

**Java String Operations and Performance Lab Practice Problems**

**Best Programming Practices**

1. Use Variables including for Fixed, User Inputs, and Results

2. Use Methods instead of writing code in the main() function

3. Proper naming conventions for all variables and methods

4. Proper Program Name and Class Name

5. Handle checked and Unchecked Exceptions wherever possible

6. Proper Method Name, which indicates action-taking inputs and provides rthe esult

**Lab Practice Programs (Any Six)**

**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

SOLUTION:

import java.util.\*;

public class FindAndReplaceDemo {

public static List<Integer> findOccurrences(String text, String findStr) {

List<Integer> positions = new ArrayList<>();

int index = text.indexOf(findStr);

while (index != -1) {

positions.add(index);

index = text.indexOf(findStr, index + 1);

}

return positions;

}

public static String manualReplace(String text, String findStr, String replaceStr) {

StringBuilder result = new StringBuilder();

int i = 0;

while (i < text.length()) {

if (i <= text.length() - findStr.length() &&

text.substring(i, i + findStr.length()).equals(findStr)) {

result.append(replaceStr);

i += findStr.length();

} else {

result.append(text.charAt(i));

i++;

}

}

return result.toString();

}

public static boolean compareResults(String text, String findStr, String replaceStr, String manualResult) {

String builtInResult = text.replace(findStr, replaceStr);

return manualResult.equals(builtInResult);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String text = sc.nextLine();

System.out.print("Enter the substring to find: ");

String findStr = sc.nextLine();

System.out.print("Enter the replacement substring: ");

String replaceStr = sc.nextLine();

List<Integer> positions = findOccurrences(text, findStr);

if (positions.isEmpty()) {

System.out.println("\nSubstring not found in the text!");

} else {

System.out.println("\nOccurrences of \"" + findStr + "\" found at positions: " + positions);

}

String manualResult = manualReplace(text, findStr, replaceStr);

System.out.println("\nManual Replace Result: " + manualResult);

String builtInResult = text.replace(findStr, replaceStr);

System.out.println("Built-in Replace Result: " + builtInResult);

boolean isSame = compareResults(text, findStr, replaceStr, manualResult);

System.out.println("\nDo both results match? " + (isSame ? "Yes" : "No"));

sc.close();

}

}

1



**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

SOLUTION:  
  
import java.util.\*;

public class TextCaseConversion {

public static char toUpper(char c) {

if (c >= 97 && c <= 122) return (char)(c - 32);

return c;

}

public static char toLower(char c) {

if (c >= 65 && c <= 90) return (char)(c + 32);

return c;

}

public static String toUpperCase(String text) {

StringBuilder sb = new StringBuilder();

for (int i = 0; i < text.length(); i++) sb.append(toUpper(text.charAt(i)));

return sb.toString();

}

public static String toLowerCase(String text) {

StringBuilder sb = new StringBuilder();

for (int i = 0; i < text.length(); i++) sb.append(toLower(text.charAt(i)));

return sb.toString();

}

public static String toTitleCase(String text) {

StringBuilder sb = new StringBuilder();

boolean newWord = true;

for (int i = 0; i < text.length(); i++) {

char c = text.charAt(i);

if (c == ' ') {

sb.append(c);

newWord = true;

} else {

if (newWord) {

sb.append(toUpper(c));

newWord = false;

} else sb.append(toLower(c));

}

}

return sb.toString();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter text: ");

String text = sc.nextLine();

String manualUpper = toUpperCase(text);

String manualLower = toLowerCase(text);

String manualTitle = toTitleCase(text);

String builtInUpper = text.toUpperCase();

String builtInLower = text.toLowerCase();

System.out.println("\n------------------------------");

System.out.printf("%-15s | %s\n", "Conversion", "Result");

System.out.println("------------------------------");

System.out.printf("%-15s | %s\n", "Manual Upper", manualUpper);

System.out.printf("%-15s | %s\n", "ABuilt-in Upper", builtInUpper);

System.out.printf("%-15s | %s\n", "Manual Lower", manualLower);

System.out.printf("%-15s | %s\n", "Built-in Lower", builtInLower);

System.out.printf("%-15s | %s\n", "Manual Title", manualTitle);

System.out.println("------------------------------");

sc.close();

}

}

OUTPUT:

Enter text: hello

------------------------------

Conversion | Result

------------------------------

Manual Upper | HELLO

Built-in Upper | HELLO

Manual Lower | hello

Built-in Lower | hello

Manual Title | Hello

------------------------------

**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 2



● 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

SOLUTION:  
  
import java.util.\*;

public class StringPerformanceAnalysis {

public static long[] testStringConcat(int n) {

long start = System.currentTimeMillis();

String s = "";

for (int i = 0; i < n; i++) s += "A";

long end = System.currentTimeMillis();

return new long[]{end - start, s.length()};

}

public static long[] testStringBuilder(int n) {

long start = System.currentTimeMillis();

StringBuilder sb = new StringBuilder();

for (int i = 0; i < n; i++) sb.append("A");

long end = System.currentTimeMillis();

return new long[]{end - start, sb.length()};

}

public static long[] testStringBuffer(int n) {

long start = System.currentTimeMillis();

StringBuffer sb = new StringBuffer();

for (int i = 0; i < n; i++) sb.append("A");

long end = System.currentTimeMillis();

return new long[]{end - start, sb.length()};

}

public static String getMemoryEfficiency(long time, long length) {

double efficiency = (double) length / (time + 1);

if (efficiency > 10000) return "High";

else if (efficiency > 1000) return "Medium";

else return "Low";

}

public static void displayResults(long[] strResult, long[] sbResult, long[] sbufResult) {

System.out.println("\n-----------------------------------------------");

System.out.printf("%-15s | %-15s | %-15s\n", "Method", "Time (ms)", "Memory Efficiency");

System.out.println("-----------------------------------------------");

System.out.printf("%-15s | %-15d | %-15s\n", "String", strResult[0], getMemoryEfficiency(strResult[0], strResult[1]));

System.out.printf("%-15s | %-15d | %-15s\n", "StringBuilder", sbResult[0], getMemoryEfficiency(sbResult[0], sbResult[1]));

System.out.printf("%-15s | %-15d | %-15s\n", "StringBuffer", sbufResult[0], getMemoryEfficiency(sbufResult[0], sbufResult[1]));

System.out.println("-----------------------------------------------");

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of iterations: ");

int n = sc.nextInt();

long[] strResult = testStringConcat(n);

long[] sbResult = testStringBuilder(n);

long[] sbufResult = testStringBuffer(n);

displayResults(strResult, sbResult, sbufResult);

sc.close();

}

}

OUPUT:

Enter number of iterations: 5000

-----------------------------------------------

Method | Time (ms) | Memory Efficiency

-----------------------------------------------

String | 11 | Low

StringBuilder | 0 | Medium

StringBuffer | 1 | Medium

-----------------------------------------------

**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

3



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

SOLUTION:

import java.util.\*;

public class CaesarCipherDemo {

public static String encrypt(String text, int shift) {

StringBuilder encrypted = new StringBuilder();

for (int i = 0; i < text.length(); i++) {

char c = text.charAt(i);

if (c >= 'A' && c <= 'Z') {

encrypted.append((char) ((c - 'A' + shift) % 26 + 'A'));

} else if (c >= 'a' && c <= 'z') {

encrypted.append((char) ((c - 'a' + shift) % 26 + 'a'));

} else {

encrypted.append(c);

}

}

return encrypted.toString();

}

public static String decrypt(String text, int shift) {

return encrypt(text, 26 - (shift % 26));

}

public static void displayAsciiValues(String label, String text) {

System.out.print(label + " ASCII: ");

for (int i = 0; i < text.length(); i++) {

System.out.print((int) text.charAt(i));

if (i < text.length() - 1) System.out.print(" ");

}

System.out.println();

}

public static boolean validate(String original, String decrypted) {

return original.equals(decrypted);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter text to encrypt: ");

String text = sc.nextLine();

System.out.print("Enter shift value: ");

int shift = sc.nextInt();

String encrypted = encrypt(text, shift);

String decrypted = decrypt(encrypted, shift);

System.out.println("\n----------------------------");

System.out.println("Original Text: " + text);

displayAsciiValues("Original", text);

System.out.println("Encrypted Text: " + encrypted);

displayAsciiValues("Encrypted", encrypted);

System.out.println("Decrypted Text: " + decrypted);

displayAsciiValues("Decrypted", decrypted);

System.out.println("Decryption Valid: " + (validate(text, decrypted) ? "Yes" : "No"));

System.out.println("----------------------------");

sc.close();

}

}

OUTPUT:

Enter text to encrypt: this is java program

Enter shift value: 3

----------------------------

Original Text: this is java program

Original ASCII: 116 104 105 115 32 105 115 32 106 97 118 97 32 112 114 111 103 114 97 109

Encrypted Text: wklv lv mdyd surjudp

Encrypted ASCII: 119 107 108 118 32 108 118 32 109 100 121 100 32 115 117 114 106 117 100 112

Decrypted Text: this is java program

Decrypted ASCII: 116 104 105 115 32 105 115 32 106 97 118 97 32 112 114 111 103 114 97 109

Decryption Valid: Yes

----------------------------

**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

SOLUTION:

import java.util.\*;

public class EmailAnalyzer {

public static boolean isValidEmail(String email) {

int atPos = email.indexOf('@');

int lastAt = email.lastIndexOf('@');

if (atPos == -1 || atPos != lastAt) return false;

int dotPos = email.indexOf('.', atPos);

if (dotPos == -1) return false;

String username = email.substring(0, atPos);

String domain = email.substring(atPos + 1);

if (username.isEmpty() || domain.isEmpty()) return false;

return true;

}

public static String[] extractComponents(String email) {

int atPos = email.indexOf('@');

String username = email.substring(0, atPos);

String domain = email.substring(atPos + 1);

int dotPos = domain.lastIndexOf('.');

String domainName = dotPos != -1 ? domain.substring(0, dotPos) : domain;

String extension = dotPos != -1 ? domain.substring(dotPos + 1) : "";

return new String[]{username, domain, domainName, extension};

}

public static void displayTable(List<String> emails) {

System.out.println("\n--------------------------------------------------------------------------------");

System.out.printf("%-30s | %-15s | %-20s | %-15s | %-10s | %-8s\n",

"Email", "Username", "Domain", "Domain Name", "Extension", "Valid?");

System.out.println("--------------------------------------------------------------------------------");

for (String email : emails) {

if (isValidEmail(email)) {

String[] parts = extractComponents(email);

System.out.printf("%-30s | %-15s | %-20s | %-15s | %-10s | %-8s\n",

email, parts[0], parts[1], parts[2], parts[3], "Yes");

} else {

System.out.printf("%-30s | %-15s | %-20s | %-15s | %-10s | %-8s\n",

email, "-", "-", "-", "-", "No");

}

}

System.out.println("--------------------------------------------------------------------------------");

}

public static void analyzeStatistics(List<String> emails) {

int validCount = 0, invalidCount = 0, totalUsernameLength = 0;

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

for (String email : emails) {

if (isValidEmail(email)) {

validCount++;

String[] parts = extractComponents(email);

totalUsernameLength += parts[0].length();

domainCount.put(parts[1], domainCount.getOrDefault(parts[1], 0) + 1);

} else invalidCount++;

}

String mostCommonDomain = "-";

int maxCount = 0;

for (Map.Entry<String, Integer> entry : domainCount.entrySet()) {

if (entry.getValue() > maxCount) {

maxCount = entry.getValue();

mostCommonDomain = entry.getKey();

}

}

double avgUsernameLength = validCount > 0 ? (double) totalUsernameLength / validCount : 0;

System.out.println("\n================= Email Statistics =================");

System.out.println("Total Emails: " + emails.size());

System.out.println("Valid Emails: " + validCount);

System.out.println("Invalid Emails: " + invalidCount);

System.out.println("Most Common Domain: " + mostCommonDomain);

System.out.printf("Average Username Len: %.2f\n", avgUsernameLength);

System.out.println("====================================================");

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

List<String> emails = new ArrayList<>();

System.out.print("Enter number of email addresses: ");

int n = sc.nextInt();

sc.nextLine();

for (int i = 1; i <= n; i++) {

System.out.print("Enter email " + i + ": ");

emails.add(sc.nextLine().trim());

}

displayTable(emails);

analyzeStatistics(emails);

sc.close();

}

}

OUTPUT:

Enter number of email addresses: 3

Enter email 1: ar9876@srmist.edu.in

Enter email 2: sv2735@srmist.edu.in

Enter email 3: kk4567@srmist.edu.in

--------------------------------------------------------------------------------

Email | Username | Domain | Domain Name | Extension | Valid?

--------------------------------------------------------------------------------

ar9876@srmist.edu.in | ar9876 | srmist.edu.in | srmist.edu | in | Yes

sv2735@srmist.edu.in | sv2735 | srmist.edu.in | srmist.edu | in | Yes

kk4567@srmist.edu.in | kk4567 | srmist.edu.in | srmist.edu | in | Yes

--------------------------------------------------------------------------------

================= Email Statistics =================

Total Emails: 3

Valid Emails: 3

Invalid Emails: 0

Most Common Domain: srmist.edu.in

Average Username Len: 6.00

====================================================

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

4



**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

import java.util.\*;

public class TextFormatter {

// Split text into words manually without using split()

public static List<String> getWords(String text) {

List<String> words = new ArrayList<>();

int start = 0;

for (int i = 0; i < text.length(); i++) {

if (text.charAt(i) == ' ') {

if (start != i) {

words.add(text.substring(start, i));

}

start = i + 1;

}

}

if (start < text.length()) {

words.add(text.substring(start));

}

return words;

}

// Left-justify text using StringBuilder

public static List<String> justifyText(List<String> words, int width) {

List<String> lines = new ArrayList<>();

int index = 0;

while (index < words.size()) {

int lineLength = words.get(index).length();

int last = index + 1;

while (last < words.size()) {

if (lineLength + words.get(last).length() + 1 > width) break;

lineLength += words.get(last).length() + 1;

last++;

}

StringBuilder sb = new StringBuilder();

int gaps = last - index - 1;

if (last == words.size() || gaps == 0) {

for (int i = index; i < last; i++) {

sb.append(words.get(i)).append(" ");

}

sb.deleteCharAt(sb.length() - 1);

while (sb.length() < width) sb.append(" ");

} else {

int totalSpaces = width - lineLength + gaps;

int spaceBetween = totalSpaces / gaps;

int extraSpaces = totalSpaces % gaps;

for (int i = index; i < last - 1; i++) {

sb.append(words.get(i));

for (int s = 0; s < spaceBetween + 1; s++) sb.append(" ");

if (extraSpaces-- > 0) sb.append(" ");

}

sb.append(words.get(last - 1));

}

lines.add(sb.toString());

index = last;

}

return lines;

}

// Center-align text using StringBuilder

public static List<String> centerAlign(List<String> words, int width) {

List<String> lines = new ArrayList<>();

StringBuilder sb = new StringBuilder();

for (String word : words) {

if (sb.length() + word.length() + 1 > width) {

String line = sb.toString().trim();

int padding = (width - line.length()) / 2;

StringBuilder centered = new StringBuilder();

for (int i = 0; i < padding; i++) centered.append(" ");

centered.append(line);

while (centered.length() < width) centered.append(" ");

lines.add(centered.toString());

sb.setLength(0);

}

sb.append(word).append(" ");

}

if (sb.length() > 0) {

String line = sb.toString().trim();

int padding = (width - line.length()) / 2;

StringBuilder centered = new StringBuilder();

for (int i = 0; i < padding; i++) centered.append(" ");

centered.append(line);

while (centered.length() < width) centered.append(" ");

lines.add(centered.toString());

}

return lines;

}

// Performance comparison: StringBuilder vs String concatenation

public static void comparePerformance(List<String> words, int width) {

long start1 = System.nanoTime();

justifyText(words, width);

long end1 = System.nanoTime();

long sbTime = end1 - start1;

long start2 = System.nanoTime();

String result = "";

int lineLen = 0;

for (String word : words) {

if (lineLen + word.length() + 1 > width) {

result += "\n";

lineLen = 0;

}

result += word + " ";

lineLen += word.length() + 1;

}

long end2 = System.nanoTime();

long concatTime = end2 - start2;

System.out.println("\n============= Performance Comparison =============");

System.out.println("StringBuilder Time: " + sbTime + " ns");

System.out.println("String Concatenation Time: " + concatTime + " ns");

if (sbTime < concatTime)

System.out.println("Winner: StringBuilder (Faster)");

else

System.out.println("Winner: String Concatenation (Faster)");

System.out.println("==================================================");

}

// Display formatted text with line numbers & character count

public static void displayFormattedText(List<String> lines, String title) {

System.out.println("\n========= " + title + " =========");

int lineNo = 1;

for (String line : lines) {

System.out.printf("Line %2d (%2d chars): %s\n", lineNo++, line.length(), line);

}

System.out.println("===================================");

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter text to format: ");

String text = sc.nextLine();

System.out.print("Enter desired line width: ");

int width = sc.nextInt();

List<String> words = getWords(text);

List<String> justified = justifyText(words, width);

List<String> centered = centerAlign(words, width);

System.out.println("\n========= Original Text =========");

System.out.println(text);

displayFormattedText(justified, "Justified Text");

displayFormattedText(centered, "Center Aligned Text");

comparePerformance(words, width);

sc.close();

}

}

OUTPUT:

Enter text to format: jaVA ProgRam

Enter desired line width: 4

========= Original Text =========

jaVA ProgRam

========= Justified Text =========

Line 1 ( 4 chars): jaVA

Line 2 ( 7 chars): ProgRam

===================================

========= Center Aligned Text =========

Line 1 ( 4 chars):

Line 2 ( 4 chars): jaVA

Line 3 ( 7 chars): ProgRam

===================================

============= Performance Comparison =============

StringBuilder Time: 26200 ns

String Concatenation Time: 5450600 ns

Winner: StringBuilder (Faster)

==================================================

5