**BY: Aishwarya TJ**

1. **Code**

public class StringBuiltInMethods {

public static void main(String[] args) {

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

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

String trimmedText = sampleText.trim();

System.out.println("2. Trimmed text length: " + trimmedText.length());

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

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

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

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

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

System.out.println("6. Contains \"Java\": " + trimmedText.contains("Java"));

System.out.println("7. Starts with \"Java\": " + trimmedText.startsWith("Java"));

System.out.println("8. Ends with '!': " + trimmedText.endsWith("!"));

System.out.println("9. Uppercase: " + trimmedText.toUpperCase());

System.out.println("10. Lowercase: " + trimmedText.toLowerCase());

int vowelCount = countVowels(trimmedText);

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

System.out.print("12. Occurrences of 'a': ");

findAllOccurrences(trimmedText, 'a');

}

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;

}

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("Character not found");

}

System.out.println();

}

}

1. **Code**

import java.util.\*;

public class StringManipulation {

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("1. Trimmed: " + trimmed);

String replaced = trimmed.replace(" ", "\_");

System.out.println("2. Spaces replaced with underscores: " + replaced);

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

System.out.println("3. Digits removed: " + noDigits);

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

System.out.println("4. Words array: " + Arrays.toString(words));

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

System.out.println("5. Rejoined with | : " + joined);

String noPunct = removePunctuation(trimmed);

System.out.println("6. Without punctuation: " + noPunct);

String capitalized = capitalizeWords(noPunct);

System.out.println("7. Capitalized: " + capitalized);

String reversedOrder = reverseWordOrder(noPunct);

System.out.println("8. Words reversed: " + reversedOrder);

System.out.println("9. Word Frequency:");

countWordFrequency(noPunct);

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 result = new StringBuilder();

for (String word : words) {

if (!word.isEmpty()) {

result.append(Character.toUpperCase(word.charAt(0)))

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

.append(" ");

}

}

return result.toString().trim();

}

public static String reverseWordOrder(String text) {

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

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

return String.join(" ", words);

}

public static void countWordFrequency(String text) {

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

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

for (String word : words) {

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

}

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

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

}

}

}

1. **Code**

import java.util.\*;

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("\nCharacter: " + ch + " | ASCII: " + ascii);

System.out.println("Type: " + classifyCharacter(ch));

if (Character.isLetter(ch)) {

char upper = Character.toUpperCase(ch);

char lower = Character.toLowerCase(ch);

System.out.println("Uppercase: " + upper + " (" + (int) upper + ")");

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

System.out.println("Difference between cases: " + Math.abs((int) upper - (int) lower));

}

}

System.out.println("\nASCII Table (65-90):");

displayASCIITable(65, 90);

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

int shift = scanner.nextInt();

scanner.nextLine();

String encrypted = caesarCipher(input, shift);

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

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

int[] asciiArray = stringToASCII(input);

System.out.println("\nString to ASCII array: " + Arrays.toString(asciiArray));

String backToString = asciiToString(asciiArray);

System.out.println("ASCII array back to String: " + backToString);

scanner.close();

}

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 char toggleCase(char ch) {

if (Character.isUpperCase(ch)) {

return (char) (ch + 32);

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

return (char) (ch - 32);

}

return ch;

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();

}

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

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

System.out.println(i + " -> " + (char) i);

}

}

public static int[] stringToASCII(String text) {

int[] asciiValues = new int[text.length()];

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

asciiValues[i] = (int) text.charAt(i);

}

return asciiValues;

}

public static String asciiToString(int[] asciiValues) {

StringBuilder sb = new StringBuilder();

for (int val : asciiValues) {

sb.append((char) val);

}

return sb.toString();

}

}

1. **Code**

public class StringPerformanceComparison {

public static void main(String[] args) {

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

long startTime = System.nanoTime();

String result1 = concatenateWithString(1000);

long endTime = System.nanoTime();

System.out.println("String concatenation time: " + (endTime - startTime) + " ns");

startTime = System.nanoTime();

String result2 = concatenateWithStringBuilder(1000);

endTime = System.nanoTime();

System.out.println("StringBuilder concatenation time: " + (endTime - startTime) + " ns");

startTime = System.nanoTime();

String result3 = concatenateWithStringBuffer(1000);

endTime = System.nanoTime();

System.out.println("StringBuffer concatenation time: " + (endTime - startTime) + " ns");

System.out.println("\n=== STRINGBUILDER METHODS DEMO ===");

demonstrateStringBuilderMethods();

System.out.println("\n=== THREAD SAFETY DEMO ===");

demonstrateThreadSafety();

}

public static String concatenateWithString(int iterations) {

String result = "";

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

result += "Java " + i + " ";

}

return result;

}

public static String concatenateWithStringBuilder(int iterations) {

StringBuilder sb = new StringBuilder();

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

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

}

return sb.toString();

}

public static String concatenateWithStringBuffer(int iterations) {

StringBuffer sb = new StringBuffer();

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

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

}

return sb.toString();

}

public static void demonstrateStringBuilderMethods() {

StringBuilder sb = new StringBuilder("Hello World");

System.out.println("Original: " + sb);

sb.append(" Java");

System.out.println("append(): " + sb);

sb.insert(6, "Beautiful ");

System.out.println("insert(): " + sb);

sb.delete(6, 16);

System.out.println("delete(): " + sb);

sb.deleteCharAt(5);

System.out.println("deleteCharAt(): " + sb);

sb.reverse();

System.out.println("reverse(): " + sb);

sb.reverse();

sb.replace(6, 11, "Universe");

System.out.println("replace(): " + sb);

sb.setCharAt(0, 'h');

System.out.println("setCharAt(): " + sb);

System.out.println("capacity(): " + sb.capacity());

sb.ensureCapacity(50);

System.out.println("ensureCapacity(50): " + sb.capacity());

sb.trimToSize();

System.out.println("trimToSize(): " + sb.capacity());

}

public static void demonstrateThreadSafety() {

StringBuffer safeBuffer = new StringBuffer("Start-");

StringBuilder unsafeBuilder = new StringBuilder("Start-");

Runnable bufferTask = () -> {

for (int i = 0; i < 100; i++) {

safeBuffer.append("B");

}

};

Runnable builderTask = () -> {

for (int i = 0; i < 100; i++) {

unsafeBuilder.append("U");

}

};

Thread t1 = new Thread(bufferTask);

Thread t2 = new Thread(bufferTask);

Thread t3 = new Thread(builderTask);

Thread t4 = new Thread(builderTask);

t1.start(); t2.start(); t3.start(); t4.start();

try {

t1.join(); t2.join(); t3.join(); t4.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("StringBuffer length: " + safeBuffer.length());

System.out.println("StringBuilder length: " + unsafeBuilder.length());

}

}