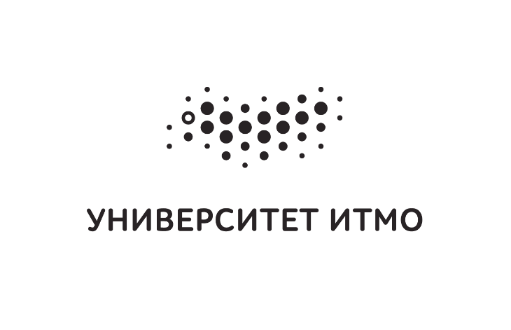
Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Оптики, кафедра Информатики и прикладной математики



Лабораторная работа №1

по дисциплине

Тестирование программного обеспечения

Вариант 2

Выполнила: Жетесова Дана Айбековна

Принимающий: Исаев Илья Владимирович

Санкт-Петербург

2020

Оглавление

[Цель 2](#_Toc35431199)

[Описание JUnit 5 2](#_Toc35431200)

[Требования 3](#_Toc35431201)

[Код с комментариями 4](#_Toc35431202)

[Найденные баги 4](#_Toc35431203)

[Выводы и заключение 4](#_Toc35431204)

# Цель

Изучить JUnit 5, использовать полученные знания на практике, а именно: реализовать консольное приложение по варианту по сформулированным мной требованиям и прокрыть его модульными тестами на 70%. Посмотреть, какие баги будут найдены и определить для себя важность модульного тестирования и какие проблемы оно решает или помогает избегать, насколько оно полезно и как влияет на мою производительность.

# Описание JUnit 5

JUnit 5 состоит из нескольких разных модулей из трех разных подпроектов: П**латформа JUnit**, **Юнит Юпитер** и **Юнит Винтаж**.

Платформа отвечает за запуск сред тестирования на JVM. Он определяет стабильный и мощный интерфейс между JUnit и его клиентом, таким как инструменты сборки. Конечная цель - как легко интегрировать своих клиентов с JUnit в обнаружение и выполнение тестов. Он также определяет [TestEngine](http://junit.org/junit5/docs/current/api/org/junit/platform/engine/TestEngine.html)API для разработки инфраструктуры тестирования, работающей на платформе JUnit (IntelliJ IDEA, Eclipse, NetBeans, and Visual Studio Code). Таким образом, вы можете подключить сторонние библиотеки тестирования непосредственно в JUnit, реализовав собственный TestEngine.

JUnit Jupiter - расширение для написания тестов и расширений в JUnit 5.

JUnit Vintage предоставляет платформу для запуска тестов на основе JUnit 3 и JUnit 4.

Основные аннотации

* @BeforeAll **и**@BeforeEach (перед всеми сразу или перед каждым по отдельности тестами в классе; важно отметить, что метод с аннотацией @BeforeAll должен быть статическим, иначе код не будет компилироваться)
* @DisplayName **и**@Disabled (изменение отображаемого имени и отключение с комментарием)
* @AfterEach **и**@AfterAll (выполнение после каждого теста в классе или после всех)

В JUnit 5 есть два способа тестирования исключений. Оба они могут быть реализованы с помощью метода assertThrows().

@Test

void shouldThrowException() {

Throwable exception = assertThrows(UnsupportedOperationException.class, () -> {

throw new UnsupportedOperationException("Not supported");

});

assertEquals(exception.getMessage(), "Not supported");

}

@Test

void assertThrowsException() {

String str = null;

assertThrows(IllegalArgumentException.class, () -> {

Integer.valueOf(str);

});

}

Первый пример используется для проверки более подробной информации о сгенерированном исключении, а второй только проверяет тип исключения.

@SelectPackages и @SelectClasses – аннотации для создания тестовых наборов (список пакетов или список классов).

@TestFactory – для создания динамических тестов.

# Требования

Веб-скрапер. Создайте инструмент, принимающий URL в качестве input и возвращающий контент в виде HTML или XML.

Функциональные требования:

1. Система должна представлять собой консольное приложение и запускаться из консоли.
2. Системе при запуске должны передаваться входные параметры: строка URL и формат вывода (-f в файл, -c в консоль, без параметра тоже в консоль) и в случае, если передан параметр -f, можно передать параметр названия выходного файла. По параметру -h будет выдаваться справка.  
   Входные параметры определены в таблице 1.  
   Таблица 1.

|  |  |  |
| --- | --- | --- |
| Параметр 1 | Параметр 2 (опционально) | Параметр 3 (опционально) |
| -h | Ничего | Ничего |
| Строка URL скачиваемого ресурса | Ничего | Ничего |
| -с | Ничего |
| -f | Ничего |
| Строка название выходного файла |

1. Система должна проверять корректность входных данных.
2. В случае неправильного ввода входных параметров система должна вывести подробное описание ошибки в консоль согласно таблице 2.  
   Таблица 2.

|  |  |
| --- | --- |
| Случай с ошибкой | Текст исключения |
| Введено 0 параметров | Please provide URL of downloading resource. For help use command -h. |
| Введено более 3х параметров | You can’t input more than 3 parameters! For help use command -h. |
| Введён только 2ой или 2ой и 3ий параметры без URL | Please provide URL of downloading resource. For help use command -h. |
| URL введён, второй параметр не соответствует -с или -f | Second parameter have to me -c or -f! For help use command -h. |
| URL введён, второй параметр -c и есть третий параметр | There’s no third argument using command -c! For help use command -h. |
| После команды -h введены ещё какие-то параметры. | -h command has no arguments! For help use command -h. |
| URL не существует | This URL address doesn’t exist! |
| URL введён правильно, далее следует команда -f и неправильное название выходного файла | Invalid filename! |

1. Система должна выводить справку по команде -h.
2. Система должна скачивать данные по переданному ей URL.
3. Система должна выводить данные в консоль, если не использованы никакие вспомогательные команды или по команде -с.
4. Система должна сохранять скаченные данные в файл по команде -f.
5. В случае, если не приведено конкретное название файла, система должна сохранить скаченные данные в файл, название которого будет равно URL.
6. В случае, если системе дано конкретное название файла, система должна сохранить скаченные данные в файл с таким названием.
7. Если файла с конкретным переданным системе названием не существует, система должна создать такой файл и сохранить в него скаченные данные.
8. Система должна сохранять данные в формате HTML/XML, в зависимости от содержимого страницы по переданной URL.
9. В случае, если на странице данные не в формате HTML/XML, система сохраняет их в формате HTML.
10. Система должна

# Код с комментариями

# Найденные баги

# Выводы и заключение