STEP LAB PROBLEMS

HRISHI

RA2411028010092

SEC: V1

1-import java.util.ArrayList;

import java.util.Scanner;

public class exp {

public static ArrayList<Integer> findOccurrences(String text, String toFind) {

ArrayList<Integer> positions = new ArrayList<>();

int index = text.indexOf(toFind);

while (index >= 0) {

positions.add(index);

index = text.indexOf(toFind, index + 1);

}

return positions;

}

public static String manualReplace(String text, String toFind, String toReplace) {

StringBuilder result = new StringBuilder();

int i = 0;

while (i < text.length()) {

if (i <= text.length() - toFind.length() &&

text.substring(i, i + toFind.length()).equals(toFind)) {

result.append(toReplace);

i += toFind.length();

} else {

result.append(text.charAt(i));

i++;

}

}

return result.toString();

}

public static boolean compareWithBuiltIn(String original, String toFind, String toReplace, String customResult) {

String builtInResult = original.replace(toFind, toReplace);

return builtInResult.equals(customResult);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String text = scanner.nextLine();

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

String toFind = scanner.nextLine();

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

String toReplace = scanner.nextLine();

ArrayList<Integer> positions = findOccurrences(text, toFind);

System.out.println("\nOccurrences at positions: " + positions);

String customReplaced = manualReplace(text, toFind, toReplace);

System.out.println("Manually Replaced Text: " + customReplaced);

boolean isEqual = compareWithBuiltIn(text, toFind, toReplace, customReplaced);

System.out.println("Built-in Replaced Text: " + text.replace(toFind, toReplace));

System.out.println("Is Manual Result Same as Built-in? " + isEqual);

scanner.close();

}

}



2-

import java.util.Scanner;

public class exp2 {

public static String toUpperCaseASCII(String input) {

StringBuilder result = new StringBuilder();

for (char c : input.toCharArray()) {

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

result.append((char)(c - 32));

} else {

result.append(c);

}

}

return result.toString();

}

public static String toLowerCaseASCII(String input) {

StringBuilder result = new StringBuilder();

for (char c : input.toCharArray()) {

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

result.append((char)(c + 32));

} else {

result.append(c);

}

}

return result.toString();

}

public static String toTitleCaseASCII(String input) {

StringBuilder result = new StringBuilder();

boolean newWord = true;

for (char c : input.toCharArray()) {

if (c == ' ') {

result.append(c);

newWord = true;

} else if (newWord && c >= 'a' && c <= 'z') {

result.append((char)(c - 32));

newWord = false;

} else if (!newWord && c >= 'A' && c <= 'Z') {

result.append((char)(c + 32));

} else {

result.append(c);

newWord = false;

}

}

return result.toString();

}

public static void compareWithBuiltIn(String input) {

System.out.printf("%-15s | %-25s | %-25s\n", "Method", "ASCII Result", "Built-in Result");

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

String upperAscii = toUpperCaseASCII(input);

System.out.printf("%-15s | %-25s | %-25s\n", "UPPERCASE", upperAscii, input.toUpperCase());

String lowerAscii = toLowerCaseASCII(input);

System.out.printf("%-15s | %-25s | %-25s\n", "lowercase", lowerAscii, input.toLowerCase());

String titleAscii = toTitleCaseASCII(input);

String[] words = input.toLowerCase().split(" ");

StringBuilder titleBuiltIn = new StringBuilder();

for (String word : words) {

if (!word.isEmpty()) {

titleBuiltIn.append(Character.toUpperCase(word.charAt(0))).append(word.substring(1)).append(" ");

}

}

System.out.printf("%-15s | %-25s | %-25s\n", "Title Case", titleAscii, titleBuiltIn.toString().trim());

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String input = scanner.nextLine();

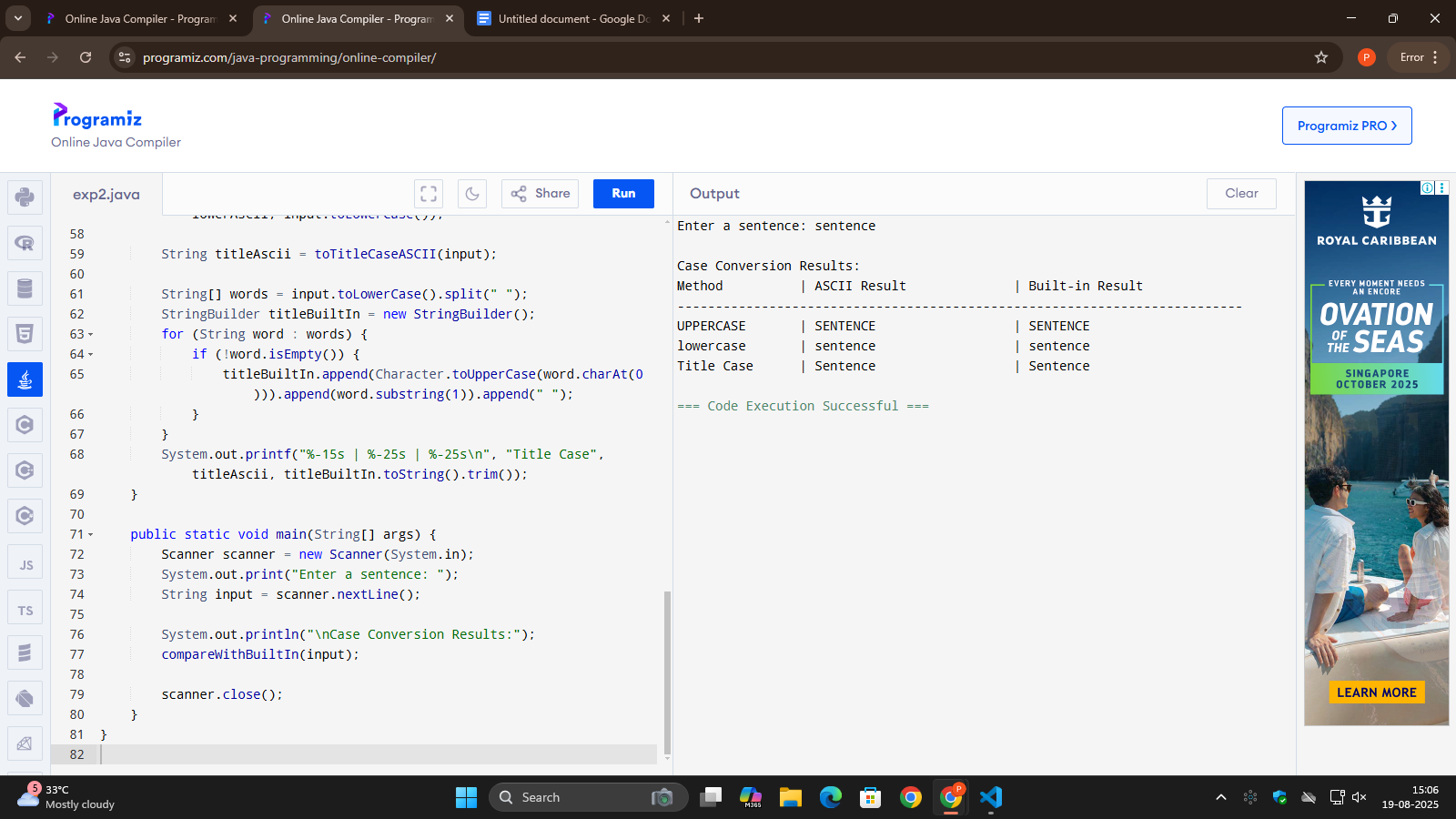
System.out.println("\nCase Conversion Results:");

compareWithBuiltIn(input);

scanner.close();

}

}



3-

import java.util.Scanner;

public class Problem3\_StringPerformance {

// Method using String (+) concatenation

public static long testStringConcat(int iterations) {

long start = System.currentTimeMillis();

String result = "";

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

result += "a";

}

long end = System.currentTimeMillis();

System.out.println("Final String Length (String): " + result.length());

return end - start;

}

// Method using StringBuilder

public static long testStringBuilder(int iterations) {

long start = System.currentTimeMillis();

StringBuilder sb = new StringBuilder();

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

sb.append("a");

}

long end = System.currentTimeMillis();

System.out.println("Final String Length (StringBuilder): " + sb.length());

return end - start;

}

// Method using StringBuffer

public static long testStringBuffer(int iterations) {

long start = System.currentTimeMillis();

StringBuffer sb = new StringBuffer();

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

sb.append("a");

}

long end = System.currentTimeMillis();

System.out.println("Final String Length (StringBuffer): " + sb.length());

return end - start;

}

// Method to display performance comparison

public static void displayResults(int iterations, long stringTime, long builderTime, long bufferTime) {

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

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

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

System.out.printf("%-15s | %-20d | %-20s\n", "String", stringTime, "Slowest");

System.out.printf("%-15s | %-20d | %-20s\n", "StringBuilder", builderTime, "Fastest");

System.out.printf("%-15s | %-20d | %-20s\n", "StringBuffer", bufferTime, "Thread-Safe");

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of iterations (e.g. 1000, 10000): ");

int iterations = scanner.nextInt();

long time1 = testStringConcat(iterations);

long time2 = testStringBuilder(iterations);

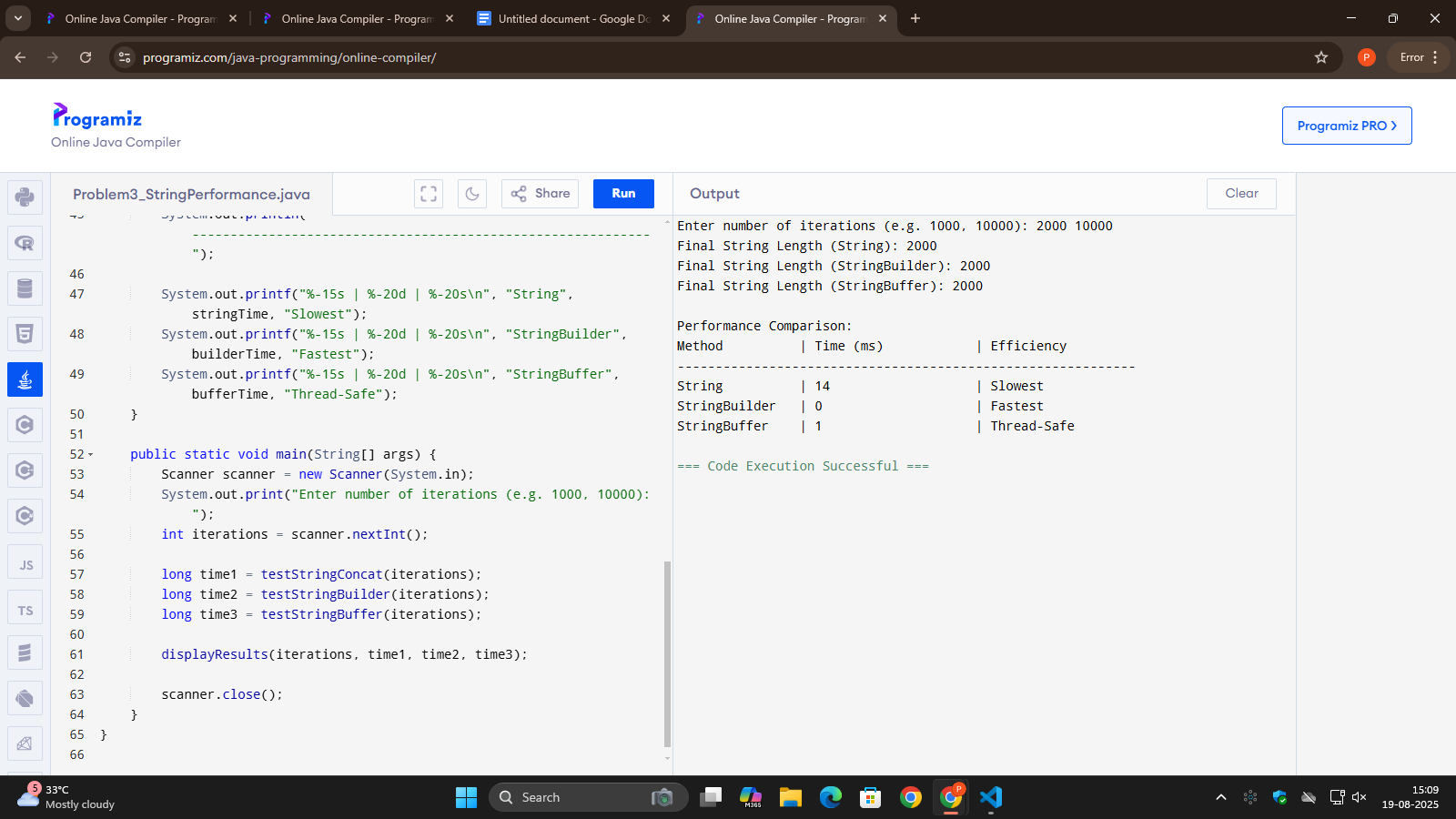
long time3 = testStringBuffer(iterations);

displayResults(iterations, time1, time2, time3);

scanner.close();

}

}



4-

import java.util.Scanner;

public class CaesarCipher {

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

StringBuilder result = new StringBuilder();

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

if (Character.isUpperCase(c)) {

char ch = (char) (((c - 'A' + shift) % 26 + 26) % 26 + 'A');

result.append(ch);

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

char ch = (char) (((c - 'a' + shift) % 26 + 26) % 26 + 'a');

result.append(ch);

} else {

result.append(c);

}

}

return result.toString();

}

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

return encrypt(text, -shift);

}

public static void displayAscii(String text) {

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

System.out.print(c + "(" + (int)c + ") ");

}

System.out.println();

}

public static boolean validate(String original, String decrypted) {

return original.equals(decrypted);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String text = sc.nextLine();

System.out.print("Enter shift value: ");

int shift = sc.nextInt();

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

displayAscii(text);

String encrypted = encrypt(text, shift);

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

displayAscii(encrypted);

String decrypted = decrypt(encrypted, shift);

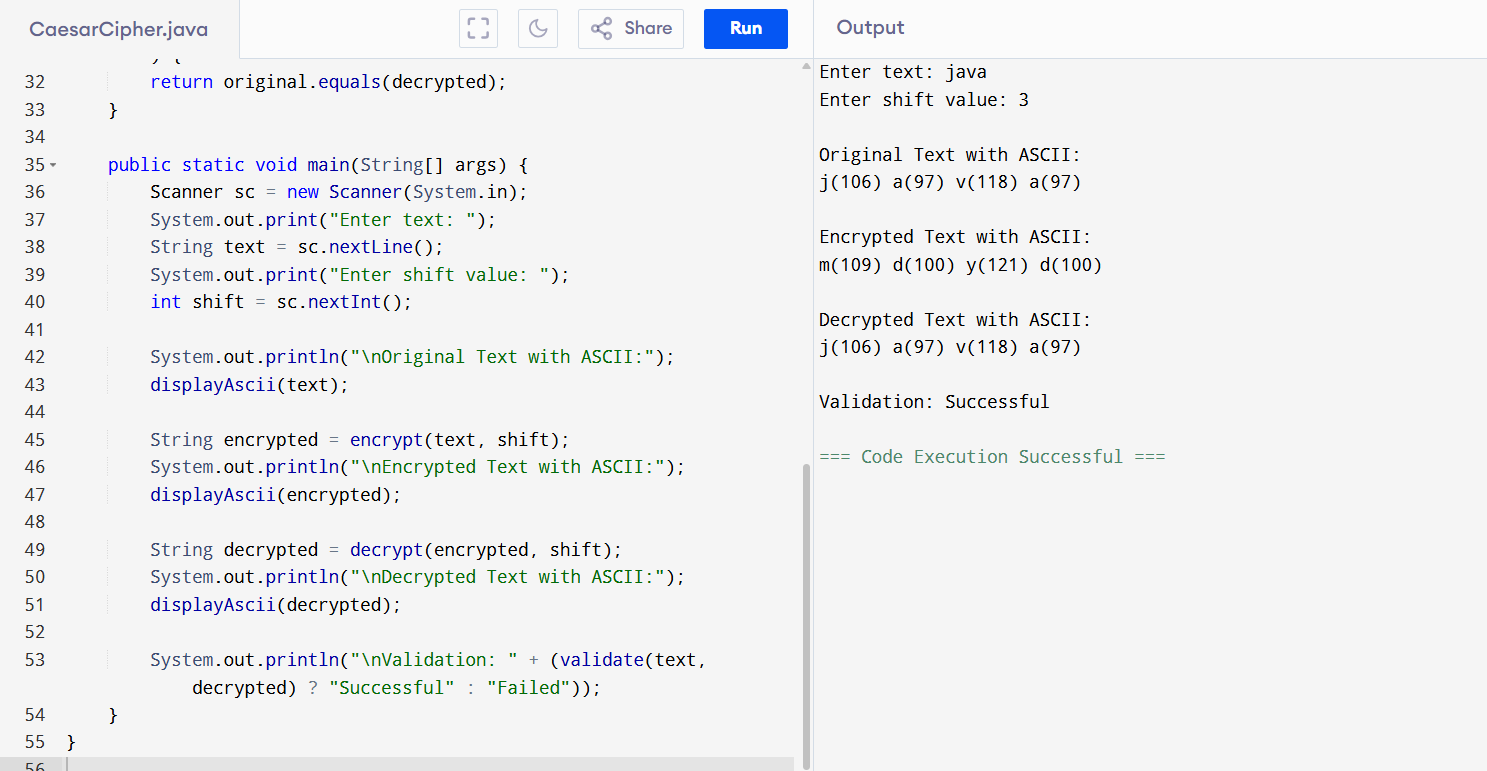
System.out.println("\nDecrypted Text with ASCII:");

displayAscii(decrypted);

System.out.println("\nValidation: " + (validate(text, decrypted) ? "Successful" : "Failed"));

}

}



5-

import java.util.\*;

public class EmailAnalyzer {

public static boolean validate(String email) {

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

int lastAt = email.lastIndexOf('@');

if (at == -1 || at != lastAt) return false;

int dot = email.indexOf('.', at);

if (dot == -1) return false;

String user = email.substring(0, at);

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

if (user.isEmpty() || domain.isEmpty()) return false;

return true;

}

public static Map<String, String> extract(String email) {

Map<String, String> parts = new HashMap<>();

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

int dot = email.lastIndexOf('.');

String user = email.substring(0, at);

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

String domainName = domain.substring(0, domain.lastIndexOf('.'));

String ext = email.substring(dot + 1);

parts.put("user", user);

parts.put("domain", domain);

parts.put("domainName", domainName);

parts.put("ext", ext);

return parts;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

System.out.print("Enter number of emails: ");

int n = sc.nextInt();

sc.nextLine();

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

System.out.print("Enter email " + (i+1) + ": ");

emails.add(sc.nextLine());

}

int validCount = 0, invalidCount = 0, totalUserLen = 0;

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

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

System.out.printf("%-25s %-15s %-20s %-15s %-10s %-10s\n", "Email", "Username", "Domain", "Domain Name", "Ext", "Valid?");

for (String email : emails) {

boolean valid = validate(email);

if (valid) {

validCount++;

Map<String, String> parts = extract(email);

String user = parts.get("user");

String domain = parts.get("domain");

String domainName = parts.get("domainName");

String ext = parts.get("ext");

totalUserLen += user.length();

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

System.out.printf("%-25s %-15s %-20s %-15s %-10s %-10s\n", email, user, domain, domainName, ext, "Valid");

} else {

invalidCount++;

System.out.printf("%-25s %-15s %-20s %-15s %-10s %-10s\n", email, "-", "-", "-", "-", "Invalid");

}

}

String mostCommonDomain = "-";

int maxCount = 0;

for (Map.Entry<String,Integer> e : domainCount.entrySet()) {

if (e.getValue() > maxCount) {

maxCount = e.getValue();

mostCommonDomain = e.getKey();

}

}

double avgUserLen = validCount > 0 ? (double) totalUserLen / validCount : 0;

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

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

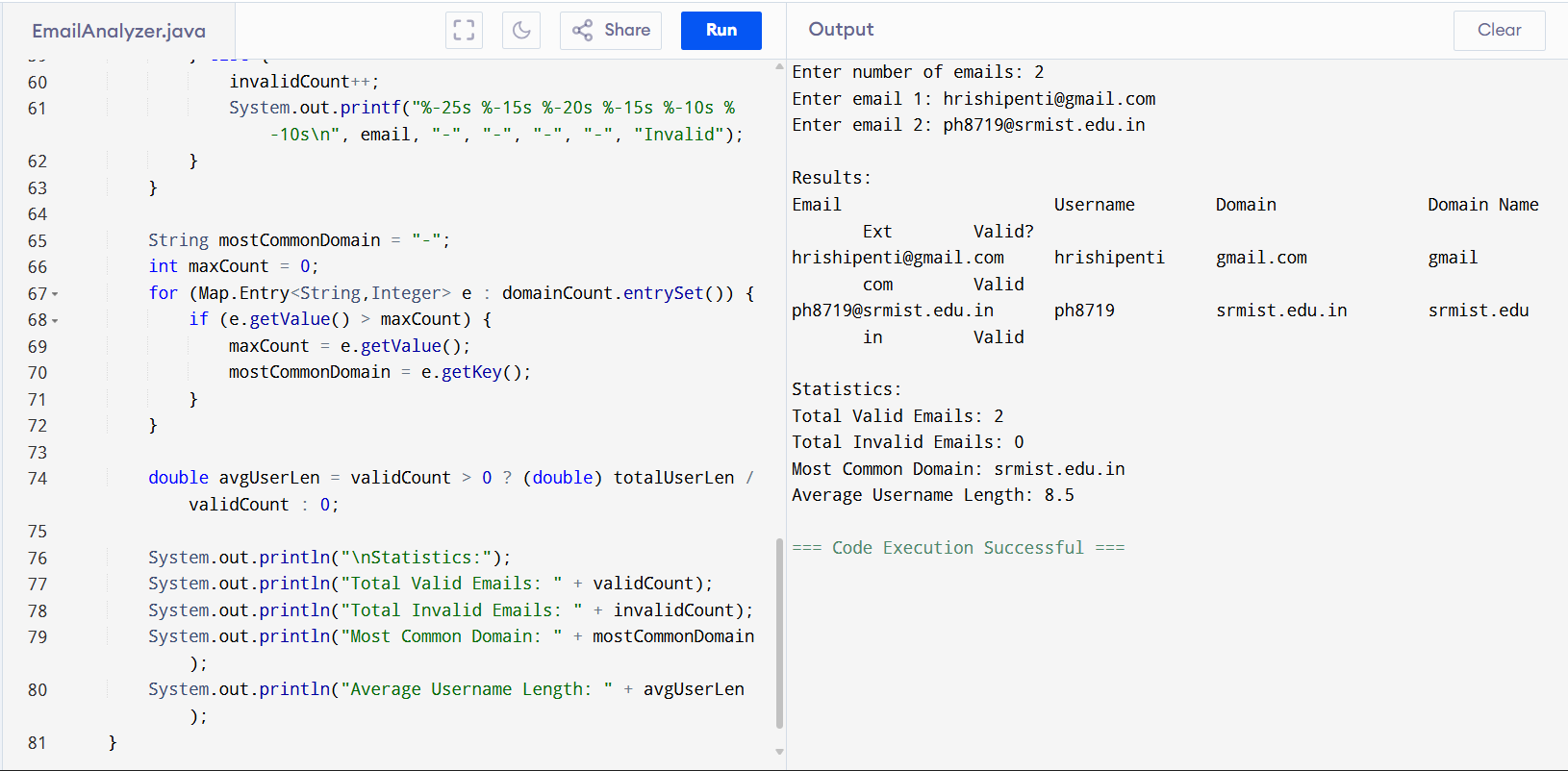
System.out.println("Total Invalid Emails: " + invalidCount);

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

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

}

}



6-

import java.util.\*;

public class TextFormatterBeginner {

public static ArrayList<String> getWords(String text) {

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

int start = 0;

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

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

if (start != i) {

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

}

start = i + 1;

}

}

if (start < text.length()) {

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

}

return words;

}

public static ArrayList<String> justify(ArrayList<String> words, int width) {

ArrayList<String> lines = new ArrayList<>();

int i = 0;

while (i < words.size()) {

int len = words.get(i).length();

int j = i + 1;

while (j < words.size() && len + 1 + words.get(j).length() <= width) {

len += 1 + words.get(j).length();

j++;

}

StringBuilder line = new StringBuilder();

int wordCount = j - i;

if (j == words.size() || wordCount == 1) {

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

line.append(words.get(k));

if (k != j - 1) line.append(" ");

}

while (line.length() < width) line.append(" ");

} else {

int totalSpaces = width - (len - (wordCount - 1));

int spaceBetween = totalSpaces / (wordCount - 1);

int extra = totalSpaces % (wordCount - 1);

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

line.append(words.get(k));

if (k != j - 1) {

for (int s = 0; s < spaceBetween; s++) line.append(" ");

if (extra > 0) {

line.append(" ");

extra--;

}

}

}

}

lines.add(line.toString());

i = j;

}

return lines;

}

public static ArrayList<String> center(ArrayList<String> words, int width) {

ArrayList<String> lines = new ArrayList<>();

StringBuilder line = new StringBuilder();

for (String word : words) {

if (line.length() + word.length() + 1 > width) {

String l = line.toString().trim();

int padding = width - l.length();

int left = padding / 2;

int right = padding - left;

StringBuilder newLine = new StringBuilder();

for (int i = 0; i < left; i++) newLine.append(" ");

newLine.append(l);

for (int i = 0; i < right; i++) newLine.append(" ");

lines.add(newLine.toString());

line = new StringBuilder();

}

line.append(word).append(" ");

}

if (line.length() > 0) {

String l = line.toString().trim();

int padding = width - l.length();

int left = padding / 2;

int right = padding - left;

StringBuilder newLine = new StringBuilder();

for (int i = 0; i < left; i++) newLine.append(" ");

newLine.append(l);

for (int i = 0; i < right; i++) newLine.append(" ");

lines.add(newLine.toString());

}

return lines;

}

public static void display(ArrayList<String> lines) {

int n = 1;

for (String l : lines) {

System.out.println(n + ". " + l + " (" + l.length() + ")");

n++;

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String text = sc.nextLine();

System.out.print("Enter width: ");

int width = sc.nextInt();

ArrayList<String> words = getWords(text);

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

System.out.println(text);

ArrayList<String> justified = justify(words, width);

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

display(justified);

ArrayList<String> centered = center(words, width);

System.out.println("\nCenter Aligned Text:");

display(centered);

}

}

