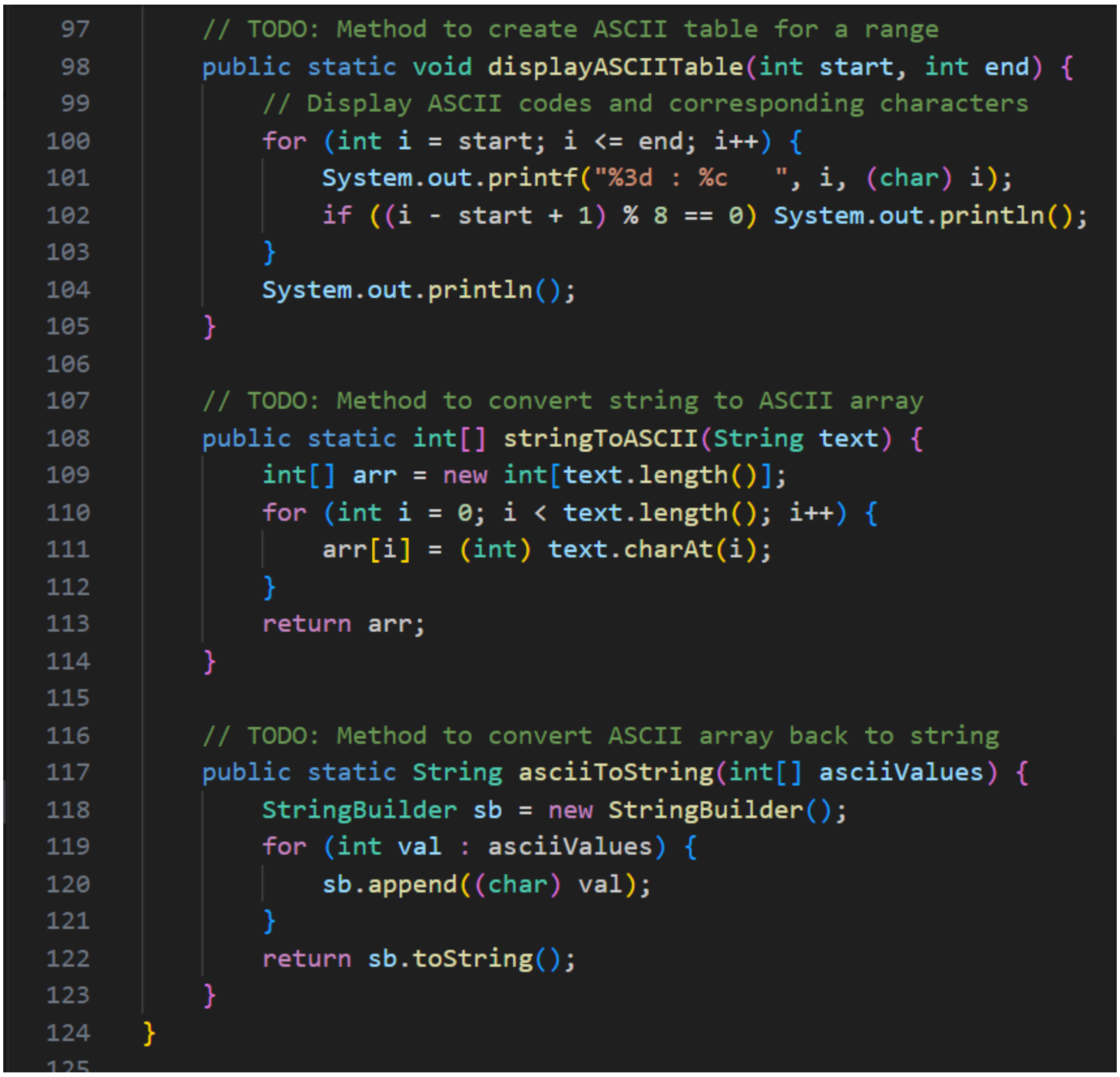


```java
import java.util.Scanner;

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

        // TODO: Ask user to enter a string
        System.out.print("Enter a string: ");
        String str = scanner.nextLine();

        // TODO: For each character in the string:
        System.out.println("\n====== ASCII Character Analysis ======\n");

        for (char ch : str.toCharArray()) {
            int ascii = (int) ch;
            // 1. Display the character and its ASCII code
            System.out.println("Character: '" + ch + "' | ASCII: " + ascii);

            // 2. Determine if it's uppercase, lowercase, digit, or special character
            String type = classifyCharacter(ch);
            System.out.println("Type: " + type);

            // 3. If letter, show both upper and lower case versions with ASCII codes
            if (Character.isLetter(ch)) {
                char upper = Character.toUpperCase(ch);
                char lower = Character.toLowerCase(ch);
                System.out.println("Uppercase: " + upper + " (" + (int) upper + ")");
                System.out.println("Lowercase: " + lower + " (" + (int) lower + ")");

                // 4. Calculate the difference between upper and lower case ASCII values
                System.out.println("Difference between cases: " + Math.abs((int) upper - (int) lower));
            }
            System.out.println();
        }
```



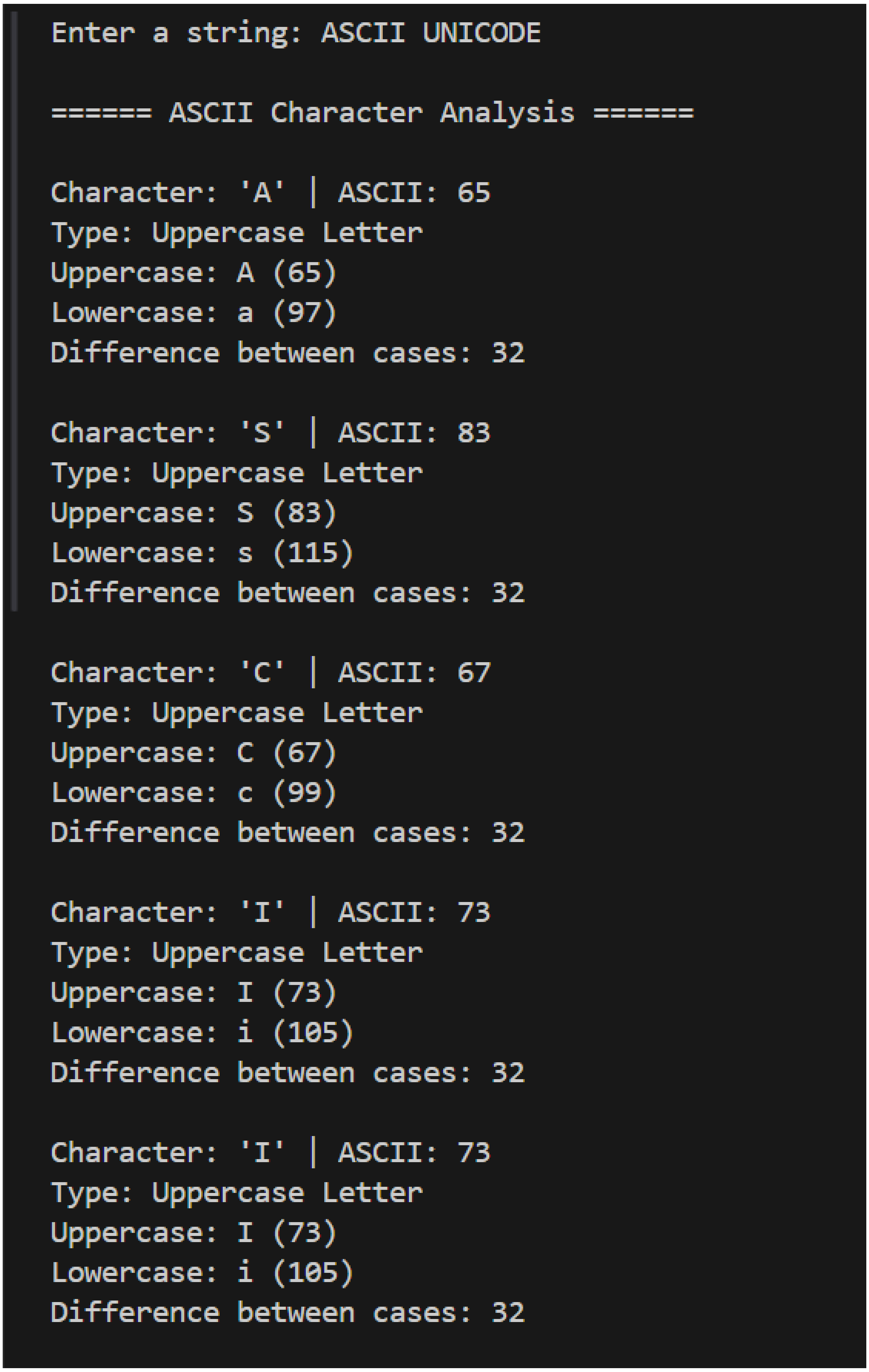
```java
        // TODO: Create ASCII art using character codes
        System.out.println("====== ASCII Table (32-126) ======");
        displayASCIITable(32, 126);

        // TODO: Convert string to ASCII array
        int[] asciiArr = stringToASCII(str);
        System.out.println("\nASCII Array: " + java.util.Arrays.toString(asciiArr));

        // TODO: Convert ASCII array back to string
        String reconstructed = asciiToString(asciiArr);
        System.out.println("Reconstructed String: " + reconstructed);

        // TODO: Implement a simple Caesar cipher using ASCII manipulation
        System.out.print("\nEnter shift value for Caesar Cipher: ");
        int shift = scanner.nextInt();
        String ciphered = caesarCipher(str, shift);
        System.out.println("Ciphered Text: " + ciphered);

        scanner.close();
    }

    // TODO: Method to classify character type
    public static String classifyCharacter(char ch) {
        // Return "Uppercase Letter", "Lowercase Letter", "Digit", or "Special Character"
        if (Character.isUpperCase(ch)) return "Uppercase Letter";
        else if (Character.isLowerCase(ch)) return "Lowercase Letter";
        else if (Character.isDigit(ch)) return "Digit";
        else return "Special Character";
    }
```


```java
    // TODO: Method to convert case using ASCII manipulation
    public static char toggleCase(char ch) {
        // Convert upper to lower and lower to upper using ASCII values
        if (Character.isUpperCase(ch)) {
            return (char) (ch + 32);
        } else if (Character.isLowerCase(ch)) {
            return (char) (ch - 32);
        } else {
            return ch;
        }
    }

    // TODO: Method to implement Caesar cipher
    public static String caesarCipher(String text, int shift) {
        // Shift each letter by 'shift' positions in ASCII
        StringBuilder result = new StringBuilder();

        for (char ch : text.toCharArray()) {
            if (Character.isUpperCase(ch)) {
                char c = (char) ((ch - 'A' + shift + 26) % 26 + 'A');
                result.append(c);
            } else if (Character.isLowerCase(ch)) {
                char c = (char) ((ch - 'a' + shift + 26) % 26 + 'a');
                result.append(c);
            } else {
                result.append(ch);
            }
        }
        return result.toString();
    }

```

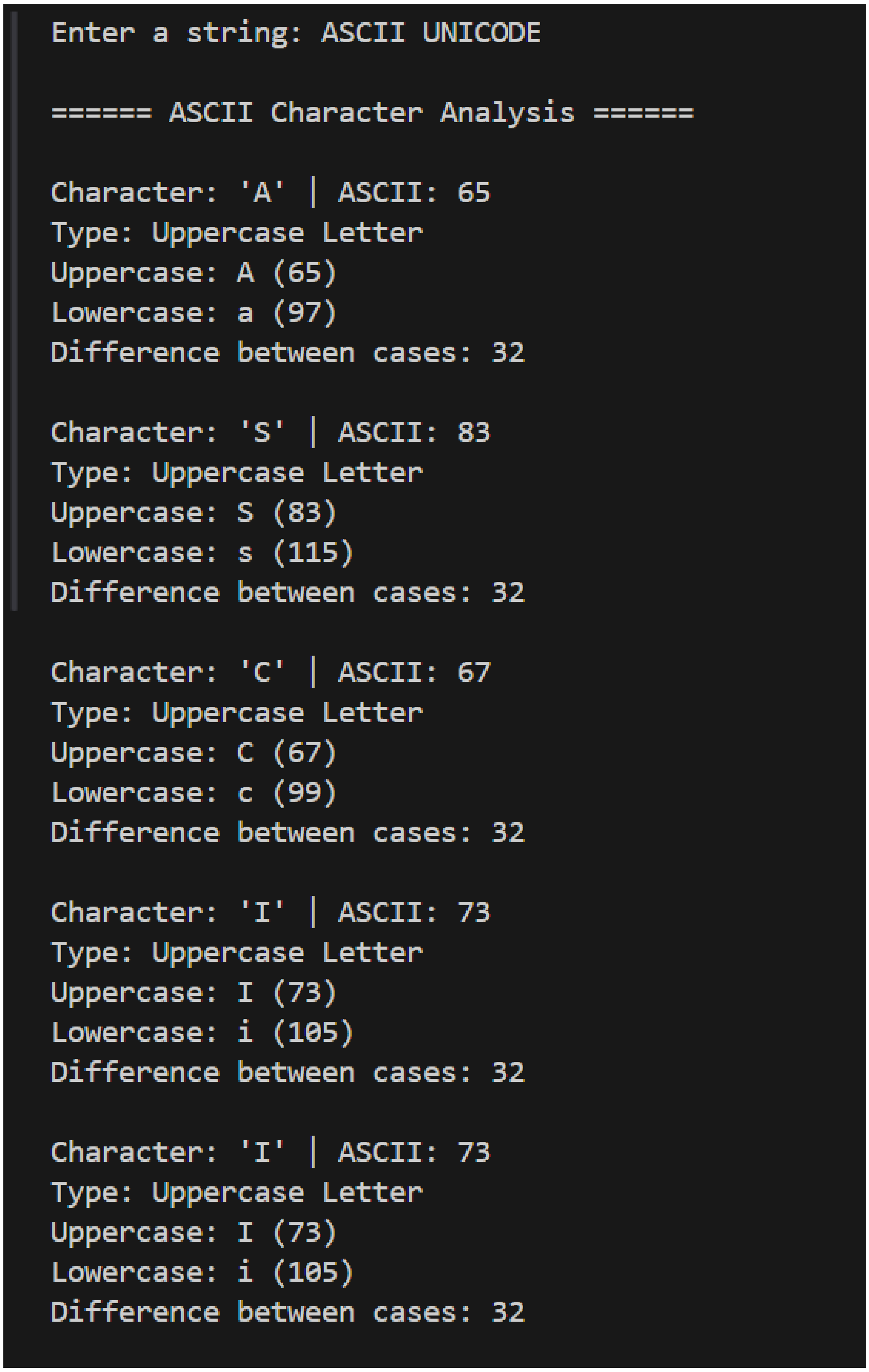


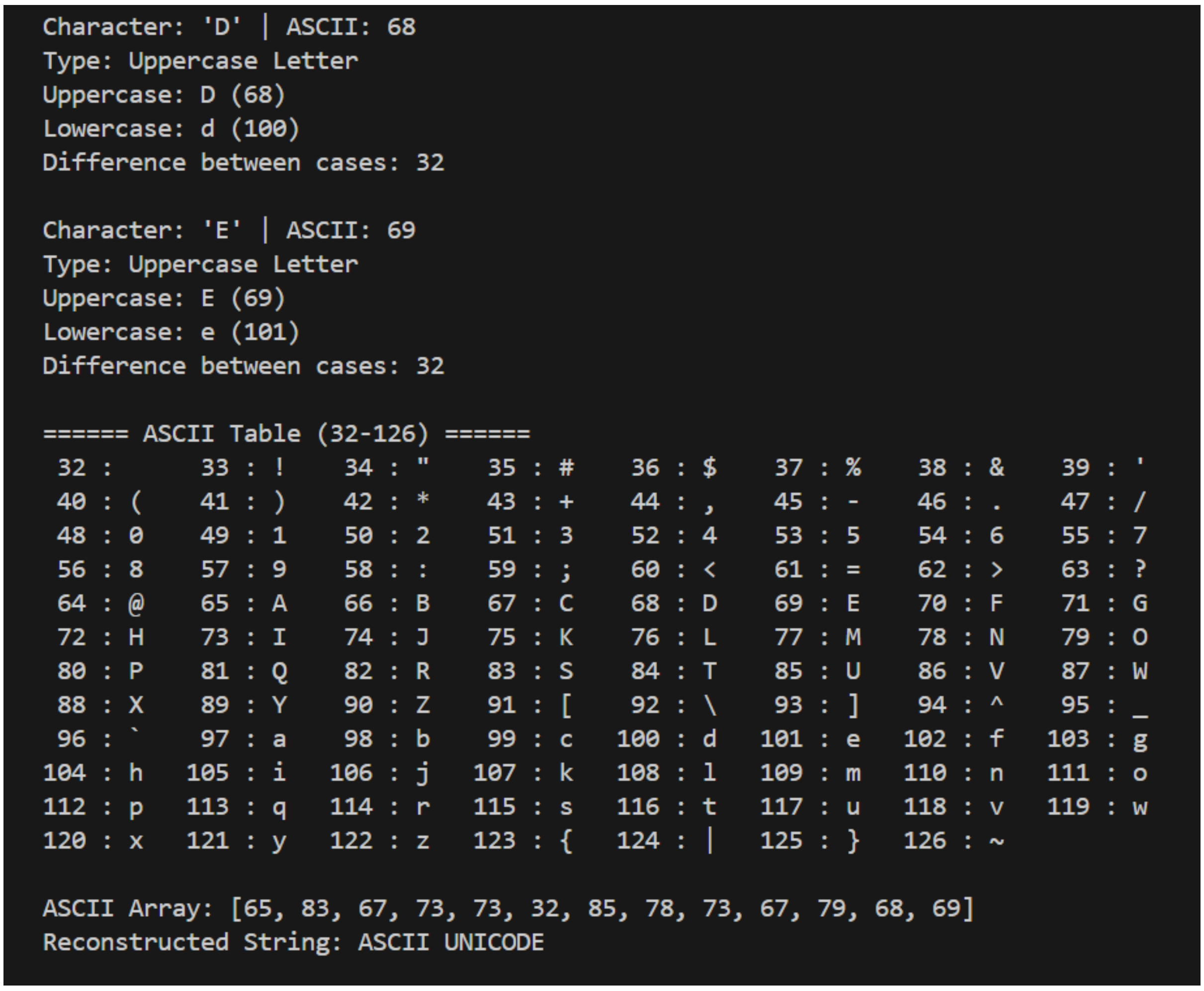
```java
    // TODO: Method to create ASCII table for a range
    public static void displayASCIITable(int start, int end) {
        // Display ASCII codes and corresponding characters
        for (int i = start; i <= end; i++) {
            System.out.printf("%3d : %c   ", i, (char) i);
            if ((i - start + 1) % 8 == 0) System.out.println();
        }
        System.out.println();
    }

    // TODO: Method to convert string to ASCII array
    public static int[] stringToASCII(String text) {
        int[] arr = new int[text.length()];
        for (int i = 0; i < text.length(); i++) {
            arr[i] = (int) text.charAt(i);
        }
        return arr;
    }

    // TODO: Method to convert ASCII array back to string
    public static String asciiToString(int[] asciiValues) {
        StringBuilder sb = new StringBuilder();
        for (int val : asciiValues) {
            sb.append((char) val);
        }
        return sb.toString();
    }
}
```

OUTPUT→



```
Enter a string: ASCII UNICODE

====== ASCII Character Analysis ======

Character: 'A' | ASCII: 65
Type: Uppercase Letter
Uppercase: A (65)
Lowercase: a (97)
Difference between cases: 32

Character: 'S' | ASCII: 83
Type: Uppercase Letter
Uppercase: S (83)
Lowercase: s (115)
Difference between cases: 32

Character: 'C' | ASCII: 67
Type: Uppercase Letter
Uppercase: C (67)
Lowercase: c (99)
Difference between cases: 32

Character: 'I' | ASCII: 73
Type: Uppercase Letter
Uppercase: I (73)
Lowercase: i (105)
Difference between cases: 32

Character: 'I' | ASCII: 73
Type: Uppercase Letter
Uppercase: I (73)
Lowercase: i (105)
Difference between cases: 32
```


```
Enter a string: ASCII UNICODE

====== ASCII Character Analysis ======

Character: 'A' | ASCII: 65
Type: Uppercase Letter
Uppercase: A (65)
Lowercase: a (97)
Difference between cases: 32

Character: 'S' | ASCII: 83
Type: Uppercase Letter
Uppercase: S (83)
Lowercase: s (115)
Difference between cases: 32

Character: 'C' | ASCII: 67
Type: Uppercase Letter
Uppercase: C (67)
Lowercase: c (99)
Difference between cases: 32

Character: 'I' | ASCII: 73
Type: Uppercase Letter
Uppercase: I (73)
Lowercase: i (105)
Difference between cases: 32

Character: 'I' | ASCII: 73
Type: Uppercase Letter
Uppercase: I (73)
Lowercase: i (105)
Difference between cases: 32
```

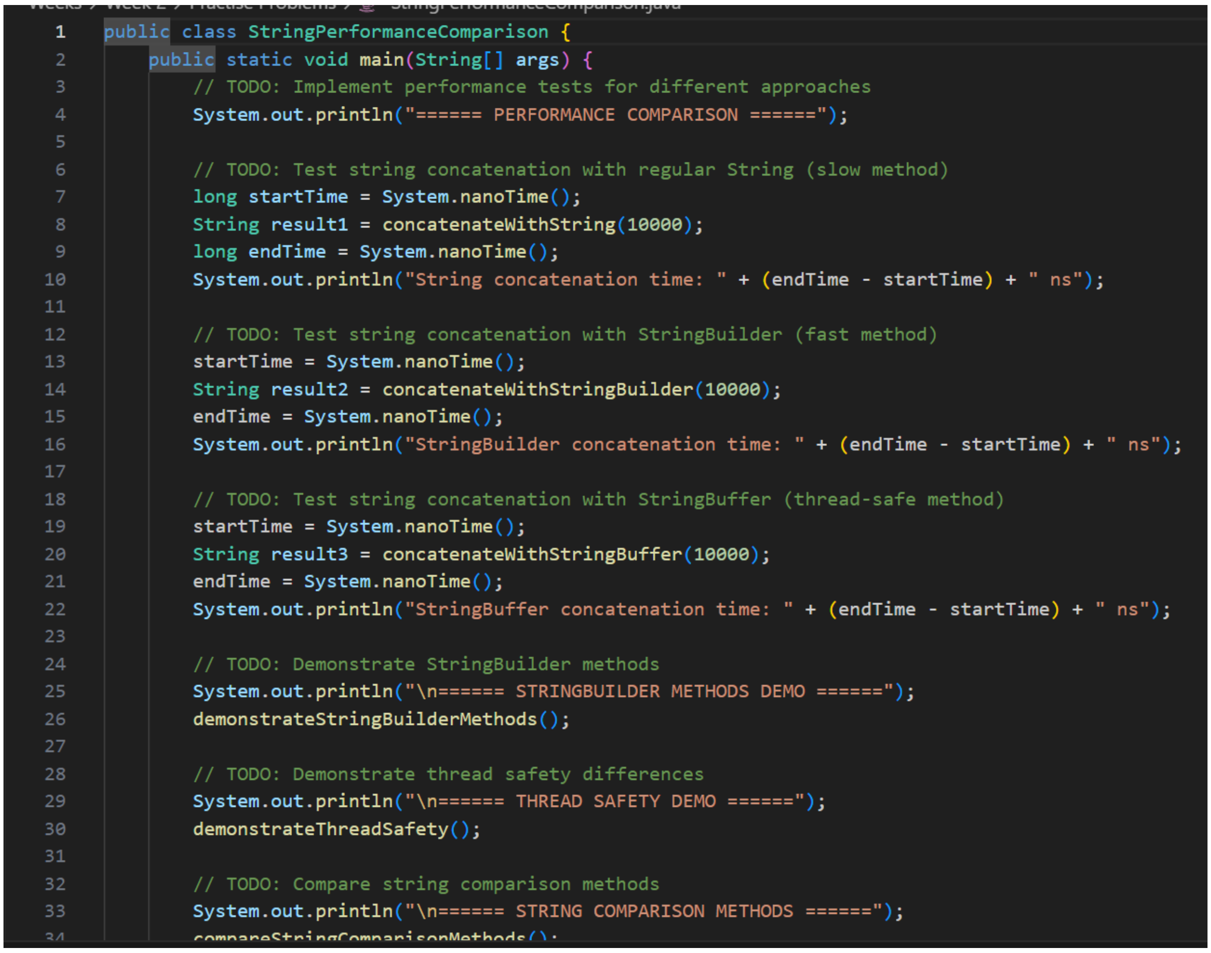


```
Character: 'D' | ASCII: 68
Type: Uppercase Letter
Uppercase: D (68)
Lowercase: d (100)
Difference between cases: 32

Character: 'E' | ASCII: 69
Type: Uppercase Letter
Uppercase: E (69)
Lowercase: e (101)
Difference between cases: 32

====== ASCII Table (32-126) ======
 32 :      33 : !    34 : "    35 : #    36 : $    37 : %    38 : &    39 : '
 40 : (    41 : )    42 : *    43 : +    44 : ,    45 : -    46 : .    47 : /
 48 : 0    49 : 1    50 : 2    51 : 3    52 : 4    53 : 5    54 : 6    55 : 7
 56 : 8    57 : 9    58 : :    59 : ;    60 : <    61 : =    62 : >    63 : ?
 64 : @    65 : A    66 : B    67 : C    68 : D    69 : E    70 : F    71 : G
 72 : H    73 : I    74 : J    75 : K    76 : L    77 : M    78 : N    79 : O
 80 : P    81 : Q    82 : R    83 : S    84 : T    85 : U    86 : V    87 : W
 88 : X    89 : Y    90 : Z    91 : [    92 : \    93 : ]    94 : ^    95 : _
 96 : `    97 : a    98 : b    99 : c   100 : d   101 : e   102 : f   103 : g
104 : h   105 : i   106 : j   107 : k   108 : l   109 : m   110 : n   111 : o
112 : p   113 : q   114 : r   115 : s   116 : t   117 : u   118 : v   119 : w
120 : x   121 : y   122 : z   123 : {   124 : |   125 : }   126 : ~

ASCII Array: [65, 83, 67, 73, 73, 32, 85, 78, 73, 67, 79, 68, 69]
Reconstructed String: ASCII UNICODE
```

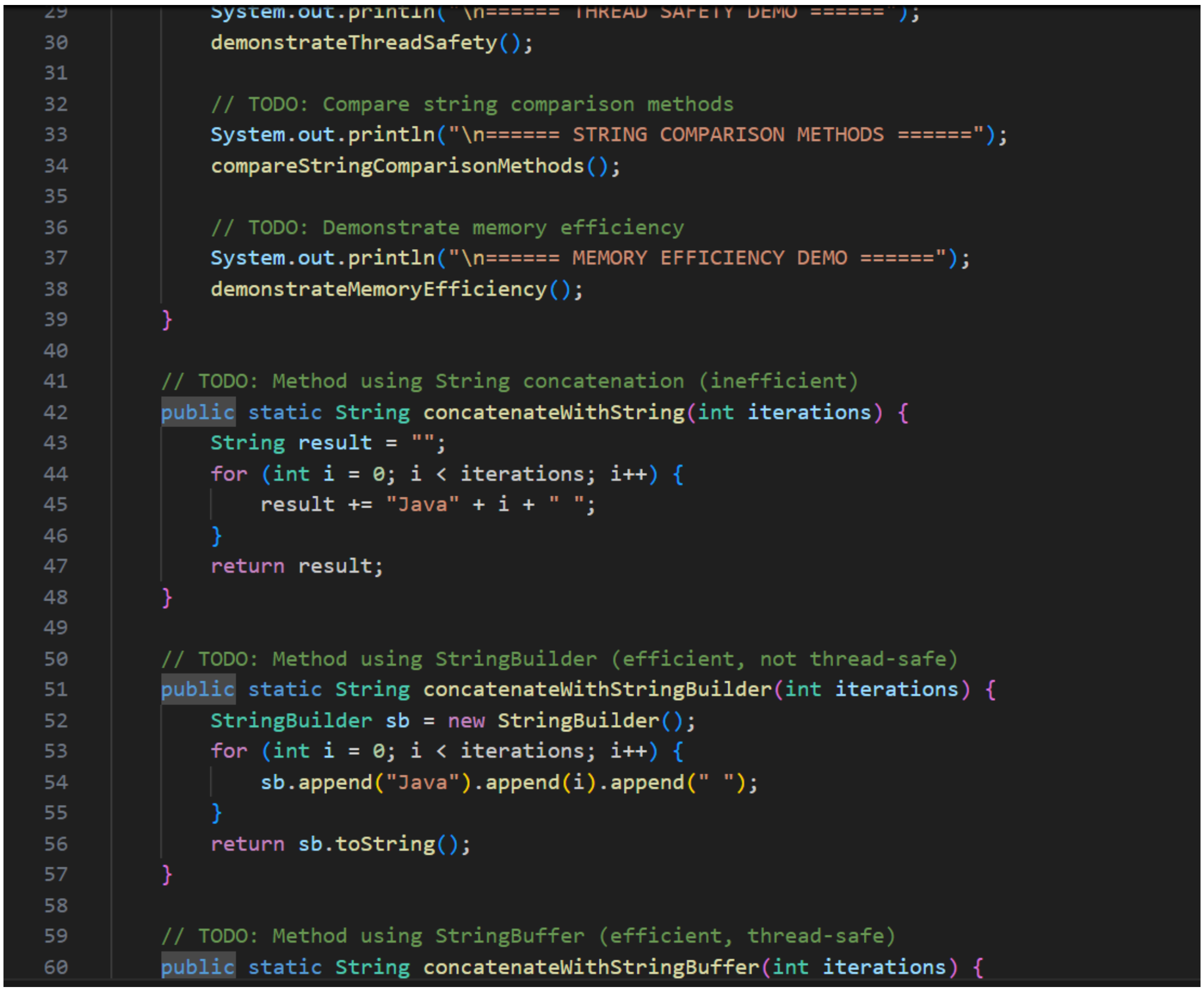
## 🛠 PRACTICE PROBLEM 4:

**StringBuilder, StringBuffer, and Performance**

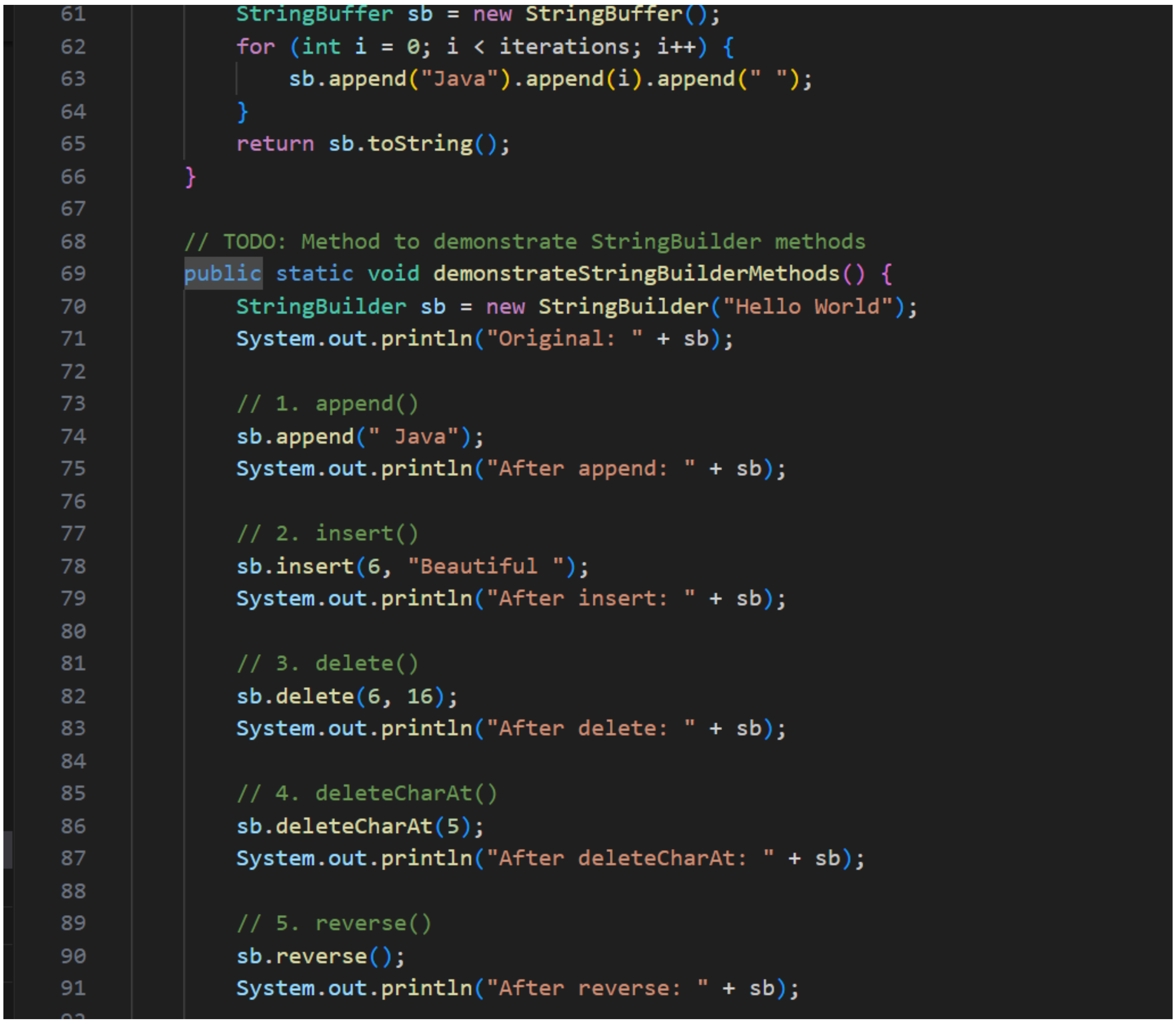
**Task: Create a performance comparison program that demonstrates the differences between String, StringBuilder, and StringBuffer.**



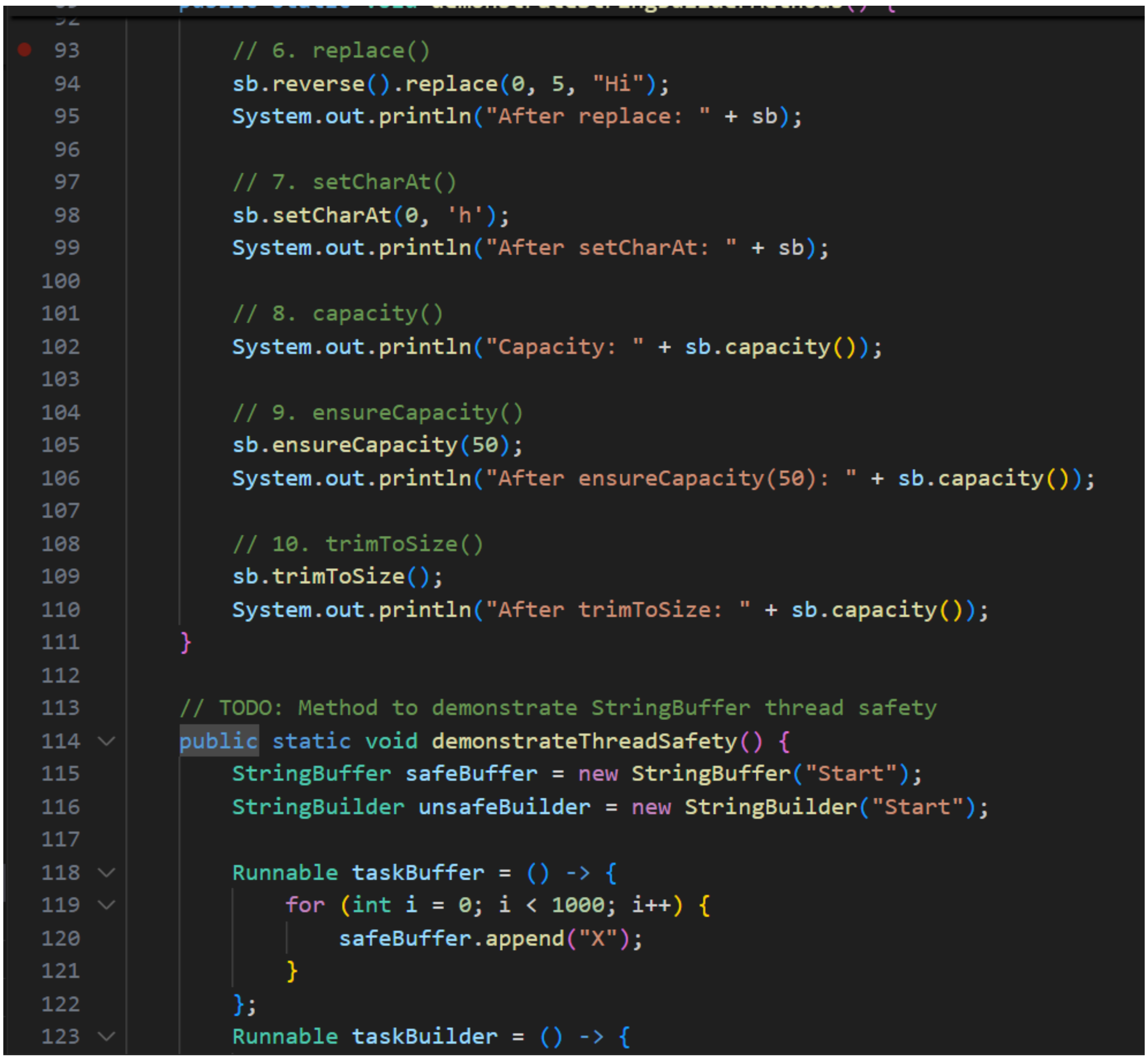
```java
public class StringPerformanceComparison {
    public static void main(String[] args) {
        // TODO: Implement performance tests for different approaches
        System.out.println("====== PERFORMANCE COMPARISON ======");

        // TODO: Test string concatenation with regular String (slow method)
        long startTime = System.nanoTime();
        String result1 = concatenateWithString(10000);
        long endTime = System.nanoTime();
        System.out.println("String concatenation time: " + (endTime - startTime) + " ns");

        // TODO: Test string concatenation with StringBuilder (fast method)
        startTime = System.nanoTime();
        String result2 = concatenateWithStringBuilder(10000);
        endTime = System.nanoTime();
        System.out.println("StringBuilder concatenation time: " + (endTime - startTime) + " ns");

        // TODO: Test string concatenation with StringBuffer (thread-safe method)
        startTime = System.nanoTime();
        String result3 = concatenateWithStringBuffer(10000);
        endTime = System.nanoTime();
        System.out.println("StringBuffer concatenation time: " + (endTime - startTime) + " ns");

        // TODO: Demonstrate StringBuilder methods
        System.out.println("\n====== STRINGBUILDER METHODS DEMO ======");
        demonstrateStringBuilderMethods();

        // TODO: Demonstrate thread safety differences
        System.out.println("\n====== THREAD SAFETY DEMO ======");
        demonstrateThreadSafety();

        // TODO: Compare string comparison methods
        System.out.println("\n====== STRING COMPARISON METHODS ======");
        compareStringComparisonMethods();
```



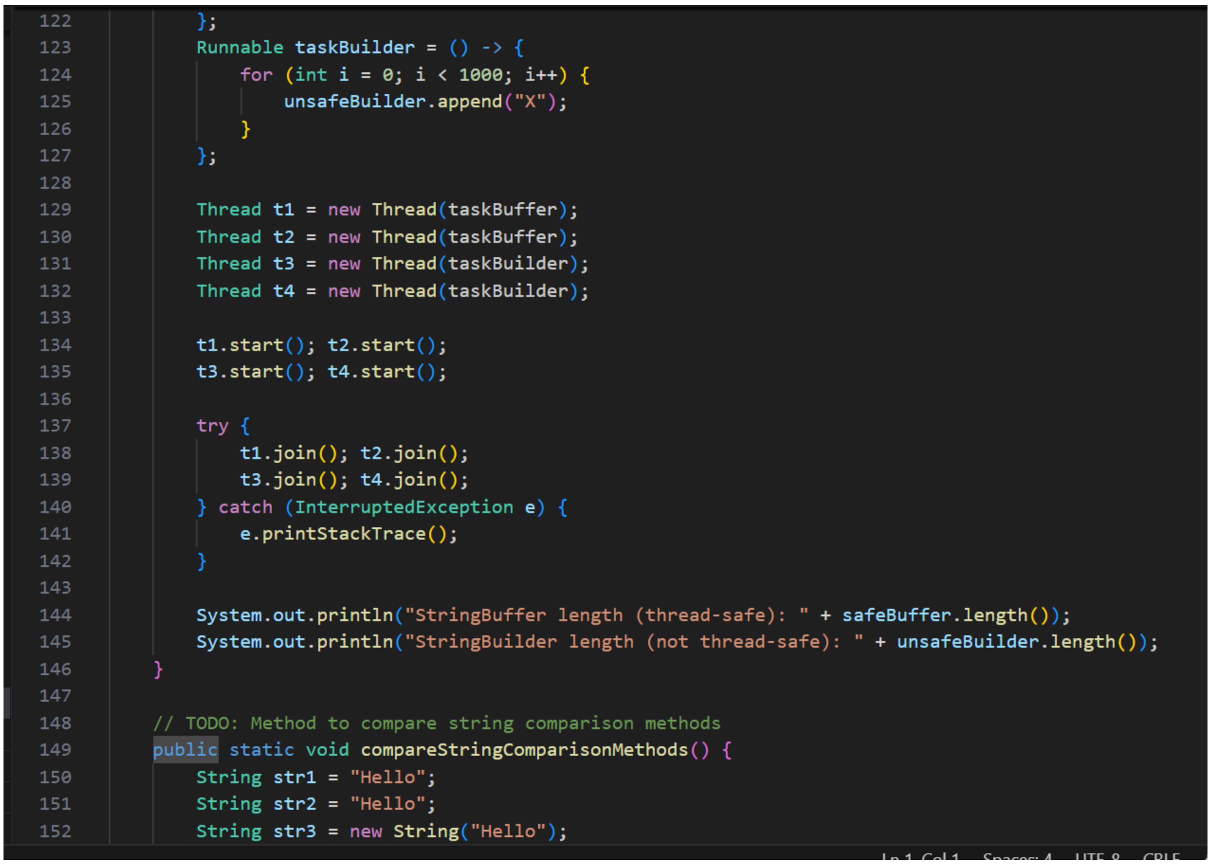
```java
        System.out.println("\n====== THREAD SAFETY DEMO ======");
        demonstrateThreadSafety();

        // TODO: Compare string comparison methods
        System.out.println("\n====== STRING COMPARISON METHODS ======");
        compareStringComparisonMethods();

        // TODO: Demonstrate memory efficiency
        System.out.println("\n====== MEMORY EFFICIENCY DEMO ======");
        demonstrateMemoryEfficiency();
    }

    // TODO: Method using String concatenation (inefficient)
    public static String concatenateWithString(int iterations) {
        String result = "";
        for (int i = 0; i < iterations; i++) {
            result += "Java" + i + " ";
        }
        return result;
    }

    // TODO: Method using StringBuilder (efficient, not thread-safe)
    public static String concatenateWithStringBuilder(int iterations) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < iterations; i++) {
            sb.append("Java").append(i).append(" ");
        }
        return sb.toString();
    }

    // TODO: Method using StringBuffer (efficient, thread-safe)
    public static String concatenateWithStringBuffer(int iterations) {
```



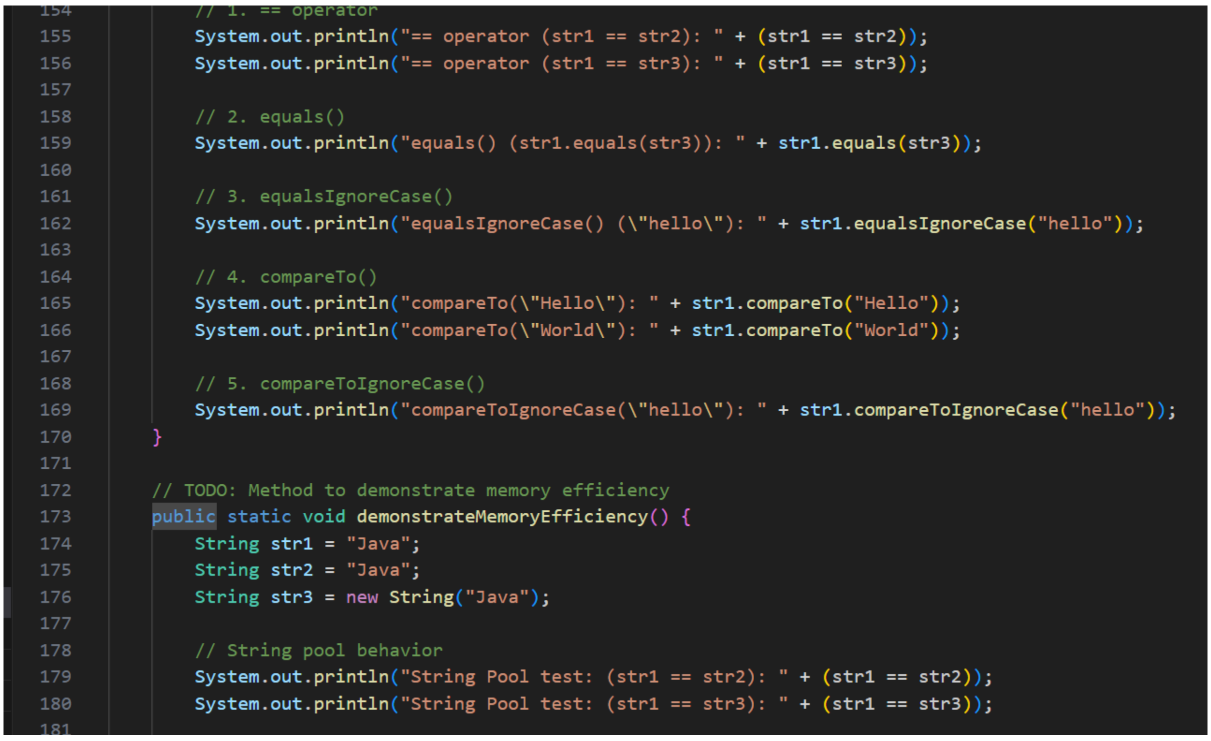
```java
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < iterations; i++) {
            sb.append("Java").append(i).append(" ");
        }
        return sb.toString();
    }

    // TODO: Method to demonstrate StringBuilder methods
    public static void demonstrateStringBuilderMethods() {
        StringBuilder sb = new StringBuilder("Hello World");
        System.out.println("Original: " + sb);

        // 1. append()
        sb.append(" Java");
        System.out.println("After append: " + sb);

        // 2. insert()
        sb.insert(6, "Beautiful ");
        System.out.println("After insert: " + sb);

        // 3. delete()
        sb.delete(6, 16);
        System.out.println("After delete: " + sb);

        // 4. deleteCharAt()
        sb.deleteCharAt(5);
        System.out.println("After deleteCharAt: " + sb);

        // 5. reverse()
        sb.reverse();
        System.out.println("After reverse: " + sb);
```



```java
        // 6. replace()
        sb.reverse().replace(0, 5, "Hi");
        System.out.println("After replace: " + sb);

        // 7. setCharAt()
        sb.setCharAt(0, 'h');
        System.out.println("After setCharAt: " + sb);

        // 8. capacity()
        System.out.println("Capacity: " + sb.capacity());

        // 9. ensureCapacity()
        sb.ensureCapacity(50);
        System.out.println("After ensureCapacity(50): " + sb.capacity());

        // 10. trimToSize()
        sb.trimToSize();
        System.out.println("After trimToSize: " + sb.capacity());
    }

    // TODO: Method to demonstrate StringBuffer thread safety
    public static void demonstrateThreadSafety() {
        StringBuffer safeBuffer = new StringBuffer("Start");
        StringBuilder unsafeBuilder = new StringBuilder("Start");

        Runnable taskBuffer = () -> {
            for (int i = 0; i < 1000; i++) {
                safeBuffer.append("X");
            }
        };
        Runnable taskBuilder = () -> {
```

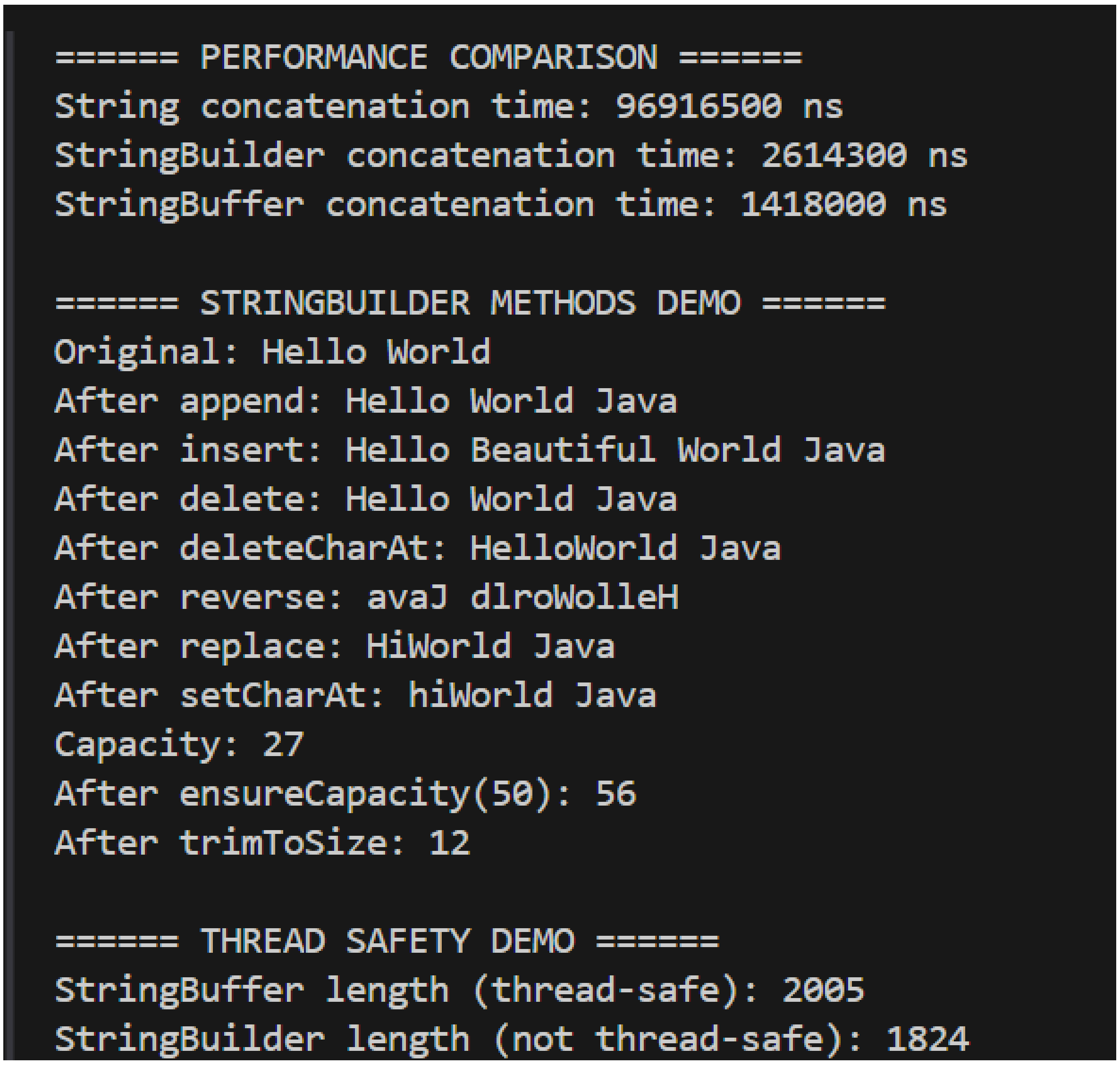


```java
        };
        Runnable taskBuilder = () -> {
            for (int i = 0; i < 1000; i++) {
                unsafeBuilder.append("X");
            }
        };

        Thread t1 = new Thread(taskBuffer);
        Thread t2 = new Thread(taskBuffer);
        Thread t3 = new Thread(taskBuilder);
        Thread t4 = new Thread(taskBuilder);

        t1.start(); t2.start();
        t3.start(); t4.start();

        try {
            t1.join(); t2.join();
            t3.join(); t4.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("StringBuffer length (thread-safe): " + safeBuffer.length());
        System.out.println("StringBuilder length (not thread-safe): " + unsafeBuilder.length());
    }

    // TODO: Method to compare string comparison methods
    public static void compareStringComparisonMethods() {
        String str1 = "Hello";
        String str2 = "Hello";
        String str3 = new String("Hello");
```

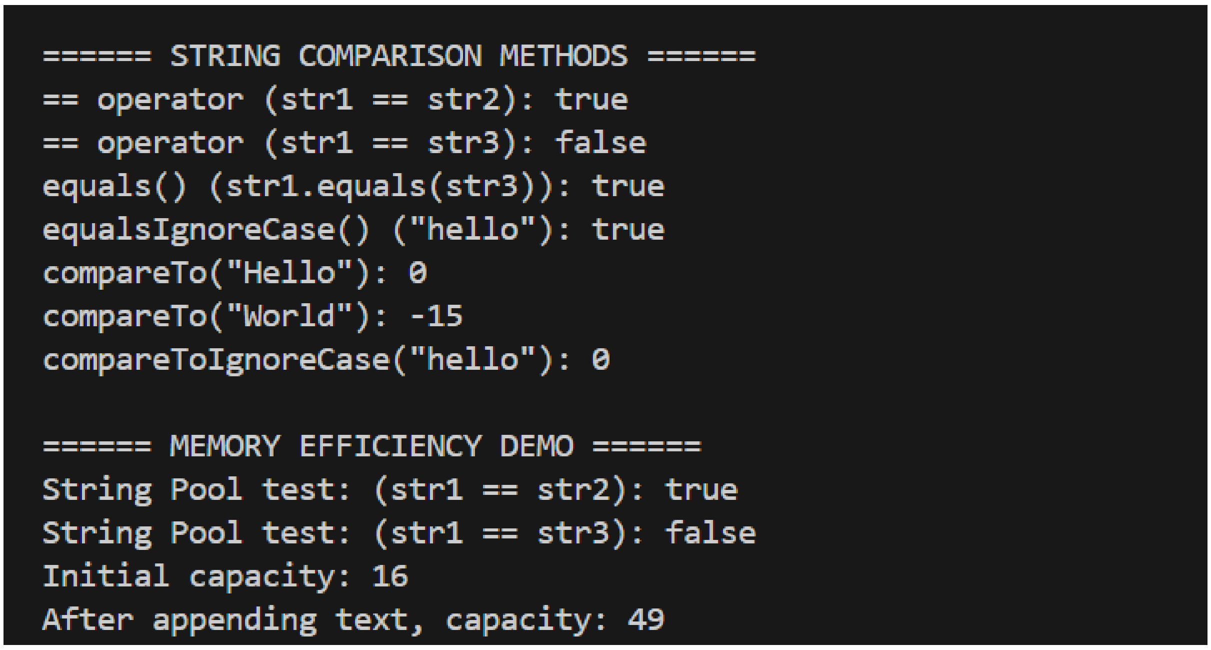


```java
        // 1. == operator
        System.out.println("== operator (str1 == str2): " + (str1 == str2));
        System.out.println("== operator (str1 == str3): " + (str1 == str3));

        // 2. equals()
        System.out.println("equals() (str1.equals(str3)): " + str1.equals(str3));

        // 3. equalsIgnoreCase()
        System.out.println("equalsIgnoreCase() (\"hello\"): " + str1.equalsIgnoreCase("hello"));

        // 4. compareTo()
        System.out.println("compareTo(\"Hello\"): " + str1.compareTo("Hello"));
        System.out.println("compareTo(\"World\"): " + str1.compareTo("World"));

        // 5. compareToIgnoreCase()
        System.out.println("compareToIgnoreCase(\"hello\"): " + str1.compareToIgnoreCase("hello"));
    }

    // TODO: Method to demonstrate memory efficiency
    public static void demonstrateMemoryEfficiency() {
        String str1 = "Java";
        String str2 = "Java";
        String str3 = new String("Java");

        // String pool behavior
        System.out.println("String Pool test: (str1 == str2): " + (str1 == str2));
        System.out.println("String Pool test: (str1 == str3): " + (str1 == str3));
```

OUTPUT→



```
====== PERFORMANCE COMPARISON ======
String concatenation time: 96916500 ns
StringBuilder concatenation time: 2614300 ns
StringBuffer concatenation time: 1418000 ns

====== STRINGBUILDER METHODS DEMO ======
Original: Hello World
After append: Hello World Java
After insert: Hello Beautiful World Java
After delete: Hello World Java
After deleteCharAt: HelloWorld Java
After reverse: avaJ dlroWolleH
After replace: HiWorld Java
After setCharAt: hiWorld Java
Capacity: 27
After ensureCapacity(50): 56
After trimToSize: 12

====== THREAD SAFETY DEMO ======
StringBuffer length (thread-safe): 2005
StringBuilder length (not thread-safe): 1824
```



```
====== STRING COMPARISON METHODS ======
== operator (str1 == str2): true
== operator (str1 == str3): false
equals() (str1.equals(str3)): true
equalsIgnoreCase() ("hello"): true
compareTo("Hello"): 0
compareTo("World"): -15
compareToIgnoreCase("hello"): 0

====== MEMORY EFFICIENCY DEMO ======
String Pool test: (str1 == str2): true
String Pool test: (str1 == str3): false
Initial capacity: 16
After appending text, capacity: 49
```