**КОД**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Лабораторна робота №3

\* Тема: Рядки в мові програмування Java.

\* Мета: Ознайомлення з рядками та використання основних методів їх обробки

\* в мові програмування Java. Здобуття навичок у використанні рядків

\* в мові програмування Java.

\*

\* Завдання:

\* 1) Визначити C3 як остачу від ділення номера залікової книжки студента на 3,

\* C17 як остачу від ділення номера залікової книжки студента на 17.

\* 2) В залежності від C3 визначити тип текстових змінних:

\* C3 Тип

\* 0 -> StringBuilder

\* 1 -> StringBuffer

\* 2 -> String

\* 3) В залежності від C17 визначити дію з рядком (див. таблицю в умові).

\* 4) Створити клас, який складається з виконавчого методу, що виконує дію

\* з текстовим рядком (п.3), тип якого визначено варіантом (п.2).

\* Необхідно обробити всі виключні ситуації, що можуть виникнути під час

\* виконання програмного коду. Всі змінні повинні бути описані

\* та значення їх задані у виконавчому методі.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class Lab3 {

// Головний виконавчий метод

public static void main(String[] args) {

// 1. Задаємо номер залікової книжки (прикладове значення):

int studentID = 12345;

// 2. Обчислюємо C3 і C17

int C3 = studentID % 3; // Для визначення типу рядка

int C17 = studentID % 17; // Для вибору дії з рядком

// 3. Задаємо тестовий текст (можна змінити на будь-який інший)

// ВАЖЛИВО: Зверніть увагу, що речення розділяються крапками, знаками питання, оклику тощо.

// Для прикладу візьмемо декілька речень:

String initialText = "Привіт! Як твої справи? Все добре. Можливо, є якісь питання?";

// 4. Створюємо змінну для обробки рядка з урахуванням C3:

// - якщо C3 = 0 -> StringBuilder

// - якщо C3 = 1 -> StringBuffer

// - якщо C3 = 2 -> String

// Для універсальності, будемо зберігати в Object, а потім кастити.

Object textObject = null;

try {

switch (C3) {

case 0: // StringBuilder

textObject = new StringBuilder(initialText);

System.out.println("Тип вибраний: StringBuilder");

break;

case 1: // StringBuffer

textObject = new StringBuffer(initialText);

System.out.println("Тип вибраний: StringBuffer");

break;

case 2: // String

textObject = new String(initialText);

System.out.println("Тип вибраний: String (immutable)");

break;

default:

// Хоча за умовою задачі C3 завжди буде 0, 1 або 2,

// але на випадок непередбаченої помилки:

throw new IllegalArgumentException("Некоректне значення C3: " + C3);

}

} catch (Exception e) {

System.err.println("Помилка при створенні рядка відповідного типу: " + e.getMessage());

return; // Завершуємо виконання, якщо сталася критична помилка

}

// 5. Дія з рядком залежно від C17

// Таблиця з умови (0..16). Зробимо спрощену реалізацію:

// Для демонстрації повного коду - кожну дію розмістимо в окремому кейсі.

// УВАГА: Тут приведені лише приклади реалізації. За бажанням можна допрацювати.

try {

switch (C17) {

case 0:

// Знайти найбільшу кількість речень заданого тексту, в яких є однакові слова.

// (Демонстраційна реалізація)

doCase0(textObject);

break;

case 1:

// Вивести всі речення заданого тексту в порядку зростання кількості слів у них.

doCase1(textObject);

break;

case 2:

// Знайти таке слово в першому реченні заданого тексту, якого немає в жодному з наступних.

doCase2(textObject);

break;

case 3:

// В усіх питальних реченнях знайти та надрукувати без повторень слова заданої довжини.

// (З умови: "В усіх питальних реченнях заданого тексту знайти та надрукувати

// без повторень слова заданої довжини.")

// Для прикладу, шукаємо слова довжиною = 4

doCase3(textObject, 4);

break;

case 4:

// В кожному реченні поміняти місцями перше та останнє слово.

doCase4(textObject);

break;

case 5:

// Надрукувати слова без повторень заданого тексту в алфавітному порядку за першою літерою.

doCase5(textObject);

break;

case 6:

// Відсортувати слова заданого тексту за зростанням кількості голосних літер.

doCase6(textObject);

break;

case 7:

// Відсортувати слова, що починаються з голосної, за другою літерою.

doCase7(textObject);

break;

case 8:

// Відсортувати слова за зростанням кількості в них заданої літери.

doCase8(textObject, 'а');

break;

case 9:

// Задано текст та масив слів. Підрахувати, у скількох реченнях зустрічається кожне слово з масиву.

doCase9(textObject);

break;

case 10:

// З кожного речення видалити підрядок найбільшої довжини, що починається та закінчується заданими літерами.

doCase10(textObject, 'п', 'и');

break;

case 11:

// Із заданого тексту видалити всі слова визначеної довжини, що починаються з приголосної літери.

doCase11(textObject, 3);

break;

case 12:

// Відсортувати слова за кількістю входжень визначеного символу.

doCase12(textObject, 'о');

break;

case 13:

// Знайти підрядок максимальної довжини, що є паліндромом.

doCase13(textObject);

break;

case 14:

// В кожному слові видалити всі наступні входження першої літери цього слова.

doCase14(textObject);

break;

case 15:

// Замінити слова заданої довжини визначеним рядком.

doCase15(textObject, 5, "[REPLACED]");

break;

case 16:

// В кожному слові видалити всі попередні входження останньої літери цього слова.

doCase16(textObject);

break;

default:

// Якщо значення C17 > 16, можна повідомити, що операція не визначена

System.out.println("Операція з C17=" + C17 + " не визначена умовами завдання.");

}

} catch (Exception e) {

// Обробка будь-яких інших виняткових ситуацій

System.err.println("Під час обробки виникла помилка: " + e.getMessage());

}

}

/\*

\* Нижче наведено демонстраційні (мінімальні) реалізації методів:

\* У реальних умовах ці методи слід доопрацьовувати відповідно до вимог.

\*/

// CASE 0: Знайти найбільшу кількість речень, в яких є однакові слова.

private static void doCase0(Object textObject) {

System.out.println("CASE 0: Знайти найбільшу кількість речень, в яких є однакові слова (демо-версія).");

// Демонстраційний вивід:

String text = objectToString(textObject);

// Спрощено: просто розбити на речення та імітувати пошук

String[] sentences = text.split("[.!?]+");

// Тут могла бути реалізація пошуку

System.out.println("Кількість речень: " + sentences.length);

}

// CASE 1: Вивести всі речення в порядку зростання кількості слів.

private static void doCase1(Object textObject) {

System.out.println("CASE 1: Вивести всі речення в порядку зростання кількості слів (демо-версія).");

String text = objectToString(textObject);

String[] sentences = text.split("[.!?]+");

// Рахуємо слова в кожному реченні

java.util.List<String> list = new java.util.ArrayList<>();

for (String s : sentences) {

list.add(s.trim());

}

// Сортуємо за кількістю слів

list.sort((s1, s2) -> {

int w1 = s1.split("\\s+").length;

int w2 = s2.split("\\s+").length;

return Integer.compare(w1, w2);

});

// Виводимо

for (String s : list) {

System.out.println("[" + s + "] - слів: " + s.split("\\s+").length);

}

}

// CASE 2: Знайти слово в першому реченні, якого немає в інших.

private static void doCase2(Object textObject) {

System.out.println("CASE 2: Знайти таке слово в першому реченні, якого немає в жодному з наступних (демо-версія).");

String text = objectToString(textObject);

String[] sentences = text.split("[.!?]+");

if (sentences.length == 0) {

System.out.println("У тексті немає жодного речення.");

return;

}

String firstSentence = sentences[0].trim();

String[] firstWords = firstSentence.split("\\s+");

StringBuilder otherSentences = new StringBuilder();

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

otherSentences.append(sentences[i]).append(" ");

}

String allOther = otherSentences.toString();

for (String word : firstWords) {

// Перевірка, чи зустрічається word у інших реченнях

// Для простоти порівнюємо без урахування регістру

if (!allOther.toLowerCase().contains(word.toLowerCase())) {

System.out.println("Слово, що зустрічається тільки в першому реченні: " + word);

return;

}

}

System.out.println("Усі слова з першого речення зустрічаються і в наступних.");

}

// CASE 3: В усіх питальних реченнях знайти і вивести (без повторень) слова заданої довжини.

private static void doCase3(Object textObject, int wordLength) {

System.out.println("CASE 3: В усіх питальних реченнях знайти і надрукувати без повторень слова довжини = " + wordLength);

String text = objectToString(textObject);

// Розіб’ємо текст на речення, які закінчуються знаком питання

// Припустимо, що питальне речення має форму "...?"

String[] questionSentences = text.split("[?]");

// questionSentences[i] - це рядок без самого знаку "?", отже треба перевірити, чи речення справді питальне.

// У реальній задачі треба точніше парсити, але тут для демо - досить.

// множина (set) для унікальних слів

java.util.Set<String> uniqueWords = new java.util.HashSet<>();

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

// Якщо це речення було перед знаком "?", в ньому можуть бути інші розділові знаки.

// Врахуємо, що останній елемент після split теж повертається, якщо текст закінчувався "?",

// та може виявитися непитальним, якщо нема "?". Перевіримо приблизно:

if (text.contains(questionSentences[i] + "?")) {

// Тобто речення i дійсно було перед знаком '?'

String sentenceTrimmed = questionSentences[i].trim();

// Видалимо зайві знаки пунктуації (., ! тощо) для простоти

sentenceTrimmed = sentenceTrimmed.replaceAll("[!.,]", "");

// Розбиваємо на слова

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

for (String w : words) {

if (w.length() == wordLength) {

uniqueWords.add(w.toLowerCase()); // додамо в Set в нижньому регістрі

}

}

}

}

System.out.println("Слова довжини " + wordLength + " в усіх питальних реченнях (без повторень):");

for (String w : uniqueWords) {

System.out.println(w);

}

}

// CASE 4: В кожному реченні поміняти місцями перше та останнє слово.

private static void doCase4(Object textObject) {

System.out.println("CASE 4: В кожному реченні змінити місцями перше та останнє слово (демо-версія).");

String text = objectToString(textObject);

String[] sentences = text.split("(?<=[.!?])");

// Регекс (?<=[.!?]) розбиває текст з урахуванням крапки, знаку питання чи оклику,

// але зберігає ці роздільники у результаті (щоб потім відновити).

StringBuilder result = new StringBuilder();

for (String s : sentences) {

// Очищуємо від кінцевих пробілів

String trimmed = s.trim();

// Визначаємо роздільник речення (наприклад, ".", "?", "!")

String delimiter = "";

if (trimmed.endsWith(".") || trimmed.endsWith("?") || trimmed.endsWith("!")) {

delimiter = trimmed.substring(trimmed.length() - 1);

trimmed = trimmed.substring(0, trimmed.length() - 1);

}

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

if (words.length > 1) {

// Міняємо місцями перше і останнє слово

String temp = words[0];

words[0] = words[words.length - 1];

words[words.length - 1] = temp;

}

// Збираємо назад

String updated = String.join(" ", words) + delimiter + " ";

result.append(updated);

}

// Виводимо оновлений текст

System.out.println("Оновлений текст:\n" + result.toString().trim());

}

// CASE 5: Надрукувати слова без повторень в алфавітному порядку за першою літерою.

private static void doCase5(Object textObject) {

System.out.println("CASE 5: Надрукувати слова без повторень, відсортовані за першою літерою (демо-версія).");

String text = objectToString(textObject);

// Видалимо зайві знаки пунктуації

text = text.replaceAll("[.,!?]", "");

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

java.util.Set<String> set = new java.util.HashSet<>();

for (String w : words) {

set.add(w.toLowerCase());

}

// Сортуємо за першою літерою

java.util.List<String> list = new java.util.ArrayList<>(set);

list.sort((a, b) -> {

// Якщо перша літера однакова, можна далі сортувати за алфавітом повністю

// але для демо - тільки першу літеру

return Character.compare(a.charAt(0), b.charAt(0));

});

for (String w : list) {

System.out.println(w);

}

}

// CASE 6: Відсортувати слова за зростанням кількості голосних літер.

private static void doCase6(Object textObject) {

System.out.println("CASE 6: Відсортувати слова за зростанням кількості голосних (демо-версія).");

String text = objectToString(textObject).replaceAll("[.,!?]", "");

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

java.util.List<String> list = new java.util.ArrayList<>();

for (String w : words) {

list.add(w);

}

list.sort((a, b) -> {

int vowelsA = countVowels(a);

int vowelsB = countVowels(b);

return Integer.compare(vowelsA, vowelsB);

});

System.out.println("Сортування слів за кількістю голосних:");

for (String w : list) {

System.out.println(w + " (голосних: " + countVowels(w) + ")");

}

}

// CASE 7: Відсортувати слова, що починаються з голосної, за другою літерою.

private static void doCase7(Object textObject) {

System.out.println("CASE 7: Сортувати слова, що починаються з голосної, за другою літерою (демо-версія).");

String text = objectToString(textObject).replaceAll("[.,!?]", "");

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

java.util.List<String> startsWithVowel = new java.util.ArrayList<>();

for (String w : words) {

if (isVowelStart(w)) {

startsWithVowel.add(w);

}

}

// Сортуємо за другою літерою (якщо вона існує)

startsWithVowel.sort((a, b) -> {

if (a.length() < 2 && b.length() < 2) return 0;

if (a.length() < 2) return -1;

if (b.length() < 2) return 1;

return Character.compare(a.charAt(1), b.charAt(1));

});

System.out.println("Слова, що починаються з голосної, відсортовані за другою літерою:");

for (String w : startsWithVowel) {

System.out.println(w);

}

}

// CASE 8: Відсортувати слова за зростанням кількості в них заданої літери.

private static void doCase8(Object textObject, char ch) {

System.out.println("CASE 8: Відсортувати слова за кількістю літери '" + ch + "' (демо-версія).");

String text = objectToString(textObject).replaceAll("[.,!?]", "");

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

java.util.List<String> list = new java.util.ArrayList<>();

for (String w : words) {

list.add(w);

}

list.sort((a, b) -> {

int countA = countChar(a, ch);

int countB = countChar(b, ch);

return Integer.compare(countA, countB);

});

for (String w : list) {

System.out.println(w + " (кількість '" + ch + "': " + countChar(w, ch) + ")");

}

}

// CASE 9: Задано текст та масив слів. Підрахувати, у скількох реченнях зустрічається кожне слово з масиву.

private static void doCase9(Object textObject) {

System.out.println("CASE 9: Підрахувати, у скількох реченнях зустрічається кожне слово з масиву (демо-версія).");

String text = objectToString(textObject);

String[] sentences = text.split("[.!?]+");

// Приклад масиву слів:

String[] targetWords = { "привіт", "твої", "питання" };

for (String tw : targetWords) {

int count = 0;

for (String s : sentences) {

if (s.toLowerCase().contains(tw.toLowerCase())) {

count++;

}

}

System.out.println("Слово '" + tw + "' зустрічається в " + count + " реченнях.");

}

}

// CASE 10: З кожного речення видалити підрядок найбільшої довжини, що починається і закінчується заданими літерами.

private static void doCase10(Object textObject, char start, char end) {

System.out.println("CASE 10: Видалити найбільший підрядок між '" + start + "' та '" + end + "' (демо-версія).");

// Проста (спрощена) реалізація: знайти у кожному реченні найбільший підрядок, що починається з start і закінчується end

String text = objectToString(textObject);

String[] sentences = text.split("(?<=[.!?])");

StringBuilder result = new StringBuilder();

for (String sentence : sentences) {

String modified = sentence;

// Шукаємо найбільший підрядок

int startIndex = modified.toLowerCase().indexOf(Character.toLowerCase(start));

int endIndex = modified.toLowerCase().lastIndexOf(Character.toLowerCase(end));

if (startIndex != -1 && endIndex != -1 && endIndex > startIndex) {

String toRemove = modified.substring(startIndex, endIndex + 1);

modified = modified.replace(toRemove, "");

}

result.append(modified);

}

System.out.println("Результат:\n" + result.toString());

}

// CASE 11: Із заданого тексту видалити всі слова визначеної довжини, що починаються з приголосної.

private static void doCase11(Object textObject, int length) {

System.out.println("CASE 11: Видалити всі слова довжини " + length + ", що починаються з приголосної (демо).");

String text = objectToString(textObject);

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

StringBuilder sb = new StringBuilder();

for (String w : words) {

String cleared = w.replaceAll("[.,!?]", ""); // Приберемо пунктуацію для перевірки

if (cleared.length() == length && !isVowelStart(cleared)) {

// Пропускаємо це слово

} else {

sb.append(w).append(" ");

}

}

System.out.println("Результат:\n" + sb.toString());

}

// CASE 12: Відсортувати слова за кількістю входжень визначеного символу.

private static void doCase12(Object textObject, char target) {

System.out.println("CASE 12: Відсортувати слова за кількістю входжень символу '" + target + "' (демо).");

String text = objectToString(textObject).replaceAll("[.,!?]", "");

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

java.util.List<String> list = new java.util.ArrayList<>();

for (String w : words) {

list.add(w);

}

list.sort((a, b) -> {

int ca = countChar(a, target);

int cb = countChar(b, target);

return Integer.compare(ca, cb);

});

for (String w : list) {

System.out.println(w + " => " + countChar(w, target));

}

}

// CASE 13: Знайти підрядок максимальної довжини, що є паліндромом.

private static void doCase13(Object textObject) {

System.out.println("CASE 13: Знайти підрядок максимальної довжини, що є паліндромом (демо-версія).");

String text = objectToString(textObject).replaceAll("[\\s+]", "");

String longestPalindrome = "";

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

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

String sub = text.substring(i, j);

if (isPalindrome(sub) && sub.length() > longestPalindrome.length()) {

longestPalindrome = sub;

}

}

}

System.out.println("Найдовший паліндром: " + longestPalindrome);

}

// CASE 14: В кожному слові видалити всі наступні входження першої літери цього слова.

private static void doCase14(Object textObject) {

System.out.println("CASE 14: Видалити всі наступні входження першої літери у кожному слові.");

String text = objectToString(textObject);

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

StringBuilder sb = new StringBuilder();

for (String w : words) {

if (!w.isEmpty()) {

char first = w.charAt(0);

// Замінимо всі наступні входження (крім першого) на порожній рядок

// Треба зважати на регістр (для спрощення - усе до lowerCase)

String pattern = "(?i)" + Character.toLowerCase(first);

// Прибираємо лише наступні входження, а не перше.

// Спочатку збережемо першу літеру, потім замінимо у "хвості"

String tail = w.substring(1).replaceAll(pattern, "");

String reconstructed = w.substring(0, 1) + tail;

sb.append(reconstructed).append(" ");

}

}

System.out.println("Результат:\n" + sb.toString());

}

// CASE 15: Замінити слова заданої довжини визначеним рядком.

private static void doCase15(Object textObject, int length, String replacement) {

System.out.println("CASE 15: Замінити слова довжини " + length + " на '" + replacement + "'.");

String text = objectToString(textObject);

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

StringBuilder sb = new StringBuilder();

for (String w : words) {

String cleared = w.replaceAll("[.,!?]", "");

if (cleared.length() == length) {

sb.append(replacement).append(" ");

} else {

sb.append(w).append(" ");

}

}

System.out.println("Результат:\n" + sb.toString());

}

// CASE 16: В кожному слові видалити всі попередні входження останньої літери цього слова.

private static void doCase16(Object textObject) {

System.out.println("CASE 16: В кожному слові видалити всі попередні входження останньої літери (демо).");

String text = objectToString(textObject);

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

StringBuilder sb = new StringBuilder();

for (String w : words) {

if (!w.isEmpty()) {

char last = w.charAt(w.length() - 1);

String head = w.substring(0, w.length() - 1);

// Видаляємо усі входження last в head

String pattern = "(?i)" + Character.toLowerCase(last);

head = head.replaceAll(pattern, "");

// Збираємо назад

String newWord = head + last;

sb.append(newWord).append(" ");

}

}

System.out.println("Результат:\n" + sb.toString());

}

/\*

\* ===============================================

\* Допоміжні методи

\* ===============================================

\*/

// Допоміжний метод для перетворення Object (String, StringBuilder, StringBuffer) у String

private static String objectToString(Object obj) {

if (obj instanceof StringBuilder) {

return obj.toString();

} else if (obj instanceof StringBuffer) {

return obj.toString();

} else if (obj instanceof String) {

return (String) obj;

} else {

return String.valueOf(obj);

}

}

// Підрахунок голосних у слові

private static int countVowels(String word) {

int count = 0;

String vowels = "аеєиіїоуюяАЕЄИІЇОУЮЯaeiou"; // Додано і українські, і англійські голосні

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

if (vowels.indexOf(c) != -1) {

count++;

}

}

return count;

}

// Перевірка, чи слово починається з голосної

private static boolean isVowelStart(String word) {

if (word.isEmpty()) return false;

String vowels = "аеєиіїоуюяAEIOUaeiouАЕЄИІЇОУЮЯ";

return vowels.indexOf(word.charAt(0)) != -1;

}

// Підрахувати кількість входжень символу ch у рядку str (без урахування регістру)

private static int countChar(String str, char ch) {

int count = 0;

String lowerStr = str.toLowerCase();

char lowerCh = Character.toLowerCase(ch);

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

if (lowerStr.charAt(i) == lowerCh) count++;

}

return count;

}

// Перевірка, чи підрядок є паліндромом

private static boolean isPalindrome(String s) {

int left = 0;

int right = s.length() - 1;

while (left < right) {

if (s.charAt(left) != s.charAt(right)) return false;

left++;

right--;

}

return true;

}

}

**Пояснення**

### 

**Оголошення класу**Клас **Lab3** містить один головний метод main і набір додаткових методів.  
java  
Копировать код  
public class Lab3 {

public static void main(String[] args) {

...

}

// інші методи

}

1. Це стандартна структура Java-програми.
2. **Змінні та їх ініціалізація**
   * studentID – номер залікової книжки.
   * C3 = studentID % 3 – визначає тип рядкової змінної.
   * C17 = studentID % 17 – визначає дію (операцію) з рядком.
3. **Вибір типу рядкової змінної**Згідно з умовою завдання:
   * Якщо C3 = 0, то використовуємо **StringBuilder**.
   * Якщо C3 = 1, то використовуємо **StringBuffer**.
   * Якщо C3 = 2, то використовуємо **String**.
4. Для універсальності створюємо змінну textObject типу Object, а потім залежно від значення C3 присвоюємо їй відповідну реалізацію.
5. **Обробка винятків**Операція створення потрібного об’єкта (StringBuilder, StringBuffer, String) обгорнута в try / catch на випадок виникнення непередбачених помилок (наприклад, якщо в коді буде некоректне значення C3).
6. **Дії з рядком (switch по C17)**За умовою є 17 різних можливих операцій (для C17 від 0 до 16). У коді кожна операція реалізована окремим методом (наприклад, doCase0, doCase1, ...). Усередині switch (C17) викликається потрібний метод.
   * **Приклад**: Якщо C17 = 3, викликається метод doCase3(...), що реалізує завдання «В усіх питальних реченнях знайти та надрукувати без повторень слова заданої довжини».
7. **Методи doCaseX**У прикладі для кожного case показана **демонстраційна** реалізація конкретного завдання. Залежно від вимог до деталізації, методи можна розширювати, більш детально обробляти текст тощо.
8. **Допоміжні методи**
   * objectToString – конвертує Object у String, враховуючи можливі типи (String, StringBuilder, StringBuffer).
   * countVowels – підрахунок голосних у слові (українські та англійські голосні).
   * isVowelStart – перевірка, чи починається слово з голосної.
   * countChar – підрахунок кількості входжень певного символу в рядку.
   * isPalindrome – перевіряє, чи є рядок паліндромом.
9. **Результат виконання**Залежно від того, яке значення C17 згенерує обраний studentID, буде виконано відповідний блок коду з обробки рядка. У наведеному прикладі, якщо studentID = 12345, тоді:
   * C3 = 12345 % 3 = 0 → використовується **StringBuilder**.
   * C17 = 12345 % 17 = 3 → виконується метод doCase3(...), що шукає в питальних реченнях слова потрібної довжини.
10. **Обробка винятків у switch (C17)**Кожен виклик doCaseX(textObject) також обгорнуто в try/catch, щоб перехопити будь-які можливі винятки під час обробки тексту (наприклад, помилка розбиття рядка, вихід за межі масиву тощо).
11. **Розширення і доопрацювання**Код можна розширювати, враховуючи реальні потреби: точніший пошук слів, складніші правила пунктуації, обробку різних мов, регістрів, тощо. У прикладі для спрощення ми використовували прості регулярні вирази і базові методи String.

Уся програма (1) визначає номер студента, (2) обчислює потрібні значення C3 та C17, (3) обирає тип рядка, (4) виконує відповідну операцію над цим рядком, а (5) виводить результат у консоль.