Week-2: Java Practice programs

Name : Penti Hrishi

Reg no : RA2411028010092

Section : V1

1.Create a program that demonstrates common String methods for text analysis and manipulation.

Program:

public class StringBuiltInMethods { public static void main(String[] args) {

String sampleText = " Java Programming is Fun and Challenging! ";

System.out.println("Original length (including spaces): " + sampleText.length());

String trimmedText = sampleText.trim();

System.out.println("Trimmed length: " + trimmedText.length());

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

String substring = trimmedText.substring(trimmedText.indexOf("Programming"), trimmedText.indexOf("Programming") + "Programming".length());

System.out.println("Extracted substring: " + substring);

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

System.out.println("Contains 'Java': " + trimmedText.contains("Java")); System.out.println("Starts with 'Java': " + trimmedText.startsWith("Java"));

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

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

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

int vowelCount = countVowels(trimmedText);

System.out.println("Number of vowels: " + vowelCount);

System.out.print("Positions of character 'a': "); findAllOccurrences(trimmedText, 'a');

}

public static int countVowels(String text) { int count = 0;

String vowels = "aeiouAEIOU"; for (int i = 0; i < text.length(); i++) { char ch = text.charAt(i); if (vowels.indexOf(ch) != -1) { count++;

}

}

return count;

}

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:

Original length (including spaces): 42

Trimmed length: 40

Character at index 5: P

Extracted substring: Programming

Index of 'Fun': 20

Contains 'Java': true

Starts with 'Java': true

Ends with '!': true

Uppercase: JAVA PROGRAMMING IS FUN AND CHALLENGING!

Lowercase: java programming is fun and challenging!

Number of vowels: 11

Positions of character 'a': 1 3 10 24 30

2. Create a program that demonstrates ASCII character manipulation and conversion.

Program:

//2-practice pro -2 import java.util.Scanner;

public class ASCIIProcessor { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = scanner.nextLine();

for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); int ascii = (int) ch;

System.out.println("Character: '" + ch + "' ASCII: " + ascii);

String type = classifyCharacter(ch); System.out.println("Type: " + type); if (type.equals("Uppercase Letter")) { char lower = toggleCase(ch);

System.out.println("Lowercase: '" + lower + "' ASCII: " + (int) lower);

System.out.println("Difference: " + ((int) lower - ascii));

} else if (type.equals("Lowercase Letter")) { char upper = toggleCase(ch); System.out.println("Uppercase: '" + upper + "' ASCII: " + (int) upper);

System.out.println("Difference: " + (ascii - (int) upper));

}

System.out.println();

}

System.out.println("ASCII Table (32-126):"); displayASCIITable(32, 126);

System.out.print("Enter text for Caesar cipher: ");

String text = scanner.nextLine(); System.out.print("Enter shift value: "); int shift = scanner.nextInt();

String ciphered = caesarCipher(text, shift);

System.out.println("Ciphered text: " + ciphered);

scanner.close();

}

public static String classifyCharacter(char ch) { if (ch >= 'A' && ch <= 'Z') return "Uppercase Letter"; else if (ch >= 'a' && ch <= 'z') return "Lowercase Letter"; else if (ch >= '0' && ch <= '9') return "Digit"; else return "Special Character";

}

public static char toggleCase(char ch) { if (ch >= 'A' && ch <= 'Z') return (char)(ch + 32); else if (ch >= 'a' && ch <= 'z') return (char)(ch - 32); else return ch;

}

public static String caesarCipher(String text, int shift) { StringBuilder result = new StringBuilder(); for (int i = 0; i < text.length(); i++) { char ch = text.charAt(i); if (ch >= 'A' && ch <= 'Z') { char shifted = (char) ('A' + (ch - 'A' + shift + 26) % 26); result.append(shifted); } else if (ch >= 'a' && ch <= 'z') { char shifted = (char) ('a' + (ch - 'a' + shift + 26) % 26); result.append(shifted);

} else { result.append(ch);

}

}

return result.toString();

}

public static void displayASCIITable(int start, int end) { for (int i = start; i <= end; i++) {

System.out.printf("%3d : %c ", i, (char)i);

if ((i - start + 1) % 6 == 0) System.out.println();

}

System.out.println();

}

public static int[] stringToASCII(String text) { int[] asciiArr = new int[text.length()]; for (int i = 0; i < text.length(); i++) { asciiArr[i] = (int) text.charAt(i);

}

return asciiArr;

}

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: Hlo

Character: 'H' ASCII: 72

Type: Uppercase Letter

Lowercase: 'h' ASCII: 104

Difference: 32

Character: 'l' ASCII: 108

Type: Lowercase Letter

Uppercase: 'L' ASCII: 76

Difference: 32

Character: 'o' ASCII: 111

Type: Lowercase Letter

Uppercase: 'O' ASCII: 79

Difference: 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 : ~

Enter text for Caesar cipher: l

Enter shift value: 4

Ciphered text: p

3. Create a performance comparison program that demonstrates the differences between

String, StringBuilder, and StringBuffer.

Program:

//2-practice pro -3 import java.util.\*;

public class Stringchange { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a sentence with mixed formatting:"); String input = scanner.nextLine();

String trimmed = input.trim();

System.out.println("Trimmed: '" + trimmed + "'");

String replacedSpaces = trimmed.replace(' ', '\_');

System.out.println("Spaces replaced with underscores: " + replacedSpaces);

String removedDigits = replacedSpaces.replaceAll("\\d", "");

System.out.println("Digits removed: " + removedDigits);

String[] words = removedDigits.split("\_");

System.out.println("Split into words: " + Arrays.toString(words));

String joined = String.join(" | ", words);

System.out.println("Joined with ' | ': " + joined);

String noPunctuation = removePunctuation(trimmed);

System.out.println("Without punctuation: " + noPunctuation);

String capitalized = capitalizeWords(noPunctuation);

System.out.println("Capitalized each word: " + capitalized);

String reversed = reverseWordOrder(noPunctuation);

System.out.println("Reversed word order: " + reversed);

System.out.println("Word frequencies:"); countWordFrequency(noPunctuation);

scanner.close();

}

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

}

public static String capitalizeWords(String text) {

String[] words = text.split("\\s+"); StringBuilder sb = new StringBuilder(); for (String w : words) { if (w.length() > 0) { sb.append(Character.toUpperCase(w.charAt(0)));

if (w.length() > 1) { sb.append(w.substring(1).toLowerCase());

}

sb.append(" ");

}

}

return sb.toString().trim();

}

public static String reverseWordOrder(String text) {

String[] words = text.split("\\s+"); StringBuilder sb = new StringBuilder(); for (int i = words.length - 1; i >= 0; i--) { sb.append(words[i]); if (i != 0) sb.append(" ");

}

return sb.toString();

}

public static void countWordFrequency(String text) {

String[] words = text.toLowerCase().split("\\s+");

Map<String, Integer> freqMap = new LinkedHashMap<>();

for (String w : words) { freqMap.put(w, freqMap.getOrDefault(w, 0) + 1);

}

freqMap.forEach((k, v) -> System.out.println(k + ": " + v));

}

}

Output:

Enter a sentence with mixed formatting:

Hlo world @ srm

Trimmed: 'Hlo world @ srm'

Spaces replaced with underscores: Hlo\_world\_@\_srm

Digits removed: Hlo\_world\_@\_srm

Split into words: [Hlo, world, @, srm]

Joined with ' | ': Hlo | world | @ | srm

Without punctuation: Hlo world srm

Capitalized each word: Hlo World Srm

Reversed word order: srm world Hlo

Word frequencies:

hlo: 1 world: 1 srm: 1

4. Create a comprehensive string utility application that combines all learned concepts including built-in methods, ASCII manipulation, and performance optimization.

Program: //2-practice pro-4 import java.util.\*;

public class StringUtilityApp { private static StringBuilder output = new StringBuilder(); private static Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

System.out.println("=== STRING UTILITY APPLICATION ===");

boolean exit = false; while (!exit) {

System.out.println("\nSelect an option:");

System.out.println("1. Text Analysis");

System.out.println("2. String Transformation");

System.out.println("3. ASCII Operations");

System.out.println("4. Performance Testing");

System.out.println("5. String Comparison Analysis");

System.out.println("6. Custom String Algorithms");

System.out.println("7. Exit");

int choice = getIntInput("Enter choice (1-7): ");

switch (choice) {

case 1:

System.out.print("Enter text for analysis:\n> "); scanner.nextLine(); // consume newline

String text1 = scanner.nextLine(); performTextAnalysis(text1);

break; case 2:

System.out.print("Enter text for transformations:\n> "); scanner.nextLine(); // consume newline

String text2 = scanner.nextLine();

System.out.println("Enter operations separated by comma (trim, uppercase, lowercase, replaceSpaces, removeDigits):");

String opsLine = scanner.nextLine(); String[] ops = opsLine.split(","); for (int i = 0; i < ops.length; i++) ops[i] = ops[i].trim();

String transformed = performTransformations(text2, ops); System.out.println("Transformed text: " + transformed);

break; case 3:

System.out.print("Enter text for ASCII operations:\n> "); scanner.nextLine(); // consume newline String text3 = scanner.nextLine(); performASCIIOperations(text3);

break; case 4:

int iters = getIntInput("Enter number of iterations for performance test: "); performPerformanceTest(iters);

break; case 5:

System.out.println("Enter strings to compare (comma separated):"); scanner.nextLine(); // consume newline

String input = scanner.nextLine(); String[] strings = input.split(","); for (int i = 0; i < strings.length; i++) strings[i] = strings[i].trim(); performComparisonAnalysis(strings);

break; case 6:

System.out.print("Enter text for custom algorithms:\n> "); scanner.nextLine(); // consume newline String text6 = scanner.nextLine(); performCustomAlgorithms(text6);

break; case 7:

exit = true;

System.out.println("Exiting application.");

break; default:

System.out.println("Invalid choice, try again.");

}

}

scanner.close();

}

public static int getIntInput(String prompt) { while (true) {

System.out.print(prompt);

try {

return Integer.parseInt(scanner.nextLine());

} catch (NumberFormatException e) {

System.out.println("Please enter a valid number.");

}

}

}

// 1. Text Analysis public static void performTextAnalysis(String text) { output.setLength(0); output.append("Text Analysis:\n"); output.append("Length (chars): ").append(text.length()).append("\n");

// Word count

String[] words = text.trim().split("\\s+"); output.append("Word count: ").append(words.length).append("\n");

// Sentence count (simple split on ., !, ?) String[] sentences = text.split("[.!?]+"); output.append("Sentence count: ").append(sentences.length).append("\n"); // Paragraph count (split on double newlines) String[] paragraphs = text.split("\\n\\n+"); output.append("Paragraph count: ").append(paragraphs.length).append("\n");

// Character frequency

Map<Character, Integer> freq = new HashMap<>(); for (char ch : text.toCharArray()) { freq.put(ch, freq.getOrDefault(ch, 0) + 1);

}

output.append("Character frequencies:\n"); freq.entrySet().stream()

.sorted(Map.Entry.comparingByKey())

.forEach(e -> output.append("'" + e.getKey() + "': " + e.getValue() + "\n"));

// Most common words

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

for (String w : words) {

String lw = w.toLowerCase().replaceAll("\\W", ""); if (!lw.isEmpty()) { wordFreq.put(lw, wordFreq.getOrDefault(lw, 0) + 1);

}

}

output.append("Most common words:\n"); wordFreq.entrySet().stream()

.sorted((a,b) -> b.getValue() - a.getValue())

.limit(5)

.forEach(e -> output.append(e.getKey() + ": " + e.getValue() + "\n"));

displayResults();

}

// 2. String Transformation public static String performTransformations(String text, String[] operations) { StringBuilder sb = new StringBuilder(text); for (String op : operations) { switch (op.toLowerCase()) { case "trim":

text = text.trim(); sb = new StringBuilder(text); break; case "uppercase":

text = sb.toString().toUpperCase(); sb = new StringBuilder(text); break; case "lowercase":

text = sb.toString().toLowerCase(); sb = new StringBuilder(text); break;

case "replacespaces":

text = sb.toString().replace(' ', '\_'); sb = new StringBuilder(text); break;

case "removedigits":

text = sb.toString().replaceAll("\\d", ""); sb = new StringBuilder(text); break; default:

System.out.println("Unknown operation: " + op);

}

}

return sb.toString();

}

// 3. ASCII Operations public static void performASCIIOperations(String text) { output.setLength(0); output.append("ASCII Operations:\n"); for (char ch : text.toCharArray()) { int code = (int) ch; output.append("Char: '").append(ch).append("' ASCII: ").append(code); output.append(" Type: ").append(classifyCharacter(ch));

if (Character.isLetter(ch)) { char upper = Character.toUpperCase(ch); char lower = Character.toLowerCase(ch);

output.append(" | Upper: '").append(upper).append("'(").append((int) upper).append(")");

output.append(" Lower: '").append(lower).append("'(").append((int) lower).append(")");

output.append(" Diff: ").append(Math.abs((int)upper - (int)lower));

}

output.append("\n");

} displayResults();

// Simple Caesar cipher demo

System.out.println("Enter shift amount for Caesar cipher:"); int shift = getIntInput("> ");

String encrypted = caesarCipher(text, shift);

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

String decrypted = caesarCipher(encrypted, -shift);

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

}

public static String classifyCharacter(char ch) { 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";

}

public static String caesarCipher(String text, int shift) { StringBuilder result = new StringBuilder(); for (char ch : text.toCharArray()) { if (Character.isLetter(ch)) {

char base = Character.isUpperCase(ch) ? 'A' : 'a'; char shifted = (char) ((ch - base + shift + 26) % 26 + base); result.append(shifted);

} else { result.append(ch);

}

}

return result.toString();

}

/ 4. Performance Testing public static void performPerformanceTest(int iterations) { output.setLength(0); output.append("Performance Test with ").append(iterations).append(" iterations\n");

long start, end;

// String concat start = System.nanoTime(); String s = ""; for (int i = 0; i < iterations; i++) { s += "x";

}

end = System.nanoTime(); output.append("String concat time: ").append(end - start).append(" ns\n"); // StringBuilder concat start = System.nanoTime();

StringBuilder sb = new StringBuilder(); for (int i = 0; i < iterations; i++) { sb.append("x");

}

end = System.nanoTime(); output.append("StringBuilder append time: ").append(end - start).append(" ns\n");

// StringBuffer concat start = System.nanoTime();

StringBuffer sbf = new StringBuffer(); for (int i = 0; i < iterations; i++) { sbf.append("x");

}

end = System.nanoTime(); output.append("StringBuffer append time: ").append(end - start).append(" ns\n");

displayResults();

}

// 5. String Comparison Analysis public static void performComparisonAnalysis(String[] strings) { output.setLength(0); output.append("String Comparison Analysis:\n"); for (int i = 0; i < strings.length; i++) {

for (int j = i + 1; j < strings.length; j++) {

String s1 = strings[i]; String s2 = strings[j];

output.append("Comparing '").append(s1).append("' and

'").append(s2).append("':\n"); output.append("== : ").append(s1 == s2).append("\n"); output.append("equals(): ").append(s1.equals(s2)).append("\n");

output.append("equalsIgnoreCase():

").append(s1.equalsIgnoreCase(s2)).append("\n"); output.append("compareTo(): ").append(s1.compareTo(s2)).append("\n");

output.append("compareToIgnoreCase():

").append(s1.compareToIgnoreCase(s2)).append("\n\n");

}

}

displayResults();

}

// 6. Custom String Algorithms public static void performCustomAlgorithms(String text) { output.setLength(0); output.append("Custom Algorithms:\n"); output.append("Is palindrome? ").append(isPalindrome(text) ? "Yes" : "No").append("\n");

System.out.println("Enter another string to check for anagram:"); String other = scanner.nextLine();

output.append("Is anagram with '").append(other).append("'? ").append(isAnagram(text, other) ? "Yes" : "No").append("\n");

System.out.println("Enter pattern to find occurrences:");

String pattern = scanner.nextLine();

List<Integer> positions = findPatternOccurrences(text, pattern);

output.append("Pattern '").append(pattern).append("' found at positions: ").append(positions).append("\n");

displayResults();

}

public static boolean isPalindrome(String text) {

String filtered = text.replaceAll("[^a-zA-Z0-9]", "").toLowerCase(); return filtered.equals(new StringBuilder(filtered).reverse().toString());

}

public static boolean isAnagram(String s1, String s2) { char[] a1 = s1.toLowerCase().replaceAll("\\s+", "").toCharArray(); char[] a2 = s2.toLowerCase().replaceAll("\\s+", "").toCharArray();

Arrays.sort(a1); Arrays.sort(a2); return Arrays.equals(a1, a2);

}

public static List<Integer> findPatternOccurrences(String text, String pattern) { List<Integer> positions = new ArrayList<>(); for (int i = 0; i <= text.length() - pattern.length(); i++) {

if (text.regionMatches(i, pattern, 0, pattern.length())) { positions.add(i);

}

} return positions;

}

// Display accumulated results public static void displayResults() {

System.out.println("\n" + output.toString());

}

}

Output:

=== STRING UTILITY APPLICATION ===

Select an option:

1. Text Analysis
2. String Transformation
3. ASCII Operations
4. Performance Testing
5. String Comparison Analysis
6. Custom String Algorithms
7. Exit Enter choice (1-7): 1

Enter text for analysis:

> Hlo wrlds can i join with u

Text Analysis:

Length (chars): 17 Word count: 5

Sentence count: 1 Paragraph count: 1

Character frequencies:

' ': 4

'a': 1

'c': 1

'h': 1

'i': 3

'j': 1

'n': 2

'o': 1

't': 1

'u': 1

'w': 1

Most common words:

can: 1 with: 1 u: 1 i: 1

join: 1 Select an option:

1. Text Analysis
2. String Transformation
3. ASCII Operations
4. Performance Testing
5. String Comparison Analysis
6. Custom String Algorithms
7. Exit

Enter choice (1-7): 3

Enter text for ASCII operations:

> Good Afternoon

Hlo

ASCII Operations:

Char: 'H' ASCII: 72 Type: Uppercase Letter | Upper: 'H'(72) Lower: 'h'(104) Diff: 32

Char: 'l' ASCII: 108 Type: Lowercase Letter | Upper: 'L'(76) Lower: 'l'(108) Diff: 32

Char: 'o' ASCII: 111 Type: Lowercase Letter | Upper: 'O'(79) Lower: 'o'(111) Diff: 32

Enter shift amount for Caesar cipher:

> 4

Encrypted: Lps

Decrypted: Hlo

Select an option:

1. Text Analysis
2. String Transformation
3. ASCII Operations
4. Performance Testing
5. String Comparison Analysis
6. Custom String Algorithms
7. Exit Enter choice (1-7): 7

Exiting application.