Лабораторна робота 2

ПАРАМЕТРИЗАЦІЯ МОДУЛЬНИХ ТЕСТІВ

Мета: дослідити техніки параметризації модульних тестів та набути практичних навичок з їх застосування у тестовому каркасі JUnit

Інструментальні засоби і бібліотеки, необхідні для виконання роботи:

- середовище розробки Eclipse/ IntelliJ IDEA
- засіб модульного тестування JUnit5

Задання:

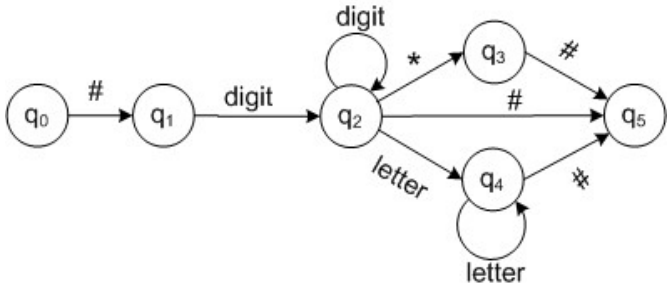
1) написати програму на мові програмування Java, яка будує скінчений автомат для розпізнавання рядка, заданого регулярним виразом (табл. 1);
2) описати параметризовані модульні тести і протестувати програму на різних наборах вхідних даних, використовуючи засоби

Порядок виконання роботи:

Задання 1.*Написати програму на мові програмування Java, яка будує скінчений автомат для розпізнавання рядка, заданого регулярним виразом*

- побудувати у вигляді графа скінчений автомат, який розпізнає текстовий образ, заданий регулярним виразом (табл. 1);

Приклад скінченого автомату для регулярного виразу ***#[4-8]+(*|[a-f]+)?#***



$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$
$q = \{ q_0 \}$
$F = \{ q_5 \}$
$DIGIT = \{ 4, 5, 6, 7, 8 \}$
$LETTER = \{ a, b, c, d, e, f \}$
$SHARP = \{ \# \}$
$ASTERISK = \{ * \}$
$\Sigma = P(DIGIT) \cup P(LETTER) \cup P(SHARP) \cup P(ASTERISK)$

- описати таблицю переходів для побудованого скінченного автомату;

Приклад функцій переходів і таблиці переходів

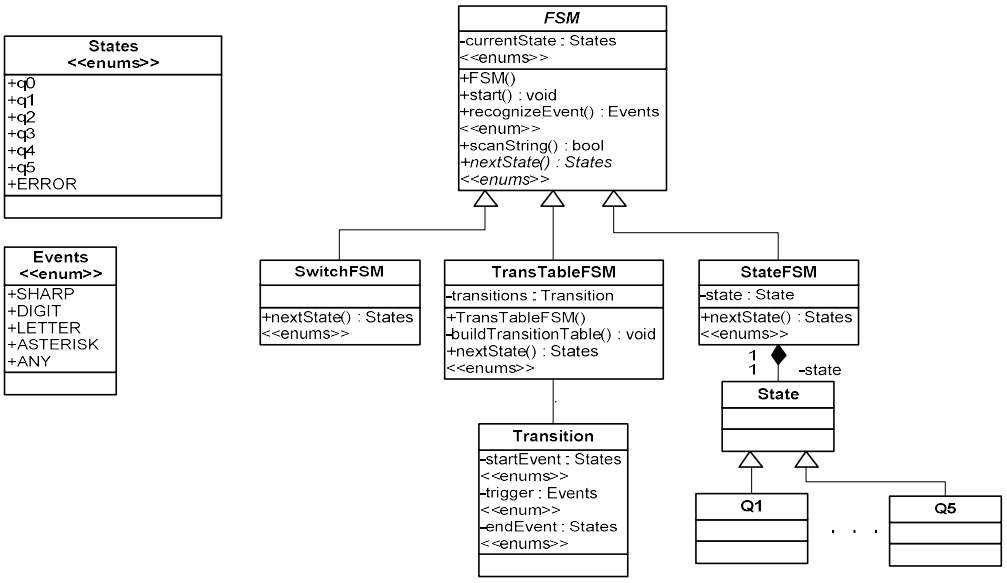
δ:

$(q_0, \text{SHARP}) \rightarrow q_1$
$(q_1, \text{DIGIT}) \rightarrow q_2$
$(q_2, \text{DIGIT}) \rightarrow q_2$
$(q_2, \text{ASTERISK}) \rightarrow q_3$
$(q_2, \text{LETTER}) \rightarrow q_4$
$(q_2, \text{SHARP}) \rightarrow q_5$
$(q_3, \text{SHARP}) \rightarrow q_5$
$(q_4, \text{LETTER}) \rightarrow q_4$
$(q_4, \text{SHARP}) \rightarrow q_5$

Q **Σ**	**Q_0**	**Q_1**	**Q_2**	**Q_3**	**Q_4**
SHARP	q_1	Error	q_5	q_5	q_5
ASTERISK	Error	Error	q_3	Error	Error
DIGIT	Error	q_2	q_2	Error	Error
LETTER	Error	Error	q_4	Error	q_4

- описати перелічувальний тип станів автомату **`States`** і подій **`Events`**
- описати три програмні реалізації синтаксичного аналізатору, який перевіряє рядок на відповідність заданому регулярному виразу. Аналізатор реалізує скінченний автомат трьома способами. За допомогою:
    1) вкладених операторів switch,
    2) таблиці переходів
    3) зразка проектування «State».

Структура програми у вигляді діаграми класів UML



Клас, який представляє скінченний автомат, зберігає поточний стан автомату у змінній **`currentState.`**

```
Abstract Data Type FSM
Поля:
State currentState          //поточний стан автомата

//початковий стан автомата при його створенні
currentState = q0
Методи:
//ідентифікація події
Event recognizeEvent (char ch)
// сканування рядка (аналіз або розпізнавання рядка)
bool scan (string str)
// обробка події (визначення наступного стану автомата)
State nextState(Event event)
```

Метод **scan** повертає **true**, якщо рядок розпізнається автоматом.

Для визначення наступного стану автомату для певного переходу (події) методі **scan** викликає абстрактний метод **nextState**, який реалізується у кожному з підкласів **SwitchFSM, TransTableFSM, StateFSM**.

Рекомендована література:

1. Paul Adamczyk. Selected Patterns for Implementing Finite State Machines. - University of Illinois at Urbana-Champaign, Department of Computer Science
   Режим доступу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.2987&rep=rep1&type=pdf

2. Р.С. Мартин, М. Мартин «Принципы, паттерны и методики гибкой разработки на языке C#». – СПб.: Символ-Плюс, 2011. – 768 с.

Задання 2. Протестувати програмні реалізації синтаксичного аналізатору, використовуючи параметризовані модульні тести у тестовому каркасі Junit5

- описати тестові класи і методи для перевірки методів класів, які реалізують скінчений автомат;
- позначити тестові методи як параметризовані (анотація `@ParameterizedTest`), а параметри для тестів описати через анотацію `@ValueSource`. Обрати параметри тестів – рядки, які ідентифікуються і не ідентифікуються синтаксичного аналізатором. Запустити тести на виконання і перевірити три реалізації аналізатора
- змінити постачальників даних для параметризованих тестових методів: надати тестові дані через метод (`@MethodSource`), csv-файл (`@CsvFileSource`) і описати власний постачальник даних (`@ArgumentsSource`). Запустити тести на виконання

Рекомендована література:

1. JUnit 5 User Guide: https://junit.org/junit5/docs/current/user-guide/#overview

Індівідуальні завдання

Таблиця 1

Варіант	Регулярний вираз	Варіант	Регулярний вираз
1	**2**	**3**	**4**
1	\++([\D]?\d*	12	\w+\.(\w+)?.#
2	\+[0-9]+\+%\+[0-9]+	13	\d+(%|*|([A-Z])+)#
3	_\d+#(#|&)[A-Z]*%	14	_(\+|-)[A-K]+\d{1,3}
4	(\+|-)[5-9]+([0-4]*|[AG]*)-	15	\^[A-Z]+\^*\^[^AZ^\d]+\^
5	<(\+|-)([0-5]+)|([P-Z])+>	16	\(([A-Z]*,\s)*[A-Z]*\)
6	+\d+[A-Z]*	17	\$([A-F]+|\d)*[^\d]+
7	(\d+E)+\d	18	(\d+\!)+\d+(e!n!d)*
8	\{(\d+|[A-Z]+)\}	19	([^A-Z])+(%|*|([AZ])+)#
9	\/[a-z]*[F-K]+	20	[A-Z]*_?([A-Z]+|\d+)
10	#(\d*|[a-f]*)@(\d*|[a-f]*)	21	(\+|-)[0-4]+([5-9]*|[al]*)-
11	[A-Z]+_?([A-Z]+|\d+)	22	\\d+(%|\~|([a-t])+)#