

�� **PRACTICE PROBLEM 1: (Any 4)**

**Built-In String Methods - Basic Operations**

**Task:** 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! ";

// 1. Display original string length including spaces

System.out.println("1. Original String: \"" + sampleText + "\"");

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

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

String trimmedText = sampleText.trim();

System.out.println("\n2. Trimmed String: \"" + trimmedText + "\"");

System.out.println(" New Length (without leading/trailing spaces): " + trimmedText.length());

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

System.out.println("\n3. Character at Index 5: " + sampleText.charAt(5));

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

String subStr = trimmedText.substring(5, 16);

System.out.println("\n4. Extracted Substring: " + subStr);

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

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

System.out.println("\n5. Index of \"Fun\": " + funIndex);

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

System.out.println("\n6. Contains \"Java\"? " + trimmedText.contains("Java"));

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

System.out.println("\n7. Starts with \"Java\"? " + trimmedText.startsWith("Java"));

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

System.out.println("\n8. Ends with '!'? " + trimmedText.endsWith("!"));

// 9. Convert the entire string to uppercase

System.out.println("\n9. Uppercase: " + trimmedText.toUpperCase());

// 10. Convert the entire string to lowercase

System.out.println("\n10. Lowercase: " + trimmedText.toLowerCase());

// Count vowels in the string

int vowelCount = countVowels(trimmedText);

System.out.println("\n11. Number of vowels: " + vowelCount);

// Find all occurrences of a character

System.out.println("\n12. Positions of character 'a':");

findAllOccurrences(trimmedText, 'a');

}

// Method to count vowels using charAt()

public static int countVowels(String text) {

int count = 0;

text = text.toLowerCase();

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

char ch = text.charAt(i);

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

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.println(" Found '" + target + "' at index: " + i);

found = true;

}

}

if (!found) {

System.out.println(" Character '" + target + "' not found in the string.");

}

}

}

OUTPUT:

1. Original String: " Java Programming is Fun and Challenging! "

Original Length (with spaces): 42

2. Trimmed String: "Java Programming is Fun and Challenging!"

New Length (without leading/trailing spaces): 40

3. Character at Index 5:

4. Extracted Substring: Programming

5. Index of "Fun": 20

6. Contains "Java"? true

7. Starts with "Java"? true

8. Ends with '!'? true

9. Uppercase: JAVA PROGRAMMING IS FUN AND CHALLENGING!

10. Lowercase: java programming is fun and challenging!

11. Number of vowels: 11

12. Positions of character 'a':

Found 'a' at index: 1

Found 'a' at index: 3

Found 'a' at index: 10

Found 'a' at index: 24

Found 'a' at index: 30

1



�� **PRACTICE PROBLEM 2:**

**String Manipulation Methods**

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

PROGRAM:

import java.util.\*;

public class StringManipulation {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String input = sc.nextLine().trim();

// 1. Trim

System.out.println("\nTrimmed: " + input);

// 2. Replace spaces with underscores

System.out.println("Spaces replaced: " + input.replace(" ", "\_"));

// 3. Remove digits

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

System.out.println("No digits: " + noDigits);

// 4. Split words

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

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

// 5. Join words

System.out.println("Joined: " + String.join(" | ", words));

// 6. Remove punctuation

String noPunct = noDigits.replaceAll("\\p{Punct}", "");

System.out.println("No punctuation: " + noPunct);

// 7. Capitalize words

System.out.println("Capitalized: " + capitalizeWords(noPunct));

// 8. Reverse words

System.out.println("Reversed: " + reverseWords(noPunct));

// 9. Word frequency

System.out.println("\nWord Frequency:");

countWordFrequency(noPunct);

sc.close();

}

// Capitalize first letter of each word

public static String capitalizeWords(String text) {

StringBuilder sb = new StringBuilder();

for (String w : text.split("\\s+")) {

sb.append(Character.toUpperCase(w.charAt(0)))

.append(w.substring(1).toLowerCase())

.append(" ");

}

return sb.toString().trim();

}

// Reverse order of words

public static String reverseWords(String text) {

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

StringBuilder sb = new StringBuilder();

for (int i = words.length - 1; i >= 0; i--) sb.append(words[i]).append(" ");

return sb.toString().trim();

}

// Count word frequency

public static void countWordFrequency(String text) {

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

boolean[] visited = new boolean[words.length];

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

if (visited[i]) continue;

int count = 1;

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

if (words[i].equals(words[j])) {

count++;

visited[j] = true;

}

}

System.out.println(" " + words[i] + " -> " + count);

}

}

}

OUTPUT:

Enter a sentence: Hello This is a java program

Trimmed: Hello This is a java program

Spaces replaced: Hello\_This\_is\_a\_java\_program

No digits: Hello This is a java program

Words: [Hello, This, is, a, java, program]

Joined: Hello | This | is | a | java | program

No punctuation: Hello This is a java program

Capitalized: Hello This Is A Java Program

Reversed: program java a is This Hello

Word Frequency:

hello ? 1

this ? 1

is ? 1

a ? 1

java ? 1

program ? 1

�� **PRACTICE PROBLEM 3:**

**ASCII Codes and Character Conversion**

**Task:** Create a program that demonstrates ASCII character manipulation and conversion.

PROGRAM:

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();

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

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

char ch = input.charAt(i);

int ascii = (int) ch;

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

// Check character type

if (Character.isUpperCase(ch)) {

System.out.print(" | Type: Uppercase");

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

System.out.print(" | Type: Lowercase");

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

System.out.print(" | Type: Digit");

} else {

System.out.print(" | Type: Special Character");

}

// Show upper and lower case versions with ASCII codes (if alphabet)

if (Character.isLetter(ch)) {

char upper = Character.toUpperCase(ch);

char lower = Character.toLowerCase(ch);

System.out.print(" | Upper: '" + upper + "'(" + (int) upper + ")");

System.out.print(" | Lower: '" + lower + "'(" + (int) lower + ")");

System.out.print(" | Diff: " + Math.abs((int) upper - (int) lower));

}

System.out.println();

}

// ASCII art example

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

for (int i = 65; i <= 69; i++) { // Prints A to E in a pattern

for (int j = 65; j <= i; j++) {

System.out.print((char) j + " ");

}

System.out.println();

}

// Simple Caesar Cipher

System.out.print("\nEnter shift value for Caesar Cipher: ");

int shift = scanner.nextInt();

String encrypted = caesarCipher(input, shift);

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

System.out.println("Decrypted: " + caesarCipher(encrypted, -shift));

scanner.close();

}

// Caesar Cipher Method

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';

ch = (char) ((ch - base + shift + 26) % 26 + base);

}

result.append(ch);

}

return result.toString();

}

}

OUTPUT:

Enter a string: Java!123

--- ASCII Character Analysis ---

Character: 'J' | ASCII: 74 | Type: Uppercase | Upper: 'J'(74) | Lower: 'j'(106) | Diff: 32

Character: 'a' | ASCII: 97 | Type: Lowercase | Upper: 'A'(65) | Lower: 'a'(97) | Diff: 32

Character: 'v' | ASCII: 118 | Type: Lowercase | Upper: 'V'(86) | Lower: 'v'(118) | Diff: 32

Character: 'a' | ASCII: 97 | Type: Lowercase | Upper: 'A'(65) | Lower: 'a'(97) | Diff: 32

Character: '!' | ASCII: 33 | Type: Special Character

Character: '1' | ASCII: 49 | Type: Digit

Character: '2' | ASCII: 50 | Type: Digit

Character: '3' | ASCII: 51 | Type: Digit

--- ASCII Art ---

A

A B

A B C

A B C D

A B C D E

Enter shift value for Caesar Cipher: program@456

Exception in thread "main" java.util.InputMismatchException

at java.base/java.util.Scanner.throwFor(Scanner.java:964)

at java.base/java.util.Scanner.next(Scanner.java:1619)

at java.base/java.util.Scanner.nextInt(Scanner.java:2284)

at java.base/java.util.Scanner.nextInt(Scanner.java:2238)

at ASCIIProcessor.main(ASCIIProcessor.java:51)

�� **PRACTICE PROBLEM 4:**

**StringBuilder, StringBuffer, and Performance**

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

PROGRAM:

public class StringPerformanceComparison {

public static void main(String[] args) {

int iterations = 10000;

System.out.println("=== STRING PERFORMANCE COMPARISON ===");

// Test 1: String (slow)

long start = System.nanoTime();

String str = "";

for (int i = 0; i < iterations; i++) {

str += "Java" + i;

}

long end = System.nanoTime();

System.out.println("String time : " + (end - start) + " ns");

// Test 2: StringBuilder (fast)

start = System.nanoTime();

StringBuilder sb = new StringBuilder();

for (int i = 0; i < iterations; i++) {

sb.append("Java").append(i);

}

end = System.nanoTime();

System.out.println("StringBuilder time: " + (end - start) + " ns");

// Test 3: StringBuffer (thread-safe, slower than StringBuilder)

start = System.nanoTime();

StringBuffer sbf = new StringBuffer();

for (int i = 0; i < iterations; i++) {

sbf.append("Java").append(i);

}

end = System.nanoTime();

System.out.println("StringBuffer time : " + (end - start) + " ns");

System.out.println("\n=== CONCLUSION ===");

System.out.println("String → Slow, avoid for many changes.");

System.out.println("StringBuilder → Fastest, best for single-threaded.");

System.out.println("StringBuffer → Thread-safe but a bit slower.");

}

}

OUTPUT:

=== STRING PERFORMANCE COMPARISON ===

String time : 223446300 ns

StringBuilder time: 2411000 ns

StringBuffer time : 2825300 ns

=== CONCLUSION ===

String ? Slow, avoid for many changes.

StringBuilder ? Fastest, best for single-threaded.

StringBuffer ? Thread-safe but a bit slower.