**AISHWARYA**

**RA2411033010131**

**AN2**

1. Code

import java.util.Scanner;

public class SubstringReplace {

public static int[] findSubstringOccurrences(String text, String substring) {

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

int count = 0;

int index = text.indexOf(substring);

while (index != -1) {

positions[count] = index;

count++;

index = text.indexOf(substring, index + 1); // Move to the next occurrence

}

int[] result = new int[count];

System.arraycopy(positions, 0, result, 0, count);

return result;

}

public static String replaceSubstring(String text, String substring, String replacement) {

int[] positions = findSubstringOccurrences(text, substring);

StringBuilder newText = new StringBuilder();

int lastIndex = 0;

for (int position : positions) {

newText.append(text, lastIndex, position);

newText.append(replacement);

lastIndex = position + substring.length();

}

newText.append(text.substring(lastIndex));

return newText.toString();

}

public static boolean compareWithBuiltInReplace(String text, String substring, String replacement) {

String manualReplacement = replaceSubstring(text, substring, replacement);

String builtInReplacement = text.replace(substring, replacement);

return manualReplacement.equals(builtInReplacement);

}

public static void main(String[] args) {

Scanner scanner = new Scanner([System.in](http://system.in));

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

String text = scanner.nextLine();

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

String substring = scanner.nextLine();

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

String replacement = scanner.nextLine();

String manualResult = replaceSubstring(text, substring, replacement);

boolean comparisonResult = compareWithBuiltInReplace(text, substring, replacement);

System.out.println("\nOriginal text: " + text);

System.out.println("Text after manual replacement: " + manualResult);

System.out.println("Comparison with built-in replace(): " + (comparisonResult ? "Match" : "No Match"));

scanner.close();

}

}

1. Code

import java.util.Scanner;

public class CaseConversion {

public static char toUpperCase(char ch) {

if (ch >= 'a' && ch <= 'z') {

ch = (char) (ch - 32);

}

return ch;

}

public static char toLowerCase(char ch) {

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

ch = (char) (ch + 32);

}

return ch;

}

public static String toTitleCase(String text) {

StringBuilder titleCase = new StringBuilder();

boolean newWord = true;

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

if (Character.isWhitespace(ch)) {

titleCase.append(ch);

newWord = true;

} else if (newWord) {

titleCase.append(toUpperCase(ch));

newWord = false;

} else {

titleCase.append(toLowerCase(ch));

}

}

return titleCase.toString();

}

public static boolean compareWithBuiltInMethods(String text) {

String manualUpper = convertToUpperCase(text);

String manualLower = convertToLowerCase(text);

String manualTitle = toTitleCase(text);

boolean upperMatch = manualUpper.equals(text.toUpperCase());

boolean lowerMatch = manualLower.equals(text.toLowerCase());

boolean titleMatch = manualTitle.equals(toTitleCase(text));

return upperMatch && lowerMatch && titleMatch;

}

public static String convertToUpperCase(String text) {

StringBuilder upperText = new StringBuilder();

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

upperText.append(toUpperCase(ch));

}

return upperText.toString();

}

public static String convertToLowerCase(String text) {

StringBuilder lowerText = new StringBuilder();

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

lowerText.append(toLowerCase(ch));

}

return lowerText.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner([System.in](http://system.in));

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

String text = scanner.nextLine();

String upperCaseText = convertToUpperCase(text);

String lowerCaseText = convertToLowerCase(text);

String titleCaseText = toTitleCase(text);

boolean isMatch = compareWithBuiltInMethods(text);

System.out.println("\nResults:");

System.out.printf("%-15s %-15s %-15s %-15s\n", "Original", "Uppercase", "Lowercase", "Title Case");

System.out.printf("%-15s %-15s %-15s %-15s\n", text, upperCaseText, lowerCaseText, titleCaseText);

System.out.println("\nComparison with built-in methods (toUpperCase, toLowerCase, toTitleCase): ");

System.out.println(isMatch ? "Manual conversion matches built-in methods!" : "Manual conversion does not match built-in methods.");

scanner.close();

}

}

1. Code

public class StringConcatenationPerformance {

public static String concatenationMethod(int iterations) {

long startTime = System.currentTimeMillis();

String str = "";

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

str += "test";

}

long endTime = System.currentTimeMillis();

long timeTaken = endTime - startTime;

return "Time taken: " + timeTaken + "ms, Final string length: " + str.length();

}

public static String stringBuilderMethod(int iterations) {

long startTime = System.currentTimeMillis();

StringBuilder sb = new StringBuilder();

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

sb.append("test");

}

long endTime = System.currentTimeMillis();

long timeTaken = endTime - startTime;

return "Time taken: " + timeTaken + "ms, Final string length: " + sb.length();

}

public static String stringBufferMethod(int iterations) {

long startTime = System.currentTimeMillis();

StringBuffer sbf = new StringBuffer();

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

sbf.append("test");

}

long endTime = System.currentTimeMillis();

long timeTaken = endTime - startTime;

return "Time taken: " + timeTaken + "ms, Final string length: " + sbf.length();

}

public static void displayComparison(int iterations) {

System.out.println("Performance Comparison for " + iterations + " iterations:");

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

System.out.printf("%-15s %-20s %-20s %-20s\n", "Method", "Time Taken (ms)", "Memory Efficiency", "Final Length");

String concatResult = concatenationMethod(iterations);

String builderResult = stringBuilderMethod(iterations);

String bufferResult = stringBufferMethod(iterations);

System.out.printf("%-15s %-20s %-20s %-20s\n", "String Concatenation", concatResult.split(",")[0], "Less Efficient", concatResult.split(",")[1]);

System.out.printf("%-15s %-20s %-20s %-20s\n", "StringBuilder", builderResult.split(",")[0], "More Efficient", builderResult.split(",")[1]);

System.out.printf("%-15s %-20s %-20s %-20s\n", "StringBuffer", bufferResult.split(",")[0], "More Efficient", bufferResult.split(",")[1]);

}

public static void main(String[] args) {

java.util.Scanner scanner = new java.util.Scanner(System.in);

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

int iterations = scanner.nextInt();

displayComparison(iterations);

scanner.close();

}

}

1. Code

import java.util.Scanner;

public class CaesarCipher {

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

StringBuilder encryptedText = new StringBuilder();

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

int asciiValue = (int) ch;

char encryptedChar;

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

encryptedChar = (char) ((asciiValue - 'A' + shift) % 26 + 'A');

if (encryptedChar < 'A') encryptedChar += 26;

}

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

encryptedChar = (char) ((asciiValue - 'a' + shift) % 26 + 'a');

if (encryptedChar < 'a') encryptedChar += 26;

}

else {

encryptedChar = ch;

}

encryptedText.append(encryptedChar);

}

return encryptedText.toString();

}

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

StringBuilder decryptedText = new StringBuilder();

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

int asciiValue = (int) ch;

char decryptedChar;

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

decryptedChar = (char) ((asciiValue - 'A' - shift + 26) % 26 + 'A');

}

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

decryptedChar = (char) ((asciiValue - 'a' - shift + 26) % 26 + 'a');

}

else {

decryptedChar = ch;

}

decryptedText.append(decryptedChar);

}

return decryptedText.toString();

}

public static void displayASCIIValues(String text) {

System.out.println("\nCharacter to ASCII values:");

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

System.out.printf("'%c' -> %d\n", ch, (int) ch);

}

}

public static boolean validateDecryption(String originalText, String decryptedText) {

return originalText.equals(decryptedText);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String text = scanner.nextLine();

System.out.print("Enter the shift value (positive or negative): ");

int shift = scanner.nextInt();

System.out.println("\nOriginal Text and ASCII values:");

displayASCIIValues(text);

String encryptedText = encrypt(text, shift);

System.out.println("\nEncrypted Text and ASCII values:");

displayASCIIValues(encryptedText);

String decryptedText = decrypt(encryptedText, shift);

System.out.println("\nDecrypted Text and ASCII values:");

displayASCIIValues(decryptedText);

boolean isValid = validateDecryption(text, decryptedText);

System.out.println("\nDecryption valid: " + (isValid ? "Yes" : "No"));

scanner.close();

}

}

1. Code

import java.util.\*;

public class EmailAnalyzer {

public static boolean validateEmail(String email) {

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

if (atPos == -1 || atPos != email.lastIndexOf('@')) {

return false;

}

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

if (dotPos == -1) {

return false;

}

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

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

return !username.isEmpty() && !domain.isEmpty();

}

public static String[] extractEmailComponents(String email) {

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

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

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

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

String domainName = domain.substring(0, dotPos);

String extension = domain.substring(dotPos + 1);

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

}

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

int validCount = 0;

int invalidCount = 0;

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

int totalUsernameLength = 0;

for (String email : emails) {

boolean isValid = validateEmail(email);

if (isValid) {

validCount++;

String[] components = extractEmailComponents(email);

String domain = components[1];

domainCount.put(domain, domainCount.getOrDefault(domain, 0) + 1);

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

} else {

invalidCount++;

}

}

String mostCommonDomain = Collections.max(domainCount.entrySet(), Map.Entry.comparingByValue()).getKey();

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

System.out.println("\nEmail Analysis Report:");

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

System.out.printf("%-30s %-20s %-20s %-15s %-15s %-10s\n", "Email", "Username", "Domain", "Domain Name", "Extension", "Status");

for (String email : emails) {

String[] components = extractEmailComponents(email);

boolean isValid = validateEmail(email);

String status = isValid ? "Valid" : "Invalid";

System.out.printf("%-30s %-20s %-20s %-15s %-15s %-10s\n", email, components[0], components[1], components[2], components[3], status);

}

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

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

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

System.out.println("Average Username Length: " + avgUsernameLength);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

System.out.println("Enter email addresses (type 'done' to stop):");

while (true) {

String email = scanner.nextLine().trim();

if (email.equalsIgnoreCase("done")) {

break;

}

emails.add(email);

}

analyzeEmailStatistics(emails);

scanner.close();

}

}

1. Code

import java.util.\*;

public class TextFormatter {

public static String[] splitText(String text) {

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

int start = 0;

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

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

if (i > start) {

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

}

start = i + 1;

}

}

if (start < text.length()) {

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

}

return words.toArray(new String[0]);

}

public static String justifyText(String[] words, int width) {

StringBuilder result = new StringBuilder();

StringBuilder currentLine = new StringBuilder();

int currentLength = 0;

for (String word : words) {

if (currentLength + currentLine.length() + word.length() <= width) {

if (currentLine.length() > 0) {

currentLine.append(" ");

currentLength++;

}

currentLine.append(word);

currentLength += word.length();

} else {

result.append(justifyLine(currentLine.toString(), width));

result.append("\n");

currentLine.setLength(0);

currentLine.append(word);

currentLength = word.length();

}

}

if (currentLine.length() > 0) {

result.append(currentLine.toString());

result.append("\n");

}

return result.toString();

}

public static String justifyLine(String line, int width) {

String[] words = line.split(" ");

if (words.length == 1) {

return words[0];

}

int totalSpaces = width - line.length() + (words.length - 1);

int spacesBetweenWords = totalSpaces / (words.length - 1);

int extraSpaces = totalSpaces % (words.length - 1);

StringBuilder justifiedLine = new StringBuilder();

for (int i = 0; i < words.length - 1; i++) {

justifiedLine.append(words[i]);

for (int j = 0; j < spacesBetweenWords; j++) {

justifiedLine.append(" ");

}

if (i < extraSpaces) {

justifiedLine.append(" ");

}

}

justifiedLine.append(words[words.length - 1]);

return justifiedLine.toString();

}

public static String centerAlignText(String[] words, int width) {

StringBuilder result = new StringBuilder();

StringBuilder currentLine = new StringBuilder();

int currentLength = 0;

for (String word : words) {

if (currentLength + currentLine.length() + word.length() <= width) {

if (currentLine.length() > 0) {

currentLine.append(" ");

currentLength++;

}

currentLine.append(word);

currentLength += word.length();

} else {

result.append(centerAlignLine(currentLine.toString(), width));

result.append("\n");

currentLine.setLength(0);

currentLine.append(word);

currentLength = word.length();

}

}

if (currentLine.length() > 0) {

result.append(centerAlignLine(currentLine.toString(), width));

result.append("\n");

}

return result.toString();

}

public static String centerAlignLine(String line, int width) {

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

int spacesAfter = width - line.length() - spacesBefore;

StringBuilder centeredLine = new StringBuilder();

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

centeredLine.append(" ");

}

centeredLine.append(line);

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

centeredLine.append(" ");

}

return centeredLine.toString();

}

public static void comparePerformance(String text, int width) {

String[] words = splitText(text);

long startTime = System.nanoTime();

justifyText(words, width);

long endTime = System.nanoTime();

long sbTime = endTime - startTime;

startTime = System.nanoTime();

justifyTextWithConcatenation(words, width);

endTime = System.nanoTime();

long concatTime = endTime - startTime;

System.out.println("\nPerformance Comparison:");

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

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

}

public static String justifyTextWithConcatenation(String[] words, int width) {

String result = "";

StringBuilder currentLine = new StringBuilder();

int currentLength = 0;

for (String word : words) {

if (currentLength + currentLine.length() + word.length() <= width) {

if (currentLine.length() > 0) {

currentLine.append(" ");

currentLength++;

}

currentLine.append(word);

currentLength += word.length();

} else {

result += justifyLine(currentLine.toString(), width) + "\n";

currentLine.setLength(0);

currentLine.append(word);

currentLength = word.length();

}

}

if (currentLine.length() > 0) {

result += currentLine.toString() + "\n";

}

return result;

}

public static void displayFormattedText(String text, int width, String alignment) {

String[] words = splitText(text);

String formattedText = "";

if (alignment.equals("justify")) {

formattedText = justifyText(words, width);

} else if (alignment.equals("center")) {

formattedText = centerAlignText(words, width);

}

String[] lines = formattedText.split("\n");

System.out.println("\nFormatted Text (" + alignment + "):");

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

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

System.out.printf("Line %d: %-20s (Length: %d)\n", i + 1, lines[i], lines[i].length());

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String text = scanner.nextLine();

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

int width = scanner.nextInt();

System.out.println("\nOriginal Text:");

System.out.println(text);

displayFormattedText(text, width, "justify");

displayFormattedText(text, width, "center");

comparePerformance(text, width);

scanner.close();

}

}