Week 2 – S2 – Lab Solution

Name: Ramesh Harisabapathi Chettiar

Roll Number: RA2411030010263

Course: Networking and Communications

Semester: 3

Date of Submission: 23/08/2025

Problem 1: Write a program to find and replace all occurrences of a substring in a text without using the replace() method

Hint =>

a. Take user input using the Scanner nextLine() method for the main text and the substring to find and replace

b. Create a method to find all occurrences of the substring using indexOf() method in a loop and store the starting positions in an array

c. Create a method to replace the substring manually by:

- **i. Building a new string character by character using charAt() method**
- **ii. Skip the characters of the original substring and insert the replacement substring**

d. Create a method to compare the result with the built-in replace() method and return a boolean

e. The main function calls all user-defined methods and displays both results along with the comparison

```java
/*Problem 1: Write a program to find and replace all occurrences of a
substring in a text without using the replace() method
Hint =>
a. Take user input using the Scanner nextLine() method for the main text and the substring
to find and replace
b. Create a method to find all occurrences of the substring using indexOf() method in a loop
and store the starting positions in an array
c. Create a method to replace the substring manually by:
• i. Building a new string character by character using charAt() method
• ii. Skip the characters of the original substring and insert the replacement substring
d. Create a method to compare the result with the built-in replace() method and return a
boolean
e. The main function calls all user-defined methods and displays both results along with the
comparison */

import java.util.ArrayList;
import java.util.Scanner;

public class CustomReplace {

    /*b. Create a method to find all occurrences of the substring using indexOf() method in a loop
    and store the starting positions in an array */
    public static ArrayList<Integer> findAllOccurrences(String text, String find) {
        ArrayList<Integer> positions = new ArrayList<>();
        int index = text.indexOf(find);

        while (index != -1) {
            positions.add(index);
            index = text.indexOf(find, index + find.length()); // FIX: move by full word, not +1
        }
        return positions;
    }
```

```java
    /*c. Create a method to replace the substring manually by:
    • i. Building a new string character by character using charAt() method
    • ii. Skip the characters of the original substring and insert the replacement substring */
    public static String manualReplace(String text, String find, String replace) {
        StringBuilder result = new StringBuilder();
        int i = 0;

        while (i < text.length()) {
            if (i <= text.length() - find.length() &&
                text.substring(i, i + find.length()).equals(find)) {
                result.append(replace);
                i += find.length();
            } else {
                result.append(text.charAt(i));
                i++;
            }
        }
        return result.toString();
    }

    /*d. Create a method to compare the result with the built-in replace() method and return a
    boolean */
    public static boolean compareResults(String manual, String builtin) {
        return manual.equals(builtin);
    }

    /*e. The main function calls all user-defined methods and displays both results along with the
    comparison */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("Enter the main text:");
```

```
66          String text = input.nextLine();
67
68          System.out.println("Enter the substring to find:");
69          String find = input.nextLine();
70
71          System.out.println("Enter the substring to replace with:");
72          String replace = input.nextLine();
73
74          ArrayList<Integer> positions = findAllOccurrences(text, find);
75          System.out.println("Found at positions: " + positions);
76
77          String manualResult = manualReplace(text, find, replace);
78          String builtInResult = text.replace(find, replace);
79
80          boolean isSame = compareResults(manualResult, builtInResult);
81
82          System.out.println("\n====== RESULTS ======");
83          System.out.println("Original text:   " + text);
84          System.out.println("Manual result:   " + manualResult);
85          System.out.println("Built-in result: " + builtInResult);
86          System.out.println("Both are same?   " + isSame);
87
88          input.close();
89      }
90  }
```

OUTPUT→

```
Enter the main text:
Ramesh Harisabapathi Chettiar
Enter the substring to find:
tia
Enter the substring to replace with:
206
Found at positions: [25]

====== RESULTS ======
Original text:   Ramesh Harisabapathi Chettiar
Manual result:   Ramesh Harisabapathi Chet206r
Built-in result: Ramesh Harisabapathi Chet206r
Both are same?   true
```

Problem 2: Write a program to convert text between different cases (uppercase, lowercase, title case) using ASCII values without using built-in case conversion methods

Hint =>

a. Take user input using the Scanner nextLine() method

b. Create a method to convert a character to uppercase using ASCII values:

- **i. Check if the character is a lowercase letter (ASCII 97-122)**
- **ii. Convert by subtracting 32 from the ASCII value**

c. Create a method to convert a character to lowercase using ASCII values:

- **i. Check if the character is an uppercase letter (ASCII 65-90)**
- **ii. Convert by adding 32 to the ASCII value**

d. Create a method for title case conversion:

- **i. Convert the first character of each word to uppercase**
- **ii. Convert all other characters to lowercase**

e. Create a method to compare results with built-in methods (toUpperCase(), toLowerCase())

f. The main function calls all methods and displays the results in a tabular format

```java
/*Problem 2: Write a program to convert text between different cases
(uppercase, lowercase, title case) using ASCII values without using built-in
case conversion methods
Hint =>
a. Take user input using the Scanner nextLine() method
b. Create a method to convert a character to uppercase using ASCII values:
• i. Check if the character is a lowercase letter (ASCII 97-122)
• ii. Convert by subtracting 32 from the ASCII value
c. Create a method to convert a character to lowercase using ASCII values:
• i. Check if the character is an uppercase letter (ASCII 65-90)
• ii. Convert by adding 32 to the ASCII value
d. Create a method for title case conversion:
• i. Convert the first character of each word to uppercase
• ii. Convert all other characters to lowercase
e. Create a method to compare results with built-in methods (toUpperCase(),
toLowerCase())
f. The main function calls all methods and displays the results in a tabular format */

import java.util.Scanner;

public class CustomASCIICaseConvertor {

    /*b. Convert char to uppercase using ASCII values */
    public static char toUpperChar(char c) {
        if (c >= 'a' && c <= 'z') {
            return (char) (c - 32);
        }
        return c;
    }

    /*c. Convert char to lowercase using ASCII values */
    public static char toLowerChar(char c) {
        if (c >= 'A' && c <= 'Z') {
            return (char) (c + 32);
        }
        return c;
    }

    /*d. Convert whole string to uppercase manually */
    public static String toUpperCaseManual(String text) {
        StringBuilder sb = new StringBuilder();
        for (char c : text.toCharArray()) {
            sb.append(toUpperChar(c));
        }
        return sb.toString();
    }

    /* Convert whole string to lowercase manually */
    public static String toLowerCaseManual(String text) {
        StringBuilder sb = new StringBuilder();
        for (char c : text.toCharArray()) {
            sb.append(toLowerChar(c));
        }
        return sb.toString();
    }

    /* Title case conversion manually */
    public static String toTitleCaseManual(String text) {
        StringBuilder sb = new StringBuilder();
        boolean newWord = true;

        for (char c : text.toCharArray()) {
            if (c == ' ') {
                sb.append(c);
                newWord = true;
```

```
                } else {
                    if (newWord) {
                        sb.append(toUpperChar(c));
                        newWord = false;
                    } else {
                        sb.append(toLowerChar(c));
                    }
                }
            }
            return sb.toString();
        }

        /* Built-in title case for comparison */
        public static String toTitleCaseBuiltIn(String text) {
            String[] words = text.toLowerCase().split(" ");
            StringBuilder sb = new StringBuilder();

            for (int i = 0; i < words.length; i++) {
                String word = words[i];
                if (word.length() > 0) {
                    sb.append(Character.toUpperCase(word.charAt(0)));
                    sb.append(word.substring(1));
                }
                if (i < words.length - 1) sb.append(" ");
            }
            return sb.toString();
        }

        /*a. Main function */
```

```
    /*a. Main function */
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("Enter a text:");
        String text = input.nextLine();

        String upperManual = toUpperCaseManual(text);
        String lowerManual = toLowerCaseManual(text);
        String titleManual = toTitleCaseManual(text);

        String upperBuiltIn = text.toUpperCase();
        String lowerBuiltIn = text.toLowerCase();
        String titleBuiltIn = toTitleCaseBuiltIn(text);

        System.out.println("\n====== CASE CONVERSION RESULTS ======");
        System.out.printf("%-20s %-40s %-40s\n", "Conversion", "Manual Result", "Built-in Result");
        System.out.println("-----------------------------------------------------------------------------------------");
        System.out.printf("%-20s %-40s %-40s\n", "UPPERCASE", upperManual, upperBuiltIn);
        System.out.printf("%-20s %-40s %-40s\n", "lowercase", lowerManual, lowerBuiltIn);
        System.out.printf("%-20s %-40s %-40s\n", "Title Case", titleManual, titleBuiltIn);

        input.close();
    }
}
```

OUTPUT→

```
Enter a text:
Cybersecurity is a necessity for the future of AI

====== CASE CONVERSION RESULTS ======
Conversion          Manual Result                          Built-in Result
--------------------------------------------------------------------------------
UPPERCASE           CYBERSECURITY IS A NECESSITY FOR THE FUTURE OF AI CYBERSECURITY IS A NECESSITY FOR THE FUTURE OF AI
lowercase           cybersecurity is a necessity for the future of ai cybersecurity is a necessity for the future of ai
Title Case          Cybersecurity Is A Necessity For The Future Of Ai Cybersecurity Is A Necessity For The Future Of Ai
```

Problem 3: Write a program to analyze and compare the performance of

String concatenation vs StringBuilder vs StringBuffer for building large

strings

Hint =>

a. Take user input for the number of iterations (e.g., 1000, 10000, 100000)

b. Create a method to perform String concatenation in a loop:

- **i. Use System.currentTimeMillis() to measure start and end time**
- **ii. Concatenate a sample string multiple times using the + operator**
- **iii. Return the time taken and final string length**

c. Create a method to perform StringBuilder operations:

- **i. Use StringBuilder.append() method in a loop**
- **ii. Measure the time taken and return results**

d. Create a method to perform StringBuffer operations:

- **i. Use StringBuffer.append() method in a loop**
- **ii. Measure the time taken and return results**

e. Create a method to display performance comparison in a tabular format showing:

- **i. Method used, Time taken (milliseconds), Memory efficiency**

f. The main function calls all methods and displays the performance analysis

```java
/*Problem 3: Write a program to analyze and compare the performance of
String concatenation vs StringBuilder vs StringBuffer for building large
strings
Hint =>
a. Take user input for the number of iterations (e.g., 1000, 10000, 100000)
b. Create a method to perform String concatenation in a loop:
    • i. Use System.currentTimeMillis() to measure start and end time
    • ii. Concatenate a sample string multiple times using the + operator
    • iii. Return the time taken and final string length
c. Create a method to perform StringBuilder operations:
    • i. Use StringBuilder.append() method in a loop
    • ii. Measure the time taken and return results
d. Create a method to perform StringBuffer operations:
    • i. Use StringBuffer.append() method in a loop
    • ii. Measure the time taken and return results
e. Create a method to display performance comparison in a tabular format */

import java.util.Scanner;

public class CustomStringTester {

    /*b. Perform String concatenation in a loop */
    public static long stringConcatTest(int n) {
        String result = "";
        long start = System.currentTimeMillis();
        for(int i=0; i<n; i++) {
            result += "x";
        }
        long end = System.currentTimeMillis();
        System.out.println("Final length using String: " + result.length());
        return (end-start);
    }

    /*c. Perform StringBuilder operations */
```

```java
    /*c. Perform StringBuilder operations */
    public static long stringBuilderTest(int n) {
        StringBuilder sb = new StringBuilder();
        long start = System.currentTimeMillis();
        for(int i=0; i<n; i++) {
            sb.append("x");
        }
        long end = System.currentTimeMillis();
        System.out.println("Final length using StringBuilder: " + sb.length());
        return (end-start);
    }

    /*d. Perform StringBuffer operations */
    public static long stringBufferTest(int n) {
        StringBuffer sb = new StringBuffer();
        long start = System.currentTimeMillis();
        for(int i=0; i<n; i++) {
            sb.append("x");
        }
        long end = System.currentTimeMillis();
        System.out.println("Final length using StringBuffer: " + sb.length());
        return (end-start);
    }

    /*e. Display performance comparison */
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of iterations: ");
        int n = sc.nextInt();

        long timeString = stringConcatTest(n);
        long timeBuilder = stringBuilderTest(n);
        long timeBuffer = stringBufferTest(n);

        System.out.println("\n====== PERFORMANCE COMPARISON ======");
        System.out.printf("%-20s %-20s\n", "Method", "Time Taken (ms)");
        System.out.println("-----------------------------------------");
        System.out.printf("%-20s %-20d\n", "String (+)", timeString);
        System.out.printf("%-20s %-20d\n", "StringBuilder", timeBuilder);
        System.out.printf("%-20s %-20d\n", "StringBuffer", timeBuffer);

        sc.close();
    }
}
```

OUTPUT→

```
Enter number of iterations: 100000
Final length using String: 100000
Final length using StringBuilder: 100000
Final length using StringBuffer: 100000

====== PERFORMANCE COMPARISON ======
Method                Time Taken (ms)
-----------------------------------------
String (+)            624
StringBuilder         4
StringBuffer          3
```

Problem 4: Write a program to create a simple encryption and decryption system using ASCII character shifting (Caesar Cipher implementation)

Hint =>

a. Take user input for the text to encrypt and the shift value

b. Create a method to encrypt text using ASCII values:

- **i. For each character, get its ASCII value using (int) casting**
- **ii. Shift the ASCII value by the given amount**
- **iii. Handle wrap-around for alphabetic characters (A-Z, a-z)**
- **iv. Keep non-alphabetic characters unchanged**

c. Create a method to decrypt text:

- **i. Reverse the shifting process**
- **ii. Handle negative shifts properly**

d. Create a method to display ASCII values of characters before and after encryption

e. Create a method to validate that decryption returns the original text

f. The main function takes inputs, calls encryption/decryption methods, and displays:

- **i. Original text with ASCII values**
- **ii. Encrypted text with ASCII values**
- **iii. Decrypted text with validation result**

```java
/*Problem 4: Write a program to implement a simple Substitution Cipher
(Caesar Cipher) for encryption and decryption
Hint =>
a. Take user input for a plain text string
b. Take user input for a key (shift value)
c. Create a method for encryption:
   • i. Convert each character using (char + shift)
   • ii. Ensure wrapping using modulo for alphabets
d. Create a method for decryption:
   • i. Convert each character using (char - shift)
   • ii. Ensure wrapping using modulo
e. Display both encrypted and decrypted results along with the shift value */

import java.util.Scanner;

public class SubstitutionCipher {

    // c. Encryption method using Caesar Cipher
    public static String encrypt(String text, int shift) {
        StringBuilder result = new StringBuilder();

        for (char c : text.toCharArray()) {
            if (c >= 'A' && c <= 'Z') {
                result.append((char) (((c - 'A' + shift) % 26) + 'A'));
            }
            else if (c >= 'a' && c <= 'z') {
                result.append((char) (((c - 'a' + shift) % 26) + 'a'));
            }
            else {
                result.append(c); // keep symbols/numbers as is
            }
        }
        return result.toString();
    }
```

```java
    // d. Decryption method using Caesar Cipher
    public static String decrypt(String text, int shift) {
        StringBuilder result = new StringBuilder();

        for (char c : text.toCharArray()) {
            if (c >= 'A' && c <= 'Z') {
                result.append((char) (((c - 'A' - shift + 26) % 26) + 'A'));
            }
            else if (c >= 'a' && c <= 'z') {
                result.append((char) (((c - 'a' - shift + 26) % 26) + 'a'));
            }
            else {
                result.append(c);
            }
        }
        return result.toString();
    }

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

        // a. Take user input for plain text
        System.out.print("Enter a plain text: ");
        String plainText = input.nextLine();

        // b. Take user input for key (shift value)
        System.out.print("Enter a shift key (public key): ");
        int shift = input.nextInt();

        // Encryption & Decryption
        String encrypted = encrypt(plainText, shift);
        String decrypted = decrypt(encrypted, shift);
```

```java
    // d. Decryption method using Caesar Cipher
    public static String decrypt(String text, int shift) {
        StringBuilder result = new StringBuilder();

        for (char c : text.toCharArray()) {
            if (c >= 'A' && c <= 'Z') {
                result.append((char) (((c - 'A' - shift + 26) % 26) + 'A'));
            }
            else if (c >= 'a' && c <= 'z') {
                result.append((char) (((c - 'a' - shift + 26) % 26) + 'a'));
            }
            else {
                result.append(c);
            }
        }
        return result.toString();
    }

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

        // a. Take user input for plain text
        System.out.print("Enter a plain text: ");
        String plainText = input.nextLine();

        // b. Take user input for key (shift value)
        System.out.print("Enter a shift key (public key): ");
        int shift = input.nextInt();

        // Encryption & Decryption
        String encrypted = encrypt(plainText, shift);
        String decrypted = decrypt(encrypted, shift);
```

OUTPUT→

```
Enter a plain text: Caesar Cipher
Enter a shift key (public key): 13

====== SUBSTITUTION CIPHER RESULTS ======
Public Key (Shift): 13
Original Text      : Caesar Cipher
Encrypted Text     : Pnrfne Pvcure
Decrypted Text     : Caesar Cipher
```

Problem 5: Write a program to extract and analyze different parts of an email address using substring() and indexOf() methods

Hint =>

a. Take user input for multiple email addresses using Scanner

b. Create a method to validate email format:

- **i. Check for exactly one '@' symbol using indexOf() and lastIndexOf()**
- **ii. Check for at least one '.' after '@' symbol**
- **iii. Validate that username and domain are not empty**

c. Create a method to extract email components:

- **i. Extract username using substring() from start to '@' position**
- **ii. Extract domain using substring() from '@' position to end**
- **iii. Extract domain name and extension separately**

d. Create a method to analyze email statistics:

- **i. Count total valid/invalid emails**
- **ii. Find most common domain**
- **iii. Calculate average username length**

e. Create a method to display results in tabular format showing:

- **i. Email, Username, Domain, Domain Name, Extension, Valid/Invalid**

f. The main function processes multiple emails and displays analysis results

```java
/*Problem 5: Write a program to analyze email addresses
Hint =>
a. Take multiple email addresses as input (comma separated)
b. Create a method to validate email format using simple rules:
   • i. Must contain '@' and '.'
   • ii. '@' must appear before '.'
   • iii. No spaces allowed
c. Extract username and domain parts
d. Count emails by domain
e. Display all results neatly */

import java.util.*;

public class CustomEmailAnalyzer {

    /*b. Validate email */
    public static boolean isValidEmail(String email) {
        if(email.contains(" ") || !email.contains("@") || !email.contains(".")) return false;
        int at = email.indexOf('@');
        int dot = email.lastIndexOf('.');
        return at > 0 && dot > at;
    }

    /*c. Extract username */
    public static String getUsername(String email) {
        return email.substring(0, email.indexOf('@'));
    }

    /*c. Extract domain */
    public static String getDomain(String email) {
        return email.substring(email.indexOf('@')+1);
    }

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

        /*a. Take input */
        System.out.print("Enter email addresses (comma separated): ");
        String input = sc.nextLine();
        String[] emails = input.split(",");

        Map<String, Integer> domainCount = new HashMap<>();

        System.out.println("\n====== EMAIL ANALYSIS ======");
        System.out.printf("%-30s %-15s %-25s %-10s\n", "Email", "Username", "Domain", "Valid?");
        System.out.println("-------------------------------------------------------------------------");

        for(String raw : emails) {
            String email = raw.trim();
            boolean valid = isValidEmail(email);
            String username = valid ? getUsername(email) : "-";
            String domain = valid ? getDomain(email) : "-";
            System.out.printf("%-30s %-15s %-25s %-10s\n", email, username, domain, valid);

            if(valid) {
                domainCount.put(domain, domainCount.getOrDefault(domain, 0)+1);
```

```
55              if(valid) {
56                  domainCount.put(domain, domainCount.getOrDefault(domain, 0)+1);
57              }
58          }
59
60          /*d. Count by domain */
61          System.out.println("\n====== DOMAIN COUNTS ======");
62          for(Map.Entry<String,Integer> entry : domainCount.entrySet()) {
63              System.out.println(entry.getKey() + " : " + entry.getValue());
64          }
65
66          sc.close();
67      }
68  }
69
```

OUTPUT→

```
Enter email addresses (comma separated): gmail@gmail.com,  google@googlecom, outlook@outlook.com

====== EMAIL ANALYSIS ======
Email                          Username        Domain                  Valid?
------------------------------------------------------------------------
gmail@gmail.com                gmail           gmail.com               true
google@googlecom               -               -                       false
outlook@outlook.com            outlook         outlook.com             true

====== DOMAIN COUNTS ======
outlook.com : 1
gmail.com : 1
```

Problem 6: Write a program to create a text formatter that justifies text to a specified width using StringBuilder for efficient string manipulation

Hint =>

a. Take user input for the text to format and desired line width

b. Create a method to split text into words without using split():

- **i. Use charAt() to identify spaces**
- **ii. Extract words using substring() method**
- **iii. Store words in an array**

c. Create a method using StringBuilder to justify text:

- **i. Add words to current line until width limit is reached**
- **ii. Distribute extra spaces evenly between words**
- **iii. Handle last line separately (left-aligned only)**

d. Create a method to center-align text:

- **i. Calculate padding needed on both sides**
- **ii. Use StringBuilder to build centered lines**

e. Create a method to compare performance:

- **i. Implement the same formatting using String concatenation**
- **ii. Measure time difference using System.nanoTime()**

f. Create a method to display the formatted text with:

- **i. Line numbers**
- **ii. Character count per line**
- **iii. Performance comparison results**

g. The main function calls all methods and displays:

- **i. Original text**
- **ii. Left-justified text**
- **iii. Center-aligned text**
- **iv. Performance analysis**

Weeks > Week 2 > Lab > CustomStringTester.java

```java
/*Problem 3: Write a program to analyze and compare the performance of
String concatenation vs StringBuilder vs StringBuffer for building large
strings
Hint =>
a. Take user input for the number of iterations (e.g., 1000, 10000, 100000)
b. Create a method to perform String concatenation in a loop:
   • i. Use System.currentTimeMillis() to measure start and end time
   • ii. Concatenate a sample string multiple times using the + operator
   • iii. Return the time taken and final string length
c. Create a method to perform StringBuilder operations:
   • i. Use StringBuilder.append() method in a loop
   • ii. Measure the time taken and return results
d. Create a method to perform StringBuffer operations:
   • i. Use StringBuffer.append() method in a loop
   • ii. Measure the time taken and return results
e. Create a method to display performance comparison in a tabular format */

import java.util.Scanner;

public class CustomStringTester {

    /*b. Perform String concatenation in a loop */
    public static long stringConcatTest(int n) {
        String result = "";
        long start = System.currentTimeMillis();
        for(int i=0; i<n; i++) {
            result += "x";
        }
        long end = System.currentTimeMillis();
```

```java
public class CustomStringTester {

    /*c. Perform StringBuilder operations */
    public static long stringBuilderTest(int n) {
        StringBuilder sb = new StringBuilder();
        long start = System.currentTimeMillis();
        for(int i=0; i<n; i++) {
            sb.append("x");
        }
        long end = System.currentTimeMillis();
        System.out.println("Final length using StringBuilder: " + sb.length());
        return (end-start);
    }

    /*d. Perform StringBuffer operations */
    public static long stringBufferTest(int n) {
        StringBuffer sb = new StringBuffer();
        long start = System.currentTimeMillis();
        for(int i=0; i<n; i++) {
            sb.append("x");
        }
        long end = System.currentTimeMillis();
        System.out.println("Final length using StringBuffer: " + sb.length());
        return (end-start);
    }

```

```java
    /*d. Perform StringBuffer operations */
    public static long stringBufferTest(int n) {
        StringBuffer sb = new StringBuffer();
        long start = System.currentTimeMillis();
        for(int i=0; i<n; i++) {
            sb.append("x");
        }
        long end = System.currentTimeMillis();
        System.out.println("Final length using StringBuffer: " + sb.length());
        return (end-start);
    }

    /*e. Display performance comparison */
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of iterations: ");
        int n = sc.nextInt();

        long timeString = stringConcatTest(n);
        long timeBuilder = stringBuilderTest(n);
        long timeBuffer = stringBufferTest(n);

        System.out.println("\n====== PERFORMANCE COMPARISON ======");
        System.out.printf("%-20s %-20s\n", "Method", "Time Taken (ms)");
        System.out.println("----------------------------------------");
        System.out.printf("%-20s %-20d\n", "String (+)", timeString);
        System.out.printf("%-20s %-20d\n", "StringBuilder", timeBuilder);
        System.out.printf("%-20s %-20d\n", "StringBuffer", timeBuffer);
```

```java
        System.out.printf("%-20s %-20d\n", "StringBuilder", timeBuilder);
        System.out.printf("%-20s %-20d\n", "StringBuffer", timeBuffer);

        sc.close();
    }
}
```

OUTPUT→

```
Enter number of iterations: 100000
Final length using String: 100000
Final length using StringBuilder: 100000
Final length using StringBuffer: 100000

====== PERFORMANCE COMPARISON ======
Method               Time Taken (ms)
----------------------------------------
String (+)           672
StringBuilder        3
StringBuffer         3
```