МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Звіт

до лабораторної роботи №5

з дисципліни «Екстримальне програмування»

на тему «Створення тестів за допомогою JUnit»

Виконала

студентка групи КН-311

Мельничук М.П.

Прийняв

Щербак С.С.

Львів-2020

Мета: створення тестів за допомогою бібліотеки модульного тестування JUnit.

Теоретичні відомості

Керована тестами розробка (КТР), Розробка через тестування (англ. Test-driven development (TDD)) — технологія розробки програмного забезпечення, яка використовує короткі ітерації розробки, що починаються з попереднього написання тестів, які визначають необхідні покращення або нові функції. Кожна ітерація має на меті розробити код, який пройде ці тести. Нарешті, програміст або група вдосконалюють код для погодження змін. Один із ключових моментів TDD полягає у тому, що підготовка тестів перед написанням самого коду пришвидшує процес внесення змін. Варто зауважити, що керована тестами розробка є методологією розробки програмного забезпечення, а не його тестування.

Test-Driven Development відноситься до концепції екстремального програмування, яка стверджує, що спершу потрібно писати тести, а вже потім код, яка веде свій початок з 1999 року, однак, останнім часом спостерігається загальніший інтерес до даної методології. Програмісти також використовують дану методологію для вдосконалення і зневадження сирцевого коду, раніше написаного з використанням інших методологій розробки.

Три закони TDD:

- Не можна писати жодного вихідного коду, доки спершу не написано падаючого юніт-тесту (англ. unit test).
- Не можна писати більше юніт-тесту ніж необхідно для падіння (непроходження тесту). Помилка компіляції - це також падіння (англ. failing).
- Не можна писати більше вихідного коду, ніж необхідно для проходження впалого юніт-тесту.

JUnit — бібліотека для тестування програмного забезпечення для мови Java. Створений Кентом Беком і Еріком Гаммою, JUnit є представником родини фреймворків xUnit для різних мов програмування, яка бере початок у SUnit Кента Бека для Smalltalk. JUnit породив екосистему розширень — JMock, EasyMock, DbUnit, HttpUnit, Selenium тощо. Досвід одержаний при роботі з JUnit був важливим у розробці концепцій тестування програмного забезпечення.

Функціональність JUnit: junit.framework.Assert: assertEquals, assertFalse, assertNotNull, assertNull, assertNotSame, assertSame, assertTrue; junit.framework.TestCase extends junit.framework.Assert: run, setUp, tearDown.

Хід роботи

Створимо тестові методи для класів-репозиторіїв(класів-доступу до даних).

Клас-репозиторій: UserRepository -> Клас тест: UserRepositoryTest:

```
class UserRepositoryTest {

   private final String email = "test@gmail.com";
   private final String password = "password";

   private User getTestUserData() {
      User inputUser = new User();
      inputUser.setEmail(email);
      inputUser.setName("testName");
      inputUser.setSurname("testSurname");
      inputUser.setPassword(password);
      return inputUser;
   }

   @Test
   void testSaveUser() {
      User inputUser = getTestUserData();
      UserRepository.saveUser(inputUser);
      List<User> expectedUsers = UserRepository.getLoginedUser(email, password);
      assertEquals(1, expectedUsers.size());
      User expectedUser = expectedUsers.get(0);
      assertEquals(expectedUser.getEmail(), inputUser.getEmail());
      assertEquals(expectedUser.getSurname(), inputUser.getSurname());
      assertEquals(expectedUser.getName(), inputUser.getName());
      UserRepository.deleteUser(expectedUser.getId());
   }

   @Test
   void testDeleteUser() {
      User inputUser = getTestUserData();
      UserRepository.saveUser(inputUser);
      User expectedUser = UserRepository.getLoginedUser(email, password).get(0);
      Integer id = expectedUser.getId();
      UserRepository.deleteUser(id);
      List<User> expectedUsers = UserRepository.getLoginedUser(email, password);
      assertEquals(0, expectedUsers.size());
   }

   @Test
   void testGetAllUsers() {
      int count = UserRepository.getUsers().size();
      User inputUser = getTestUserData();
      UserRepository.saveUser(inputUser);
      User secondUser = new User();
```

```
        secondUser.setEmail(email+"2");
        secondUser.setName("testName2");
        secondUser.setSurname("testSurname2");
        secondUser.setPassword(password+"2");
        UserRepository.saveUser(secondUser);
        List<User> expectedUsers = UserRepository.getUsers();
        assertEquals(count+2, expectedUsers.size());
        assertEquals(email, expectedUsers.get(0+count).getEmail());
        assertEquals(password, expectedUsers.get(0+count).getPassword());
        assertEquals(email+"2", expectedUsers.get(1+count).getEmail());
        assertEquals(password+"2", expectedUsers.get(1+count).getPassword());
        UserRepository.deleteUser(expectedUsers.get(0+count).getId());
        UserRepository.deleteUser(expectedUsers.get(1+count).getId());
    }

    @Test
    void testUpdateUser(){
        User inputUser = getTestUserData();
        UserRepository.saveUser(inputUser);
        User expectedUser = UserRepository.getLoginedUser(email, password).get(0);
        inputUser.setName("newName");
        inputUser.setPassword("newPassword");
        UserRepository.updateUser(expectedUser.getId(), inputUser);
        List<User> expectedUserAfterUpdate = UserRepository.getLoginedUser(email,
"newPassword");
        assertEquals(1, expectedUserAfterUpdate.size());
        assertEquals("newName", expectedUserAfterUpdate.get(0).getName());
        assertEquals(inputUser.getSurname(), expectedUserAfterUpdate.get(0).getSurname());
        UserRepository.deleteUser(expectedUserAfterUpdate.get(0).getId());
    }

}
```

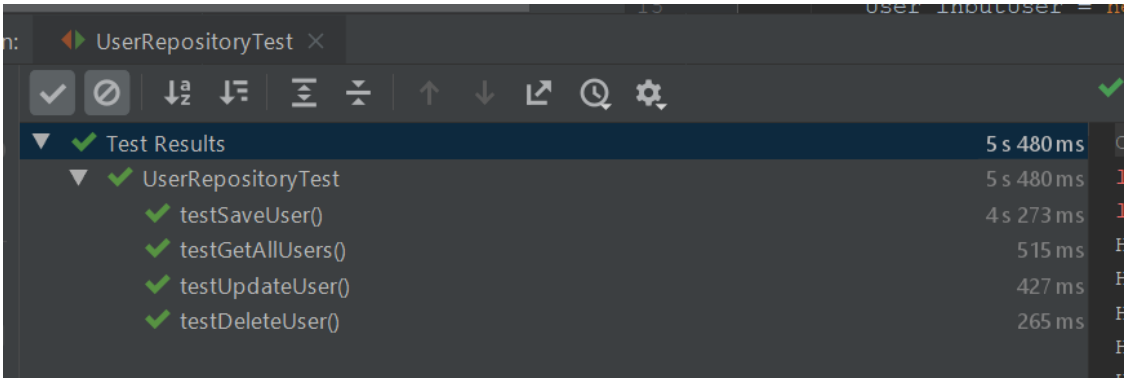

Рис.5.1 Успішне виконання тестів для UserRepository.java

Клас-репозиторій: CategoryRepository -> Клас тест: CategoryRepositoryTest:

```
class CategoryRepositoryTest {

    private final String categoryName = "testCategory";
```

```
private Category getTestCategoryData() {
    Category category = new Category();
    category.setName(categoryName);
    return category;
}

@Test
void saveCategory(){
    Category inputCategory = getTestCategoryData();
    CategoryRepository.saveCategory(inputCategory);
    Category expected = CategoryRepository.getCategoryByName(categoryName);
    assertNotEquals(null, expected);
    assertEquals(categoryName, expected.getName());
    CategoryRepository.deleteCategory(categoryName);
}

@Test
void testDeleteCategory(){
    Category inputCategory = getTestCategoryData();
    CategoryRepository.saveCategory(inputCategory);
    Category expected = CategoryRepository.getCategoryByName(categoryName);
    assertNotEquals(null, expected);
    assertEquals(categoryName, expected.getName());
    CategoryRepository.deleteCategory(categoryName);
    try {
        CategoryRepository.getCategoryByName(categoryName);
    }catch (Exception e) {
        assertEquals(IndexOutOfBoundsException.class, e.getClass());
    }
}

@Test
void testUpdateCategory(){
    Category inputCategory = getTestCategoryData();
    CategoryRepository.saveCategory(inputCategory);
    Category expected = CategoryRepository.getCategoryByName(categoryName);
    final String updatedName = "updatedName";
    expected.setName(updatedName);
    CategoryRepository.updateCategory(expected.getId(), updatedName);
    Category expectedAfterUpdate = CategoryRepository.getCategoryByName(updatedName);
    assertNotNull(expectedAfterUpdate);
    assertEquals(expectedAfterUpdate.getId(), expected.getId());
    CategoryRepository.deleteCategory(updatedName);
}

@Test
void testGetAllCategories(){
    int count = CategoryRepository.getCategories().size();
    Category inputCategory = getTestCategoryData();
    CategoryRepository.saveCategory(inputCategory);
    Category anotherCategory = new Category();
    anotherCategory.setName("anotherCategory");
```

```
        CategoryRepository.saveCategory(anotherCategory);
        List<Category> categories = CategoryRepository.getCategories();
        assertEquals(count+2, categories.size());
        assertEquals(categories.get(count).getName(), inputCategory.getName());
        CategoryRepository.deleteCategory(categoryName);
        CategoryRepository.deleteCategory(anotherCategory.getName());
    }
}
```

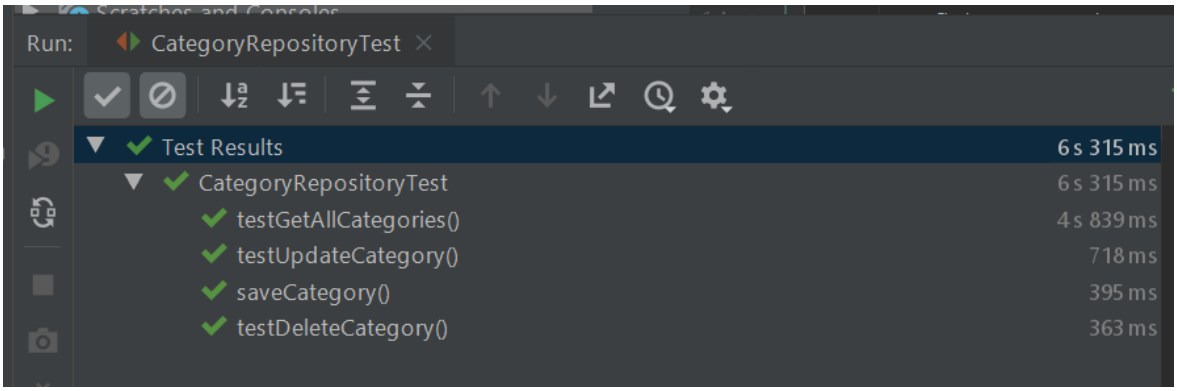

Рис.5.2 Успішне виконання тестів для CategoryRepository.java

Висновок: в результаті лабораторної роботи було покрито тестами розроблений проект за допомогою бібліотеки модульного тестування JUnit.