Week-2 Assignment

Q1) Built-In String Methods - Basic Operations

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

manipulation.

// TODO: Use built-in methods to perform the following operations:

// 1. Display original string length including spaces

// 2. Remove leading and trailing spaces, show new length

// 3. Find and display the character at index 5

// 4. Extract substring "Programming" from the text

// 5. Find the index of the word "Fun"

// 6. Check if the string contains "Java" (case-sensitive)

// 7. Check if the string starts with "Java" (after trimming)

// 8. Check if the string ends with an exclamation mark

// 9. Convert the entire string to uppercase

// 10. Convert the entire string to lowercase

// TODO: Create a method that counts vowels using charAt()

// TODO: Create a method that finds all occurrences of a character

// TODO: Display all results in a formatted manner

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("Original String: \"" + sampleText + "\"");

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

// 2. Remove leading and trailing spaces, show new length

String trimmed = sampleText.trim();

System.out.println("2. Trimmed String: \"" + trimmed + "\"");

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

// 3. Find and display the character at index 5

if (sampleText.length() > 5) {

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

} else {

System.out.println("3. String too short for index 5.");

}

// 4. Extract substring "Programming" from the text

int progStart = sampleText.indexOf("Programming");

String programming = (progStart != -1) ? sampleText.substring(progStart, progStart + "Programming".length()) : "Not found";

System.out.println("4. Substring \"Programming\": " + programming);

// 5. Find the index of the word "Fun"

int funIndex = sampleText.indexOf("Fun");

System.out.println("5. Index of \"Fun\": " + funIndex);

// 6. Check if the string contains "Java" (case-sensitive)

boolean containsJava = sampleText.contains("Java");

System.out.println("6. Contains \"Java\"? " + containsJava);

// 7. Check if the string starts with "Java" (after trimming)

boolean startsWithJava = trimmed.startsWith("Java");

System.out.println("7. Starts with \"Java\" after trim? " + startsWithJava);

// 8. Check if the string ends with an exclamation mark

boolean endsWithExclamation = trimmed.endsWith("!");

System.out.println("8. Ends with '!'? " + endsWithExclamation);

// 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());

// Count vowels

int vowelCount = countVowels(sampleText);

System.out.println("11. Number of vowels: " + vowelCount);

// Find all occurrences of a character

char targetChar = 'a';

System.out.print("12. All positions of '" + targetChar + "': ");

findAllOccurrences(sampleText, targetChar);

}

// 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++) {

if (vowels.indexOf(text.charAt(i)) != -1) {

count++;

}

}

return count;

}

// 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) {

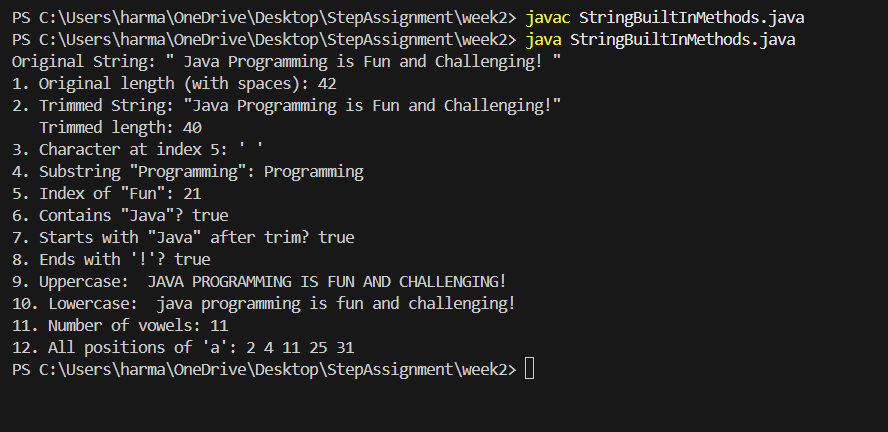
System.out.print("None");

}

System.out.println();

}

}



Q2) String Manipulation Methods

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

import java.util.Scanner;

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

// TODO: Process the input using the following methods:

// 1. trim() - Remove extra spaces

// 2. replace() - Replace all spaces with underscores

// 3. replaceAll() - Remove all digits using regex

// 4. split() - Split sentence into words array

// 5. join() - Rejoin words with " | " separator

// TODO: Create additional processing methods:

// - Remove all punctuation

// - Capitalize first letter of each word

// - Reverse the order of words

// - Count word frequency

import java.util.\*;

public class StringManipulation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Ask user to enter a sentence with mixed formatting

System.out.print("Enter a sentence with mixed formatting: ");

String input = scanner.nextLine();

// 1. trim() - Remove extra spaces

String trimmed = input.trim();

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

// 2. replace() - Replace all spaces with underscores

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

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

// 3. replaceAll() - Remove all digits using regex

String noDigits = trimmed.replaceAll("\\d", "");

System.out.println("Removed digits: " + noDigits);

// 4. split() - Split sentence into words array

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

System.out.println("Words array: " + Arrays.toString(words));

// 5. join() - Rejoin words with " | " separator

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

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

// Additional processing

String noPunct = removePunctuation(trimmed);

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

String capitalized = capitalizeWords(trimmed);

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

String reversed = reverseWordOrder(trimmed);

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

System.out.println("Word frequency:");

countWordFrequency(trimmed);

scanner.close();

}

// Method to remove punctuation

public static String removePunctuation(String text) {

return text.replaceAll("\\p{Punct}", "");

}

// Method to capitalize each word

public static String capitalizeWords(String text) {

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

StringBuilder sb = new StringBuilder();

for (String word : words) {

if (word.length() > 0) {

sb.append(Character.toUpperCase(word.charAt(0)));

if (word.length() > 1) {

sb.append(word.substring(1).toLowerCase());

}

sb.append(" ");

}

}

return sb.toString().trim();

}

// Method to reverse word order

public static String reverseWordOrder(String text) {

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

Collections.reverse(Arrays.asList(words));

return String.join(" ", words);

}

// Method to count word frequency

public static void countWordFrequency(String text) {

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

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

for (String word : words) {

word = word.replaceAll("\\p{Punct}", "");

if (word.isEmpty()) continue;

freq.put(word, freq.getOrDefault(word, 0) + 1);

}

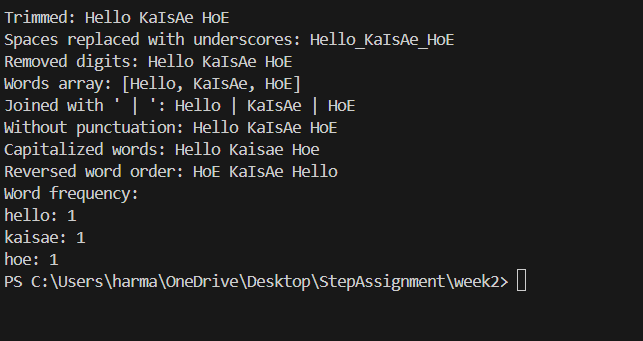
for (Map.Entry<String, Integer> entry : freq.entrySet()) {

System.out.println(entry.getKey() + ": " + entry.getValue());

}

}

}



Q3) 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

// TODO: For each character in the string:

// 1. Display the character and its ASCII code

// 2. Determine if it's uppercase, lowercase, digit, or special

character

// 3. If letter, show both upper and lower case versions with ASCII

codes

// 4. Calculate the difference between upper and lower case ASCII

values

// TODO: Create ASCII art using character codes

// TODO: Implement a simple Caesar cipher using ASCII manipulation

scanner.close();

}

3

// TODO: Method to classify character type

public static String classifyCharacter(char ch) {

// Return "Uppercase Letter", "Lowercase Letter", "Digit", or

"Special Character"

// Your code here

}

// 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

// Your code here

}

// TODO: Method to implement Caesar cipher

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

// Shift each letter by 'shift' positions in ASCII

// Your code here

}

// TODO: Method to create ASCII table for a range

public static void displayASCIITable(int start, int end) {

// Display ASCII codes and corresponding characters

// Your code here

}

// TODO: Method to convert string to ASCII array

public static int[] stringToASCII(String text) {

// Your code here

}

// TODO: Method to convert ASCII array back to string

public static String asciiToString(int[] asciiValues) {

// Your code here

}

}

import java.util.Scanner;

public class ASCIIProcessor {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Ask user to enter a string

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

String input = scanner.nextLine();

// For each character in the string:

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

char ch = input.charAt(i);

int ascii = (int) ch;

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

// 2. Determine type

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 + "' (ASCII: " + (int) upper + ")");

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

// 4. Calculate the difference between upper and lower case ASCII values

System.out.println("ASCII difference (upper - lower): " + ((int) upper - (int) lower));

}

System.out.println();

}

// ASCII art using character codes

System.out.print("ASCII Art: ");

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

System.out.print((int) input.charAt(i) + " ");

}

System.out.println();

// Caesar cipher

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

int shift = scanner.nextInt();

scanner.nextLine(); // consume newline

String ciphered = caesarCipher(input, shift);

System.out.println("Caesar Cipher: " + ciphered);

// Display ASCII table for a range

System.out.print("Display ASCII table from: ");

int start = scanner.nextInt();

System.out.print("to: ");

int end = scanner.nextInt();

displayASCIITable(start, end);

// Convert string to ASCII array and back

int[] asciiArr = stringToASCII(input);

System.out.print("ASCII Array: ");

for (int val : asciiArr) System.out.print(val + " ");

System.out.println();

String fromAscii = asciiToString(asciiArr);

System.out.println("String from ASCII array: " + fromAscii);

scanner.close();

}

// Method to classify character type

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";

}

// Method to convert case using ASCII manipulation

public static char toggleCase(char ch) {

if (Character.isUpperCase(ch)) return (char) (ch + 32);

if (Character.isLowerCase(ch)) return (char) (ch - 32);

return ch;

}

// Method to implement Caesar cipher

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

StringBuilder sb = new StringBuilder();

for (char ch : text.toCharArray()) {

if (Character.isUpperCase(ch)) {

sb.append((char) ('A' + (ch - 'A' + shift + 26) % 26));

} else if (Character.isLowerCase(ch)) {

sb.append((char) ('a' + (ch - 'a' + shift + 26) % 26));

} else {

sb.append(ch);

}

}

return sb.toString();

}

// Method to create ASCII table for a range

public static void displayASCIITable(int start, int end) {

System.out.println("ASCII Table:");

for (int i = start; i <= end; i++) {

System.out.println(i + " : '" + (char) i + "'");

}

}

// 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;

}

// 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();

}

}

PS C:\Users\harma\OneDrive\Desktop\StepAssignment\week2> javac ASCIIProcessor.java

PS C:\Users\harma\OneDrive\Desktop\StepAssignment\week2> java ASCIIProcessor.java

Enter a string: HELLO JI KAISAE HO

Character: 'H' | ASCII: 72

Type: Uppercase Letter

Uppercase: 'H' (ASCII: 72)

Lowercase: 'h' (ASCII: 104)

ASCII difference (upper - lower): -32

Character: 'E' | ASCII: 69

Type: Uppercase Letter

Uppercase: 'E' (ASCII: 69)

Lowercase: 'e' (ASCII: 101)

ASCII difference (upper - lower): -32

Character: 'L' | ASCII: 76

Type: Uppercase Letter

Uppercase: 'L' (ASCII: 76)

Lowercase: 'l' (ASCII: 108)

ASCII difference (upper - lower): -32

Character: 'L' | ASCII: 76

Type: Uppercase Letter

Uppercase: 'L' (ASCII: 76)

Lowercase: 'l' (ASCII: 108)

ASCII difference (upper - lower): -32

Character: 'O' | ASCII: 79

Type: Uppercase Letter

Uppercase: 'O' (ASCII: 79)

Lowercase: 'o' (ASCII: 111)

ASCII difference (upper - lower): -32

Character: ' ' | ASCII: 32

Type: Special Character

Character: 'J' | ASCII: 74

Type: Uppercase Letter

Uppercase: 'J' (ASCII: 74)

Lowercase: 'j' (ASCII: 106)

ASCII difference (upper - lower): -32

Character: 'I' | ASCII: 73

Type: Uppercase Letter

Uppercase: 'I' (ASCII: 73)

Lowercase: 'i' (ASCII: 105)

ASCII difference (upper - lower): -32

Character: ' ' | ASCII: 32

Type: Special Character

Character: 'K' | ASCII: 75

Type: Uppercase Letter

Uppercase: 'K' (ASCII: 75)

Lowercase: 'k' (ASCII: 107)

ASCII difference (upper - lower): -32

Character: 'A' | ASCII: 65

Type: Uppercase Letter

Uppercase: 'A' (ASCII: 65)

Lowercase: 'a' (ASCII: 97)

ASCII difference (upper - lower): -32

Character: 'I' | ASCII: 73

Type: Uppercase Letter

Uppercase: 'I' (ASCII: 73)

Lowercase: 'i' (ASCII: 105)

ASCII difference (upper - lower): -32

Character: 'S' | ASCII: 83

Type: Uppercase Letter

Uppercase: 'S' (ASCII: 83)

Lowercase: 's' (ASCII: 115)

ASCII difference (upper - lower): -32

Character: 'A' | ASCII: 65

Type: Uppercase Letter

Uppercase: 'A' (ASCII: 65)

Lowercase: 'a' (ASCII: 97)

ASCII difference (upper - lower): -32

Character: 'E' | ASCII: 69

Type: Uppercase Letter

Uppercase: 'E' (ASCII: 69)

Lowercase: 'e' (ASCII: 101)

ASCII difference (upper - lower): -32

Character: ' ' | ASCII: 32

Type: Special Character

Character: 'H' | ASCII: 72

Type: Uppercase Letter

Uppercase: 'H' (ASCII: 72)

Lowercase: 'h' (ASCII: 104)

ASCII difference (upper - lower): -32

Character: 'O' | ASCII: 79

Type: Uppercase Letter

Uppercase: 'O' (ASCII: 79)

Lowercase: 'o' (ASCII: 111)

ASCII difference (upper - lower): -32

Q4) PS C:\Users\harma\OneDrive\Desktop\StepAssignment\week2> javac ASCIIProcessor.java

PS C:\Users\harma\OneDrive\Desktop\StepAssignment\week2> java ASCIIProcessor.java

Enter a string: HELLO JI KAISAE HO

Character: 'H' | ASCII: 72

Type: Uppercase Letter

Uppercase: 'H' (ASCII: 72)

Lowercase: 'h' (ASCII: 104)

ASCII difference (upper - lower): -32

Character: 'E' | ASCII: 69

Type: Uppercase Letter

Uppercase: 'E' (ASCII: 69)

Lowercase: 'e' (ASCII: 101)

ASCII difference (upper - lower): -32

Character: 'L' | ASCII: 76

Type: Uppercase Letter

Uppercase: 'L' (ASCII: 76)

Lowercase: 'l' (ASCII: 108)

ASCII difference (upper - lower): -32

Character: 'L' | ASCII: 76

Type: Uppercase Letter

Uppercase: 'L' (ASCII: 76)

Lowercase: 'l' (ASCII: 108)

ASCII difference (upper - lower): -32

Character: 'O' | ASCII: 79

Type: Uppercase Letter

Uppercase: 'O' (ASCII: 79)

Lowercase: 'o' (ASCII: 111)

ASCII difference (upper - lower): -32

Character: ' ' | ASCII: 32

Type: Special Character

Character: 'J' | ASCII: 74

Type: Uppercase Letter

Uppercase: 'J' (ASCII: 74)

Lowercase: 'j' (ASCII: 106)

ASCII difference (upper - lower): -32

Character: 'I' | ASCII: 73

Type: Uppercase Letter

Uppercase: 'I' (ASCII: 73)

Lowercase: 'i' (ASCII: 105)

ASCII difference (upper - lower): -32

Character: ' ' | ASCII: 32

Type: Special Character

Character: 'K' | ASCII: 75

Type: Uppercase Letter

Uppercase: 'K' (ASCII: 75)

Lowercase: 'k' (ASCII: 107)

ASCII difference (upper - lower): -32

Character: 'A' | ASCII: 65

Type: Uppercase Letter

Uppercase: 'A' (ASCII: 65)

Lowercase: 'a' (ASCII: 97)

ASCII difference (upper - lower): -32

Character: 'I' | ASCII: 73

Type: Uppercase Letter

Uppercase: 'I' (ASCII: 73)

Lowercase: 'i' (ASCII: 105)

ASCII difference (upper - lower): -32

Character: 'S' | ASCII: 83

Type: Uppercase Letter

Uppercase: 'S' (ASCII: 83)

Lowercase: 's' (ASCII: 115)

ASCII difference (upper - lower): -32

Character: 'A' | ASCII: 65

Type: Uppercase Letter

Uppercase: 'A' (ASCII: 65)

Lowercase: 'a' (ASCII: 97)

ASCII difference (upper - lower): -32

Character: 'E' | ASCII: 69

Type: Uppercase Letter

Uppercase: 'E' (ASCII: 69)

Lowercase: 'e' (ASCII: 101)

ASCII difference (upper - lower): -32

Character: ' ' | ASCII: 32

Type: Special Character

Character: 'H' | ASCII: 72

Type: Uppercase Letter

Uppercase: 'H' (ASCII: 72)

Lowercase: 'h' (ASCII: 104)

ASCII difference (upper - lower): -32

Character: 'O' | ASCII: 79

Type: Uppercase Letter

Uppercase: 'O' (ASCII: 79)

Lowercase: 'o' (ASCII: 111)

ASCII difference (upper - lower): -32

import java.util.Scanner;

public class AdvancedStringAnalyzer {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("=== ADVANCED STRING ANALYZER ===");

// Ask user for two strings to compare

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

String str1 = scanner.nextLine();

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

String str2 = scanner.nextLine();

// Perform comprehensive comparison analysis

performAllComparisons(str1, str2);

// Performance analysis of different string operations

analyzeMemoryUsage(str1, str2);

String[] arr = {str1, str2, "Extra", "Strings", "For", "Testing"};

String result = optimizedStringProcessing(arr);

System.out.println("Optimized concatenation: " + result);

// Demonstrate intern() method

demonstrateStringIntern();

scanner.close();

}

// Method to calculate string similarity percentage using Levenshtein distance

public static double calculateSimilarity(String str1, String str2) {

int len1 = str1.length();

int len2 = str2.length();

int[][] dp = new int[len1 + 1][len2 + 1];

for (int i = 0; i <= len1; i++) dp[i][0] = i;

for (int j = 0; j <= len2; j++) dp[0][j] = j;

for (int i = 1; i <= len1; i++) {

for (int j = 1; j <= len2; j++) {

if (str1.charAt(i - 1) == str2.charAt(j - 1))

dp[i][j] = dp[i - 1][j - 1];

else

dp[i][j] = 1 + Math.min(dp[i - 1][j - 1], Math.min(dp[i - 1][j], dp[i][j - 1]));

}

}

int maxLen = Math.max(len1, len2);

if (maxLen == 0) return 100.0;

int distance = dp[len1][len2];

return 100.0 \* (maxLen - distance) / maxLen;

}

// Method to perform all comparison types

public static void performAllComparisons(String str1, String str2) {

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

// 1. Reference equality (==)

System.out.println("Reference equality (==): " + (str1 == str2));

// 2. Content equality (equals)

System.out.println("Content equality (equals): " + str1.equals(str2));

// 3. Case-insensitive equality (equalsIgnoreCase)

System.out.println("Case-insensitive equality: " + str1.equalsIgnoreCase(str2));

// 4. Lexicographic comparison (compareTo)

System.out.println("Lexicographic compareTo: " + str1.compareTo(str2));

// 5. Case-insensitive lexicographic comparison

System.out.println("Case-insensitive compareTo: " + str1.compareToIgnoreCase(str2));

// 6. Similarity percentage calculation

double similarity = calculateSimilarity(str1, str2);

System.out.printf("Similarity percentage: %.2f%%\n", similarity);

}

// Method to analyze string memory usage (approximate)

public static void analyzeMemoryUsage(String... strings) {

System.out.println("\n--- Memory Usage Analysis ---");

for (String s : strings) {

// Approximate: 40 bytes object overhead + 2 bytes per char

int mem = 40 + s.length() \* 2;

System.out.println("String: \"" + s + "\" | Length: " + s.length() + " | Approx. memory: " + mem + " bytes");

}

}

// Method to optimize string operations using StringBuilder

public static String optimizedStringProcessing(String[] inputs) {

StringBuilder sb = new StringBuilder();

for (String s : inputs) {

sb.append(s).append(" ");

}

return sb.toString().trim();

}

// Method to demonstrate intern() method

public static void demonstrateStringIntern() {

System.out.println("\n--- String Intern Demonstration ---");

String a = "hello";

String b = new String("hello");

System.out.println("a == b: " + (a == b));

String c = b.intern();

System.out.println("a == c (after intern): " + (a == c));

}

}

