# Лабораторна робота 3

# ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ПОВЕДІНКИ

***Мета роботи:***

- засвоїти знання з методик тестування, керованого поведінкою (Behavior-Driven Testing), на основі історій та сценаріїв поведінки системи;
- отримати досвід роботи з інструментальним засобом тестування на основі поведінки JBehave.

***Інструментальні засоби і бібліотеки, необхідні для виконання роботи:***

- середовище розробки Eclipse
- засіб зборки проектів Maven
- засіб модульного тестування JUnit
- бібліотека JBehave

## Завдання

1) написати синтаксичний аналізатор на мові програмування Java, який виконує розпізнає команди і виконує матричні обчислення, реалізовані у лаб. роботі 1 (табл.1)
2) Протестувати програму, використовуючи методику тестування, основану на повединці

***Рекомендації до виконання задання 1***

- Матричні обчислення виконуються за допомогою методів класу, реалізованого в лабораторній роботі 1. Крім команд для матричних обчислень всього або окремих частин виразу, повинні бути команди обробки даних: ініціалізація, збереження, отримання збережених даних. Наприклад, для виразу $(A + B)^T$ діалог може таким:

```
> A = [[1, 2, 3], [4, 78.2, 6], [7, -4.56, 10]]

  1.00   2.00   3.00
  4.00  78.20   6.00
  7.00   -4.56  10.00

> A + [[1.45, -2.11, 3], [54.4, -8.27, 0.56],[74.5, 1.22, 3.45]]

   2.45  -0.11   6.00
  58.40  69.93   6.56
  81.50 -3.34   13.45

> B = [[5, 211, 5.36], [88.1, 7.4, 91.11], [4.2, -9.05, 58]]

   5.00 211.00   5.36
  88.10   7.40  91.11
   4.20  -9.05  58.00
```

```
> A + B
 6.00 213.00   8.36
92.10  85.60  97.11
11.20 -13.61  68.00

> (A + B) ^ T
  5.00 211.00   5.36
 88.10   7.40  91.11
  4.20  -9.05  58.00

> ([[1, 2, 3], [4, 78.2, 6], [7, -4.56, 10]] + [[5, 211, 5.36], [88.1, 7.4, 91.11],
[4.2, -9.05, 58]]) ^ T

 5.00 211.00   5.36
88.10   7.40  91.11
 4.20  -9.05  58.00


> (A + (B + [[15, -2.1, 0.88], [34, 7.1, 0.12]], [-1.87, 92.5, -5.8])) ^ T

5.00 211.00   5.36
88.10   7.40  91.11
4.20 -9.05  58.00
```

- синтаксичний аналізатор побудувати на основі регулярних виразів, використовуючи пакет **`java.util.regex`**.
- назви для операцій транспонування, визначення рангу, скалярного добутку, визначення детермінанту, обернення матриці можна обирати як спеціальними символами (як в прикладі вище), так і літерами;
- організувати перетворення і запис зчитаних даних у змінні типу Matrix або Vector
- регулярні вирази специфікують, які рядки будуть вважатися припустимими для введення, а які ні. Ці специфікації слід використати при побудові тестових даних. У разі введення даних слід вивести відповідне повідомлення, а діалог припиняти не треба;
- обробити помилки неправильного введення даних. В цій частині роботи не перевіряються помилки неправильного розміру матриці або відсутності операнду для виконання операції

***Рекомендації до виконання частини 2***

1) За допомогою засобу JBehave (http://jbehave.org/) написати тести та протестувати реалізацію лексичного аналізатора для роботи з матрицями та матричними операціями, побудованого у першій частині лабораторної роботи:

- описати сценарії для двох історій:
  - «Матриці, які вводяться або обчислюються, зберігаються у сховищі»
    Приблизний перелік сценаріїв для історії:
    - введення та збереження матриці;
    - введення матриці у неправильному форматі;
    - введення матриці з неправильної назвою;
    - запит матриці, яка є у сховищі;

- запит матриці, якої немає у сховищі;
- матриця з таким же ім'ям вже зберігається у сховищі;
- збереження результатів обчислень у сховищі

Перелік сценаріїв для історії можна редагувати залежно від особливостей реалізації лексичного аналізатора.

- «Обчислення виразу з матрицями»

  Приблизний перелік сценаріїв для історії:
  - вираз містить матриці у строковому форматі; вираз обчислюється без збереження результату;
  - вираз містить матриці у строковому форматі; вираз обчислюється і результат зберігається у сховищі;
  - вираз містить матриці у неправильному строковому форматі;
  - вираз посилається на матриці зі сховища; вираз обчислюється без збереження результату;
  - вираз посилається на матриці зі сховища; вираз обчислюється і результат зберігається у сховищі;
  - вираз посилається на матриці, яких немає у сховищі.

  Перелік сценаріїв для історії можна редагувати залежно від особливостей реалізації лексичного аналізатора.

2) описати і виконати модульні тести, використовуючи бібліотеку JBehave: відобразити кроки сценаріїв у програмну реалізацію, конфігурувати і запустити на виконання тести JBehave

## Індівідуальні завдання

*Таблиця*

| Варіант | Вираз | Варіант | Вираз |
|---|---|---|---|
| **1** | **2** | **3** | **4** |
| 1 | $A^T * B + C * k$ | 12 | $\left(B^T + C^{-1}\right) * k$ |
| 2 | $rank\left(A^{-1} + C * B\right)$ | 13 | $(V1 \times V2) * B^T$ |
| 3 | $k * A + B * C^{-1}$ | 14 | $rank(\lvert V\rvert * B)$ |
| 4 | $\det(A / k - C)$ | 15 | $A * B - C^{-1}$ |
| 5 | $\left(V1 \bullet V2\right) * A^{-1}$ | 16 | $V1 \bullet V2 + \det(A + B)$ |
| 6 | $C^T / rank(A)$ | 17 | $A / k * rank(B^T)$ |
| 7 | $C^{-1} - k * B^T$ | 18 | $\left(k * A^{-1} + C\right) / k$ |
| 8 | $\det\left(A + k * B\right)$ | 19 | $\left(\lvert V1\rvert + rank(A)\right) * B$ |
| 9 | $(V1 \bullet V2) * \lvert V3\rvert$ | 20 | $(B^{-1} - C) * k$ |
| 10 | $\left(A - B^T\right) * rank(C)$ | 21 | $(A + B)^T * k$ |
| 11 | $\det\left(A\right) * B - C^{-1}$ | 22 | $det\ (A + B^{-1})$ |

1

* Примітка до позначень, застосованих у виразі:

| | |
|---|---|
| A, B, C | матриці |
| V1, V2 , V3 | вектори |
| k | скалярне значення |
| $A + B$ | сума матриць |
| $\det(A)$ | детермінант матриці |
| $A / k$ | ділення матриці на скалярне значення |
| $A^{-1}$ | обернення матриці (A^1) |
| $A * B$ | добуток матриць |
| $A * k$ | добуток матриці і скалярного значення |
| $rank(A)$ | ранг матриці |
| $A - B$ | віднімання матриць |
| $A^T$ | транспозиція матриці (A^T) |
| $V1 \bullet V2$ | скалярний добуток двух векторів (V1*V2) |
| $\lvert V \rvert$ | модуль вектора |
| $V1 \times V2$ | векторний добуток двох векторів (V1xV2) |