



Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра інформаційних систем та технологій

ЛАБОРАТОРНА РОБОТА № 5  
з дисципліни «Мова програмування Java»  
Тема: «I/O Streams»  
Варіант 119

**Виконала:**

Студентка групи ІА-31

Соколова Поліна

Мета роботи – створення додатків для роботи з файлами з використанням потоків вводу/виводу, опанування користувацької серіалізації, підключення до інтернет джерел та їх аналіз їх вмісту.

## Хід роботи

(у лр 3 виконано Middle Tasks)

Клас `FileManager` відповідає за роботу з файлами — читання, пошук потрібного рядка та серіалізацію результатів.

Метод `findLineWithMostWords()` відкриває текстовий файл для читання через `BufferedReader` (символьний потік), проходить по кожному рядку файлу, обчислює кількість слів (за допомогою розділення рядка по пробілах) і визначає рядок із максимальною кількістю слів. Після обробки всіх рядків повертає знайдений рядок.

Метод `saveResults()` виконує серіалізацію результатів за замовчуванням: зберігає рядок (об'єкт типу `String`) у файл за допомогою `ObjectOutputStream` (об'єктного потоку).

Клас `Encryptor` відповідає за шифрування та дешифрування файлів за заданим символьним ключем.

Метод `encryptFile()` читає вхідний файл через `FileInputStream`, створює вихідний потік `FileOutputStream` і фільтрований потік `EncryptFilterOutputStream`, який змінює кожен байт, додаючи до нього числове значення ключового символу (код ASCII). Потік записує змінені байти у зашифрований файл.

Метод `decryptFile()` робить зворотну операцію — створює `DecryptFilterInputStream`, який віднімає код ключового символу від кожного байта, відновлюючи вихідний вміст.

Використання `try-with-resources` забезпечує автоматичне закриття потоків після завершення операцій.

Клас TagCounter відповідає за підрахунок частоти появи HTML-тегів на вебсторінці.

Зчитує HTML-код сторінки за URL через BufferedReader (символьний потік, отриманий від url.openStream()), використовує регулярний вираз для пошуку назв тегів, зберігає результати у Map.

У класі Main результати сортуються двічі: за назвою тегу — Map.Entry.comparingByKey(), за частотою — Map.Entry.comparingByValue().

Клас Main керує логікою виконання програми та взаємодією з користувачем.

Зчитує шляхи до файлів і URL із клавіатури через Scanner.

Викликає FileManager.findLineWithMostWords() для пошуку рядка з максимальною кількістю слів.

Зчитує ключовий символ і викликає Encryptor.encryptFile() / Encryptor.decryptFile() для шифрування та дешифрування файлів.

Зчитує URL, викликає TagCounter.countTags() і виводить відсортований список тегів.

Виконує серіалізацію результатів (рядка з найбільшою кількістю слів) у файл через FileManager.saveResults().

У main() використовується узагальнений try-catch для повідомлення про помилки користувачу.

```
Введіть шлях до вхідного файлу: test.txt

Рядок із максимальною кількістю слів:
Цей рядок містить найбільшу кількість слів серед усіх рядків у цьому файлі.

Введіть ключовий символ для шифрування: k
Введіть ім'я файлу для збереження зашифрованих даних: encr.txt
Файл зашифровано
Введіть ім'я файлу для збереження дешифрованих даних: decr.txt
Файл дешифровано

Введіть URL для аналізу: https://kpi.ua/

Введіть шлях для збереження результатів: result.txt
Результати збережено у файл: result.txt
```

| Сортування лексикографічно: | Сортування за частотою: |
|-----------------------------|-------------------------|
| a : 274                     | h1 : 1                  |
| aside : 2                   | main : 1                |
| blockquote : 1              | title : 1               |
| body : 1                    | body : 1                |
| br : 17                     | head : 1                |
| div : 208                   | html : 1                |
| footer : 4                  | blockquote : 1          |
| h1 : 1                      | header : 1              |
| h2 : 10                     | i : 2                   |
| head : 1                    | aside : 2               |
| header : 1                  | footer : 4              |
| html : 1                    | link : 5                |
| i : 2                       | script : 5              |
| iframe : 6                  | iframe : 6              |
| img : 24                    | nav : 6                 |
| li : 227                    | p : 7                   |
| link : 5                    | meta : 8                |
| main : 1                    | time : 9                |
| meta : 8                    | h2 : 10                 |
| nav : 6                     | br : 17                 |
| p : 7                       | img : 24                |
| script : 5                  | span : 28               |
| span : 28                   | ul : 38                 |
| time : 9                    | div : 208               |
| title : 1                   | li : 227                |
| ul : 38                     | a : 274                 |

У результаті отримано файли:

test.txt

```

Це короткий рядок.
Цей рядок має трохи більше слів ніж попередній.
Цей рядок містить найбільшу кількість слів серед усіх рядків у цьому файлі.

```

encr.txt

```

|;@; <;%;)<l;)<n;%;#;$<<l<ь;@;);%™хu;@; ;$<<l<ь;@;);%<;';@<я<<n<l;)<r;#<;@<@;&<ч<y; <<m;
&<@;@<;(<@;!<;*);*;<l; ;@;(<@;$™хu;@; ;$<<l<ь;@;);%<;'<@<m<n;#<n<ч<;(<@;$;@<@;&<ч<y<o<;%<@;
&<ч;%<@<m<n<ч<<m;&<@;@<<m; <l; ;@<<o<m<@<r<<l<ь;@;%<@;@<<o<<c<ч;);'<o<<p;@;$;&<™хu

```

decr.txt

```

Це короткий рядок.
Цей рядок має трохи більше слів ніж попередній.
Цей рядок містить найбільшу кількість слів серед усіх рядків у цьому файлі.

```

result.txt

```

-н @t ЉP!PμPNº СЂСЊPГPsPє PјC-СѓC,РёC,Сђ PSP°PNºP±C-Р»СђСєCѓ РєC-Р»СђPєC-СѓC,Сђ СѓP»C-PI
СѓPμСЂPГ СѓCѓC-С... СЂСЊPГPєC-PI Сѓ СѓСђPSPјCѓ C,,P°PNºP»C-.

```

Висновок: у результаті виконання лабораторної роботи я реалізувала консольний застосунок для роботи з файлами, який демонструє використання потоків вводу/виводу, фільтрованих потоків, серіалізації даних та обробки мережесих ресурсів.