Built-In String Methods - Basic Operations
Task: Create a program that demonstrates common String methods for text analysis and manipulation.

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

        System.out.println("1. Original String: \"" + sampleText + "\"");
        System.out.println("   Length (with spaces): " +
sampleText.length());

        String trimmed = sampleText.trim();
        System.out.println("2. Trimmed String: \"" + trimmed + "\"");
        System.out.println("   Length (after trim): " + trimmed.length());

        System.out.println("3. Character at index 5: " +
sampleText.charAt(5));

        String sub = trimmed.substring(5, 16);
        System.out.println("4. Substring: " + sub);

        System.out.println("5. Index of 'Fun': " + trimmed.indexOf("Fun"));

        System.out.println("6. Contains 'Java': " +
trimmed.contains("Java"));

        System.out.println("7. Starts with 'Java': " +
trimmed.startsWith("Java"));

        System.out.println("8. Ends with '!': " + trimmed.endsWith("!"));

        System.out.println("9. Uppercase: " + trimmed.toUpperCase());

        System.out.println("10. Lowercase: " + trimmed.toLowerCase());

        System.out.println("11. Vowel Count: " + countVowels(trimmed));

        System.out.print("12. Occurrences of 'a': ");
        findAllOccurrences(trimmed, 'a');
    }
```

```
    public static int countVowels(String text) {
        int count = 0;
        text = text.toLowerCase();
        for (int i = 0; i < text.length(); i++) {
            char ch = text.charAt(i);
            if ("aeiou".indexOf(ch) != -1) {
                count++;
            }
        }
        return count;
    }

    public static void findAllOccurrences(String text, char target) {
        for (int i = 0; i < text.length(); i++) {
            if (text.charAt(i) == target) {
                System.out.print(i + " ");
            }
        }
        System.out.println();
    }
}
```

String Manipulation Methods
Task: Create a text processing utility that uses various string manipulation methods.

```
import java.util.*;

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

        String trimmed = input.trim();
        String replaced = trimmed.replace(" ", "_");
        String removedDigits = replaced.replaceAll("\\d", "");
        String[] words = removedDigits.split("_");
        String joined = String.join(" | ", words);
```

```
        System.out.println("Trimmed: " + trimmed);
        System.out.println("Spaces replaced: " + replaced);
        System.out.println("Digits removed: " + removedDigits);
        System.out.println("Words: " + Arrays.toString(words));
        System.out.println("Joined: " + joined);

        System.out.println("No punctuation: " + removePunctuation(input));
        System.out.println("Capitalized: " + capitalizeWords(input));
        System.out.println("Reversed order: " + reverseWordOrder(input));
        countWordFrequency(input);

        scanner.close();
    }

    public static String removePunctuation(String text) {
        return text.replaceAll("\\p{Punct}", "");
    }

    public static String capitalizeWords(String text) {
        String[] words = text.trim().split("\\s+");
        StringBuilder sb = new StringBuilder();
        for (String word : words) {
            sb.append(Character.toUpperCase(word.charAt(0)))
              .append(word.substring(1).toLowerCase())
              .append(" ");
        }
        return sb.toString().trim();
    }

    public static String reverseWordOrder(String text) {
        String[] words = text.trim().split("\\s+");
        Collections.reverse(Arrays.asList(words));
        return String.join(" ", words);
    }

    public static void countWordFrequency(String text) {
        String[] words = text.toLowerCase().replaceAll("\\p{Punct}",
"").split("\\s+");
        Map<String, Integer> freq = new HashMap<>();
        for (String word : words) freq.put(word, freq.getOrDefault(word, 0)
+ 1);
        System.out.println("Word Frequency: " + freq);
    }
```

```
}
```

ASCII Codes and Character Conversion
Task: Create a program that demonstrates ASCII character manipulation and conversion.

```
import java.util.*;

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

        for (char ch : input.toCharArray()) {
            System.out.println(ch + " -> " + (int) ch + " (" +
classifyCharacter(ch) + ")");
            if (Character.isLetter(ch)) {
                char toggled = toggleCase(ch);
                System.out.println("   Toggled: " + toggled + " -> " +
(int) toggled);
            }
        }

        System.out.println("Caesar Cipher (shift 3): " +
caesarCipher(input, 3));

        displayASCIITable(65, 90);

        int[] asciiArr = stringToASCII(input);
        System.out.println("ASCII Array: " + Arrays.toString(asciiArr));
        System.out.println("Back to String: " + asciiToString(asciiArr));

        scanner.close();
    }

    public static String classifyCharacter(char ch) {
        if (Character.isUpperCase(ch)) return "Uppercase Letter";
        if (Character.isLowerCase(ch)) return "Lowercase Letter";
        if (Character.isDigit(ch)) return "Digit";
```

```
        return "Special Character";
    }

    public static char toggleCase(char ch) {
        if (Character.isUpperCase(ch)) return (char)(ch + 32);
        if (Character.isLowerCase(ch)) return (char)(ch - 32);
        return ch;
    }

    public static String caesarCipher(String text, int shift) {
        StringBuilder sb = new StringBuilder();
        for (char ch : text.toCharArray()) {
            if (Character.isLetter(ch)) {
                char base = Character.isUpperCase(ch) ? 'A' : 'a';
                sb.append((char) ((ch - base + shift) % 26 + base));
            } else sb.append(ch);
        }
        return sb.toString();
    }

    public static void displayASCIITable(int start, int end) {
        for (int i = start; i <= end; i++) {
            System.out.println(i + " -> " + (char) i);
        }
    }

    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;
    }

    public static String asciiToString(int[] asciiValues) {
        StringBuilder sb = new StringBuilder();
        for (int val : asciiValues) sb.append((char) val);
        return sb.toString();
    }
}
```

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

```
public class StringPerformanceComparison {
    public static void main(String[] args) {
        System.out.println("=== PERFORMANCE COMPARISON ===");

        long startTime = System.nanoTime();
        concatenateWithString(1000);
        long endTime = System.nanoTime();
        System.out.println("String concatenation time: " + (endTime -
startTime) + " ns");

        startTime = System.nanoTime();
        concatenateWithStringBuilder(1000);
        endTime = System.nanoTime();
        System.out.println("StringBuilder concatenation time: " + (endTime
- startTime) + " ns");

        startTime = System.nanoTime();
        concatenateWithStringBuffer(1000);
        endTime = System.nanoTime();
        System.out.println("StringBuffer concatenation time: " + (endTime -
startTime) + " ns");

        demonstrateStringBuilderMethods();
        compareStringComparisonMethods();
    }

    public static String concatenateWithString(int iterations) {
        String result = "";
        for (int i = 0; i < iterations; i++) {
            result += "Java " + i + " ";
        }
        return result;
    }

    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();
    }

    public static String concatenateWithStringBuffer(int iterations) {
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < iterations; i++) {
            sb.append("Java ").append(i).append(" ");
        }
        return sb.toString();
    }

    public static void demonstrateStringBuilderMethods() {
        StringBuilder sb = new StringBuilder("Hello World");
        sb.append("!!!");
        sb.insert(6, "Java ");
        sb.delete(0, 6);
        sb.deleteCharAt(0);
        sb.reverse();
        sb.replace(0, 4, "Hi");
        sb.setCharAt(0, 'X');
        System.out.println("StringBuilder result: " + sb);
        System.out.println("Capacity: " + sb.capacity());
        sb.ensureCapacity(50);
        sb.trimToSize();
    }

    public static void compareStringComparisonMethods() {
        String str1 = "Hello";
        String str2 = "Hello";
        String str3 = new String("Hello");

        System.out.println("== operator: " + (str1 == str2));
        System.out.println("== with new String: " + (str1 == str3));
        System.out.println("equals(): " + str1.equals(str3));
        System.out.println("equalsIgnoreCase(): " +
str1.equalsIgnoreCase("hello"));
        System.out.println("compareTo(): " + str1.compareTo(str3));
        System.out.println("compareToIgnoreCase(): " +
str1.compareToIgnoreCase("hello"));
    }
}
```