**Лабораторна робота №2**

Вивчення принципів модульного тестування з використанням бібліотеки Junit

**2.1 Мета роботи**

Навчитися розробляти Unit-тести з використанням інструментальної бібліотеки JUnit. Розробити тести для одного зі стандартних класів JDK.

**2.2 Теоретичні відомості**

Тестування програмного забезпечення – процес виявлення помилок в програмному забезпеченні. Виділяють види тестування згідно таких критеріїв:

­ ступень ізольованості компонентів: модульне, інтеграційне, системне тестування;

­ об’єкт тестування: функціональне, тестування навантаження, тестування інтерфейсу; тестування безпеки;

­ наявність інформації про структуру та функціонування системи:

тестування «чорного», «білого» ящику;

­ час проведення: альфа-, бета-тестування, приймальне тестування,

регрес тестування;

­ ступінь автоматизації: ручне, автоматизоване, напів автоматизоване;

­ виконання кода: статичне та динамічне тестування.

Кінцевою метою будь-якого процесу тестування є забезпечення якості програмного забезпечення. При цьому жоден вид тестування не надає 100% гарантії працездатності програми. Планування тестових сценаріїв має бути виконано таким чином, щоб перевірялися різні варіанти використання програми.

Модульне тестування передбачає перевірку працездатності найменших складових програми: зазвичай це окремий клас і його методи. У випадках, якщо при виконанні методів тестованого класу відбувається виклик методів стороннього класу, необхідно відокремити тестовані класи один від одного, замінити їх за допомогою mock-об'єктів, заглушок, імітацій. Наприклад, якщо тестується метод класу, який приймає ім'я і пароль користувача, а потім звертається до бази даних для перевірки правильності введених користувачем даних аутентифікації (інший клас), то потрібно замінити клас, який працює з реальними даними «імітацією», яка не виконує звернення до бази даних, а зберігає дані користувачів всередині самого класу. Імітація повинна бути досить

проста, щоб не містити помилок - в іншому випадку говорити про модульному тестуванні не можна.

JUnit - популярний в даний час інструмент модульного тестування java-додатків. JUnit 4 заснований на використанні анотацій.

**Хід роботи**

Для реалізація завдання була вибрана група 4 та варіант 14 за списком, потрібний клас для тестування java.lang.Short. Вихідний код классу тестування зображено в лістингу 2.1. Для створення методів використовувалась анотація @Test, для перевірки результату методи assertTrue, assertEquals. Результат виконання тестів зображено на рисунку 2.1.

Лістинг 2.1

package lab;

import org.junit.Test;

import static org.junit.Assert.\*;

public class ShortTest {

@Test

public void testShortParse() {

assertEquals(32767, Short.parseShort("32767"));

}

@Test

public void testShortValue() {

Short s = new Short("2345");

assertEquals(2345, s.shortValue());

}

@Test

public void testToString() {

Short s = new Short("-1234");

assertEquals("-1234", s.toString());

}

@Test

public void testMaxValue() {

assertEquals(32767, Short.MAX\_VALUE);

}

@Test

public void testMinValue() {

assertEquals(-32768, Short.MIN\_VALUE);

}

@Test

public void testCompare() {

Short s1 = new Short("10");

Short s2 = new Short("20");

assertTrue(s1.compareTo(s2) < 0);

}

@Test

public void testEquals() {

Short s1 = new Short("55");

Short s2 = new Short("55");

assertTrue(s1.equals(s2));

}

@Test

public void testHashCode() {

Short s = new Short("987");

assertEquals(987, s.hashCode());

}

@Test(expected = NumberFormatException.class)

public void testNumberFormatException() {

Short.parseShort("abcde");

}

@Test

public void testDecode() {

String dec = "12";

String hex = "0x0C";

String oct = "014";

assertEquals(12, Short.decode(dec).shortValue());

assertEquals(12, Short.decode(hex).shortValue());

assertEquals(12, Short.decode(oct).shortValue());

}

}

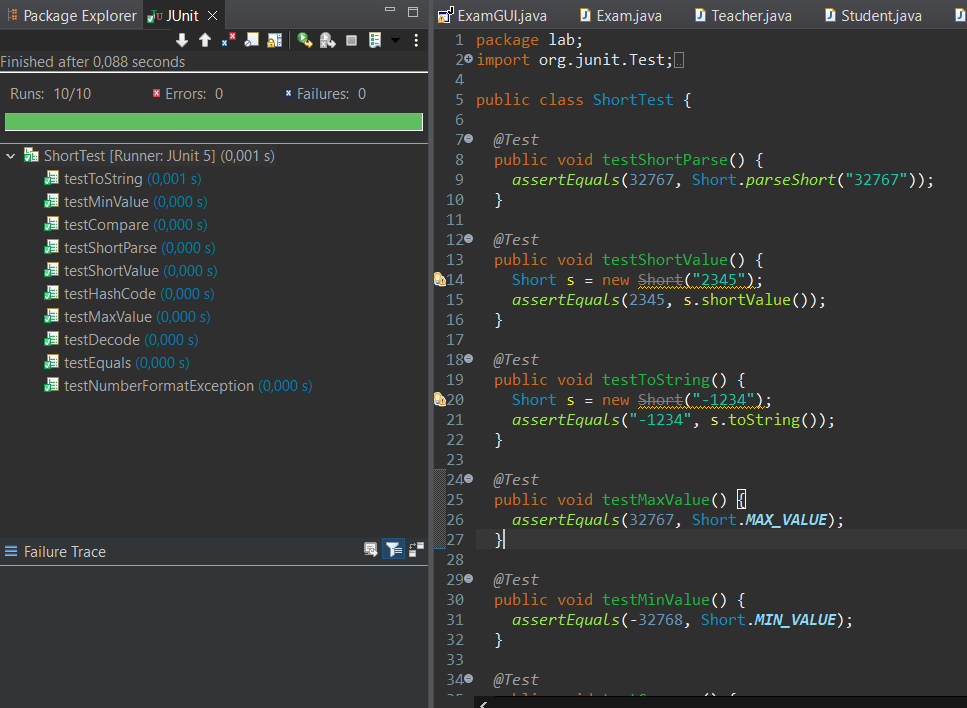


Рисунок 2.1 – Результат проходження тестів

Висновок: на даній лабораторній роботі я навчився розробляти Unit-тести з використанням інструментальної бібліотеки JUnit. Розробити тести для стандартного класу String, та його основних методів.