

# Лабораторна робота №3

## Опрацювання рядків

### Завдання

1. Створити проект, що складається з двох класів: основного (Main) та класу для завдання відповідно до варіанту.
2. Виконати програму, та пересвідчитись, що дані зберігаються та коректно виводяться на екран відповідно до вказаних критеріїв.

### Варіанти завдань

В усіх варіантах завдань вважати, що текст складається з літер та пропусків. Словом будемо називати непорожню послідовність літер, яка не містить пропусків.

**Варіант 1.** Ввести з клавіатури ціле число  $k$  та деяку літеру (символ). Після введення цих даних, ввести один рядок тексту. В кожному слові тексту  $k$ -у літеру замінити введеним. Якщо  $k$  більше довжини слова, коригування не виконувати. Вивести змінений текст.

**Варіант 2.** Ввести один рядок тексту. В тексті кожен літеру замінити її порядковим номером в алфавіті. Вивести вхідний рядок та отриманий рядок наступним чином: при виведенні в одному рядку друкувати текст з двома пропусками між літерами, в наступному рядку внизу під кожною літерою друкувати її номер.

**Варіант 3.** Ввести текст, що складається з декількох слів. В тексті після літери  $P$  (або  $p$ ), якщо вона не остання в слові, може зустрічатися літера  $A$  (або  $a$ ). У такому випадку замінити її на літеру  $O$  (або  $o$ ). Вивести початковий та змінений рядок.

**Варіант 4.** Ввести текст, що складається з декількох слів, ввести рядок, що буде використовуватися для заміни та довжину слів, які треба замінити. В тексті слова вказаної довжини замінити введеним рядком. Довжина рядка, який використовуватиметься для заміни, може не збігатися з довжиною слова. Вивести змінений рядок.

**Варіант 5.** Ввести текст, короткий рядок та ціле число  $k$ . В тексті після  $k$ -го символу вставити заданий короткий рядок. Вивести рядок після опрацювання та ціле число – кількість слів у отриманому рядку.

**Варіант 6.** Ввести текст і два коротких рядки  $s$  та  $t$ . Після кожного слова тексту, яке закінчується рядком  $s$ , вставити вказане слово  $t$ . Вивести початковий рядок та рядок після опрацювання.

**Варіант 7.** Ввести текст, один символ та ціле число  $k$ . В кожному слові тексту, що має довжину більшу  $k$  видалити введений символ. Вивести рядок до та після опрацювання.

**Варіант 8.** Ввести текст, що крім літер та пропусків може містити інші символи. Вилучити з нього всі символи, крім пропусків, які не є літерами. Між послідовностями літер, що йдуть

безпосередньо одна за одною, залишити хоча би по одному пропуску. Вивести рядок до та після опрацювання.

**Варіант 9.** Ввести текст та ціле число – довжину слова. З тексту вилучити всі слова вказаної довжини, які починаються на приголосну літеру. Вивести початковий рядок та рядок після опрацювання.

**Варіант 10.** Ввести текст, що містить пару спеціальних символів, які вводяться (наприклад, дужки '(' та ')') або зірочки '\*'). Видалити з тексту всі символи між вказаними символами. Вивести початковий рядок та рядок після опрацювання та число - кількість слів у опрацьованому рядку.

### Додаткове завдання (для всіх варіантів)

Виконати теж саме завдання, вважаючи, що слова у текстах можуть містити лише літери, але відокремлюються не тільки пропусками, а й іншими символами, що не є літерами й цифрами (наприклад, знаки пунктуації).

## Клас String

Кожен рядок, створений за допомогою оператора `new` або за допомогою літерала (укладений в подвійні апострофи), є об'єктом класу `String`. Особливістю об'єкта класу `String` є те, що його значення не може бути змінено після створення об'єкту. При виклику будь-якого методу класу, результатом може бути новий рядок, але не змінюється початковий. Таким чином будь-який метод, що намагається змінити рядок приводить до створення нового об'єкта.

Клас `String` підтримує кілька конструкторів, наприклад:

```
String(),  
String(String str),  
String(byte asciiChar[]),  
String(char[] unicodeChar),  
String(StringBuffer sbuf),  
String(StringBuilder sbuild)
```

та інші.

Коли Java зустрічає літерал, укладений в подвійні лапки, автоматично створюється об'єкт типу `String`, на який можна встановити посилання. Таким чином, об'єкт класу `String` можна створити, присвоюючи посиланню на клас значення літерала, що вже існує, або за допомогою оператора `new` і конструктора, наприклад:

```
String s1 = "www.berkut.mk.ua";  
String s2 = new String("www.berkut.mk.ua ");
```

Клас `String` містить наступні (тут наведені основні, але не всі) методи для роботи з рядками:

`String concat(String s)` або “+” – злиття рядків;

`boolean equals(Object ob)` та `equalsIgnoreCase(String s)` – порівняння рядків з урахуванням та без урахування регістра відповідно;

`int compareTo(String s)` та `compareToIgnoreCase(String s)` – лексикографічне порівняння рядків з урахуванням та без урахування регістра. Метод виконує віднімання кодів символів рядка, що викликає метод, та рядка, що передається до методу і повертає ціле значення. Метод повертає значення нуль у випадку, якщо `equals()` повертає значення `true`;

`boolean contentEquals(StringBuffer sb)` – порівняння рядка та вмісту об'єкта типу `StringBuffer`;

`String substring(int n, int m)` – отримання з рядка підрядка, починаючи з позиції `n` (включаючи) до позиції `m` (не включаючи). Нумерація символів в рядку починається з нуля;

`String substring(int n)` – отримання з рядка підрядка, починаючи з позиції `n`;

`int length()` – визначення довжини рядка;

`int indexOf(char ch)` – визначення позиції символу у рядку;

`int indexOf(String s)` – визначення позиції підрядка у рядку;

`int indexOf(String s, int fromIndex)` – визначення позиції підрядка у рядку починаючи з вказаного індексу;

`int lastIndexOf(char ch)` – визначення останньої позиції символу у рядку;

`int lastIndexOf(String s)` – визначення останньої позиції підрядка у рядку;

`int lastIndexOf(String s, int fromIndex)` – визначення останньої позиції підрядка у рядку, але не більше вказаного індексу;

`static String valueOf(значення)` – перетворення змінної примітивного типу у рядок;

`String toUpperCase()/toLowerCase()` – перетворення всіх символів рядка, який викликає метод у верхній/нижній регістр;

`String replace(char c1, char c2)` – заміна у рядку всіх входжень першого символу другим символом;

`String replace(CharSequence target, CharSequence replacement)` – заміна у рядку всіх входжень першого символу другим символом;

`String trim()` – вилучення всіх пропусків на початку та в кінці рядка;

`char charAt(int position)` – отримання символу із вказаної позиції (нумерація з нуля);

`boolean isEmpty()` – повертає `true`, якщо довжина рядка дорівнює 0;

`static String format(String format, Object... args)` – генерує форматований рядок, отриманий з використанням формату.

`String[] split(String regex)` – пошук входження в рядок заданого регулярного виразу (розділителя або розділителів) та поділ початкового рядка відповідно до нього на масив рядків.

`String[] split(String regex, int limit)` – як і попередній метод, але кількість елементів у масиві не може перевищувати `limit`

`boolean startsWith(String prefix)` – перевірка, чи починається рядок із вказаного префіксу

`boolean endsWith(String suffix)` – перевірка, чи закінчується рядок на вказаний суфікс

## Класи `StringBuilder` та `StringBuffer`

Класи `StringBuilder` та `StringBuffer` є “близнюками” та за призначенням наближені до класу `String`, але, на відмінність від останнього, вміст та розміри об’єктів класів `StringBuilder` та `StringBuffer` можна змінювати. Тому, якщо потрібно виконувати багато операцій з перетворення рядків, треба в першу чергу, розглядати можливість використання саме об’єктів класів `StringBuilder` та `StringBuffer`. За допомогою відповідних методів та конструкторів об’єкти класів `StringBuffer`, `StringBuilder` та `String` можна перетворювати один до одного. Конструктор класу `StringBuffer` (так саме як і `StringBuilder`) може приймати як параметр об’єкт `String` або невід’ємний розмір буфера. Об’єкти цього класу можна перетворювати в об’єкт класу `String` методом `toString()` або за допомогою конструктора класу `String`.

Слід звернути увагу на такі методи:

`void setLength(int n)` – установка розміру буфера;

`void ensureCapacity(int minimum)` – установка гарантованого мінімального розміру буфера;

`int capacity()` – отримання поточного розміру буфера;

`StringBuilder append(параметри)` – додавання до вмісту об’єкта рядкового представлення аргументу, який може бути символом, значенням примітивного типу, масивом та рядком;

`StringBuilder insert(параметри)` – вставка символу, об’єкта або рядка в указану позицію;

`StringBuilder deleteCharAt(int index)` – вилучення символу;

`StringBuilder delete(int start, int end)` – вилучення підрядку;

`StringBuilder reverse()` – перестановка вмісту об’єкта у зворотному порядку;

`void setCharAt(int index, char ch)` – заміна символу у вказаній позиції.

В класах `StringBuilder` та `StringBuffer` присутні також методи, аналогічні методам класу `String`, такі як `replace()`, `substring()`, `charAt()`, `length()`, `getChars()`, `indexOf()` та інші