**Лабораторна робота №5**

з курсу «**Проектування інформаційних систем**»

на тему: «**Модульне тестуваня (Unit-тести) та рефакторинг.»**

Виконав:

студент групи ДА-61

Попов Григорій

**Мета роботи:** оволодіти навичками створення програмного забезпечення за метолологією TDD та ознайомитися з процедурами рефакторинга.

**Завдання:**

1. Розробити методику випробувань з використанням ISO/IEC/IEEE 29119.

2. Розробити код програми архітектурної моделі. Використовувати Test Driven Development.

3. Провести рефакторинг коду програми, щоб задовольнити вимоги технічного завдання.

**Оформлення результатів роботи**

1. **Методика випробувань з використанням ISO/IEC/IEEE 29119**

Буде виконано в рамках розрахунково-графічної роботи.

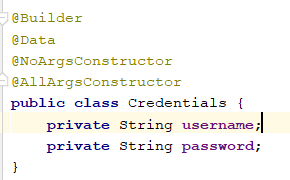
1. **Розробка код програми архітектурної моделі з використанням Test Driven Development**

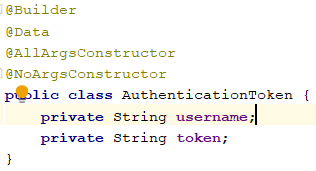
Сформуємо ітеративно AuthenticationService для імплементації бізнес-логіки необхідної для виконання тестового сценарію 2) та усіх інших, залежних від нього (згідно з результатами лабораторної роботи №2). Пригадаємо сценарій:

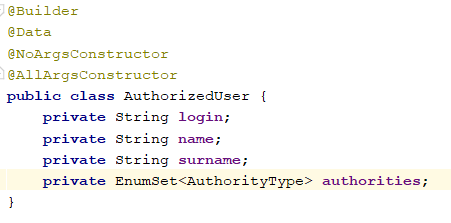
1. Авторизація

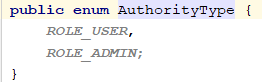
* Запустити додаток;
* Перейти на сторінку авторизації;
* Увести логін та пароль у відповідні текстові поля;
* Натиснути кнопку «Увійти»;
* Має відобразитися сторінка профілю користувача із помітками, що засвідчують, що вхід виконався успішно.

1. Спершу створимо необіхдні класи моделей:

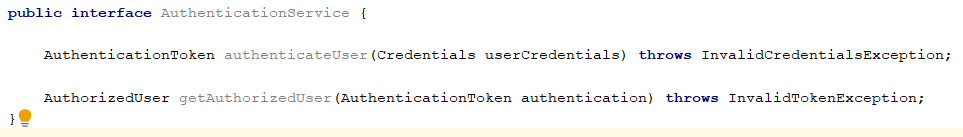


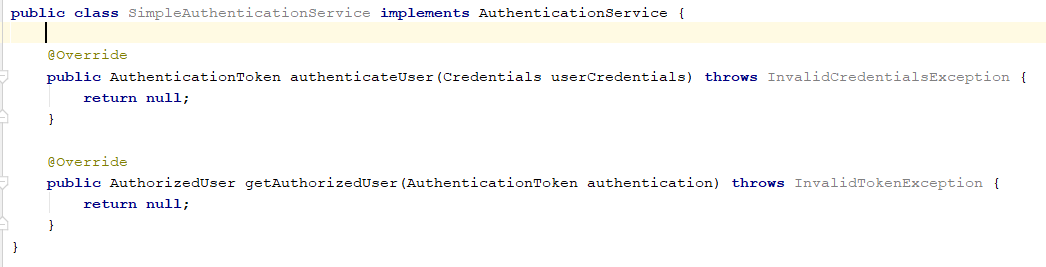




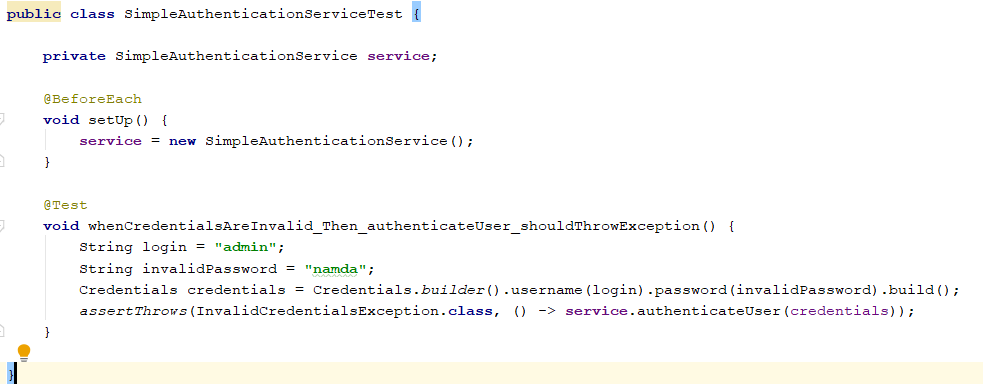


1. Створюємо інтерфейс сервісу та заглушку імплементації

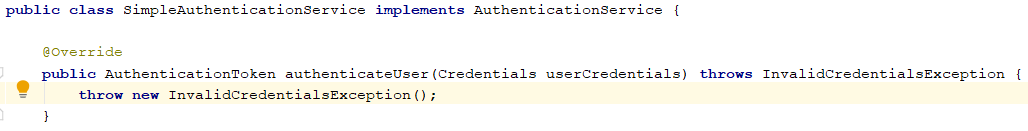




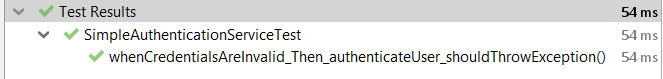
1. Перша ітерація
2. Тест:



На даному етапі тест виконується неуспішно. Тому проводимо рефакторинг:

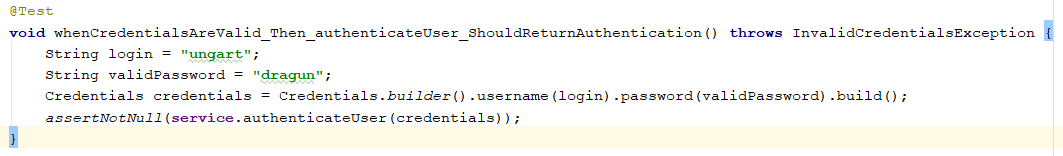


Після цього тест виконується успішно:

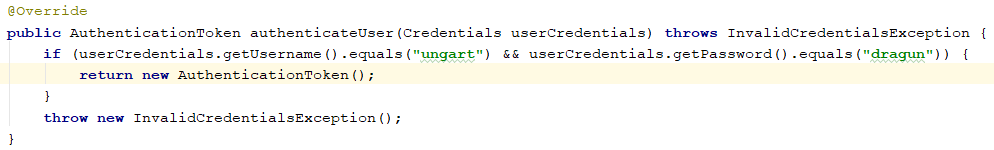


1. Друга ітерація

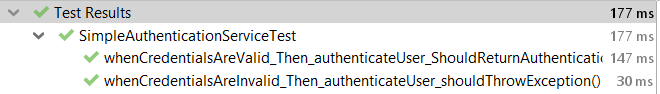
Новий тест:



Результат – Test Failed. Проводимо відповідний рефакторінг *з урахуванням першого тесту*:

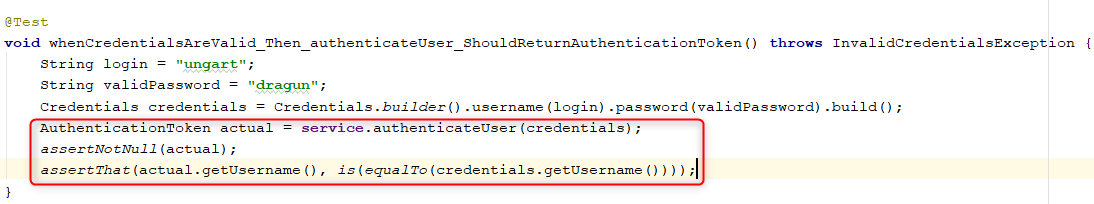


Запустимо обидва тести на виконання, результат успішний:

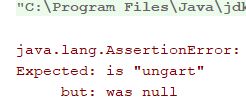


1. Третя ітерація: зажадаємо тепер також додаткової вимоги, щоб у моделі AuthenticationToken, що повертається з методу сервіса також було ненульове значення поля username, що було тим же самим, що й поле username у сутності Credentials, що передаються.

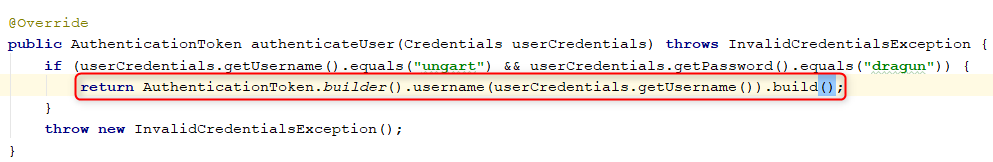
Для цього проведемо наступну модифікацію директиви assert у другому тесті:



Результат: Test failed

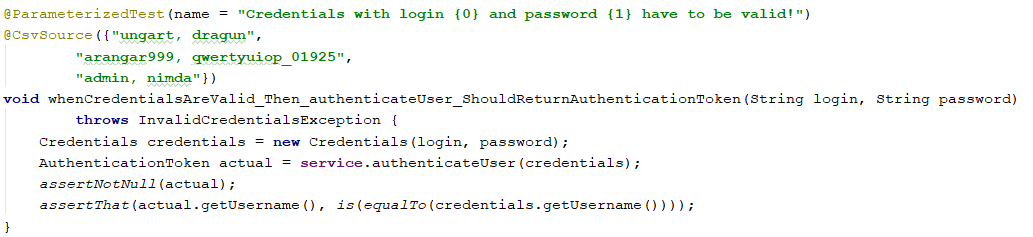


Проводимо рефакторинг:

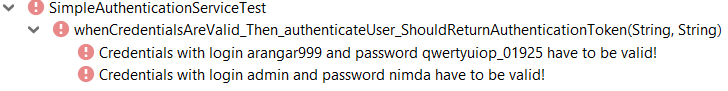


Відтепер тест виконується успішно.

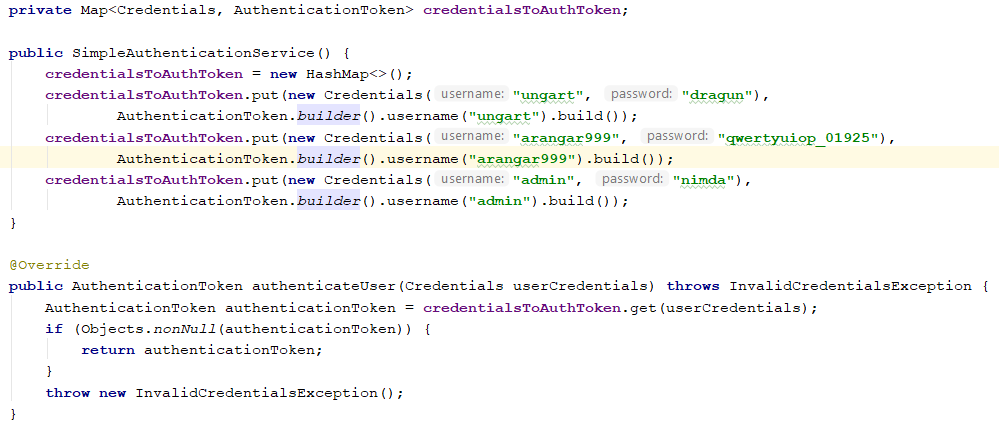
1. Четверта ітерація. Сформулюємо тепер конкретні вимоги до того, які саме користувачі зберігатимуться у нашому SimpleAuthenticationService. Для цього перетворимо другий тест у параметризований:



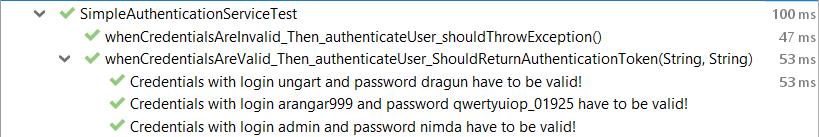
Результат: Test failed



Рефакторинг:



Результат: успішний



1. **Покриття сервісів застосунку Unit-тестами**

Модульні тести також для іншого сервісу (SimpleUserService) можна переглянути власне у вихідному коді додатку в цьому ж репозитарії.

**Висновок**

В межах даної лабораторної роботи було досліджено можливості Unit тестів в цілому та TDD в частинному випадку. Оцінено переваги TDD з точки зору модульного тестування, так само як і його недоліки з точки зору часу імплементації. Проведено необхідний рефакторинг коду, щоб він відповідав написаним модульним тестам.