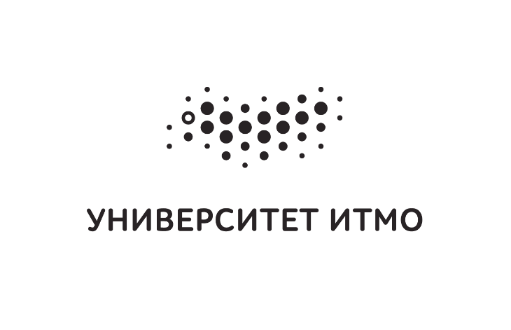
Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

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](#_Toc35651007)

[Описание JUnit 5 2](#_Toc35651008)

[Требования 3](#_Toc35651009)

[Код с комментариями 5](#_Toc35651010)

[Найденные баги 25](#_Toc35651011)

[Выводы и заключение 26](#_Toc35651012)

# 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 должны передаваться входные параметры, определенные в таблице 1.  
     
   Таблица 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр 1 | Параметр 2 (опционально) | Параметр 3 (опционально) | Параметр 4 (опционально) | Параметр 5 (опционально) |
| -h | Ничего | Ничего | Ничего | Ничего |
| Строка URL скачиваемого ресурса | Ничего / -text / -body / -head / -all | Ничего | Ничего | Ничего |
| -с | Ничего / -text / -body / -head / -all | Ничего | Ничего |
| -f | Ничего / -text / -body / -head / -all | Ничего / -txt / -html / -xml | Ничего / Название выходного файла |

1. Система должна проверять корректность входных данных.
2. Система должна проверять входные данные на правильность их порядка.
3. 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 5ти параметров | You can’t input more than 5 parameters! 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! |
| Первый аргумент не URL и не -h или URL введён неверно | Invalid URL! Please, write URL correctly. For help use command -h. |
| Первый аргумент URL, второй -c, третий – не аргумент скачиваемой части контента по URL | After -c output type you can write only scrapping document part argument! (-text / -body / -head / -all) For help use command -h. |
| Первый аргумент URL, второй не тип вывода данных и не аргумент скачиваемой части контента по URL | You must write output type command or part of scrapping page after URL! For help use command -h. |
| Первый аргумент URL, второй не тип вывода данных и дальше ещё идут аргумент(ы) | You can write other args only after type command (-c or -f)! For help use command -h. |
| Первый аргумент URL, второй -f, третий не аргумент скачиваемой части контента по URL и не формат сохраняемого файла | After -f must be part of scrapping page or file type argument before filename! For help use command -h. |
| Нарушен порядок в цепочке из пяти входных параметров (первый аргумент URL, второй -f, дальше – путаница) | You must write arguments in order: url, -f, part of scrapping page argument, file type argument, filename! For help use command -h. |

1. В случае возникновения непредвиденных исключений при получении данных по URL, система должна вывести стектрейс.
2. В случае возникновения непредвиденных исключений при сохранении данных в файл, система должна вывести стектрейс.
3. Система должна выводить справку по команде -h.
4. Справка должна выглядеть так: «».
5. Система должна скачивать данные по переданному ей URL.
6. Система должна скачивать определённые данные со страницы в соответствии с переданным аргументом типа данных со страницы.
7. В случае, если системе такой аргумент не передан, система должна скачивать весь html контент по URL.
8. Если системе был передан аргумент -all, она должна скачать весь html контент страницы.
9. Если системе был передан аргумент -head, она должна скачать head html контента страницы.
10. Если системе был передан аргумент -body, она должна скачать весь body html контента страницы.
11. Если системе был передан аргумент -text, она должна скачать весь текст страницы без html тегов, преобразуя страницу из формата html в текстовый.
12. Система должна выводить данные в консоль по команде -с.
13. Система должна сохранять скаченные данные в файл по команде -f.
14. В случае если не использованы никакие вспомогательные команды, система должна выводить данные в консоль.
15. В случае, если не приведено конкретное название файла, система должна сохранить скаченные данные в файл, название которого будет равно URL.
16. В случае, если системе дано конкретное название файла, система должна сохранить скаченные данные в файл с таким названием.
17. Если файла с конкретным переданным системе названием не существует, система должна создать такой файл и сохранить в него скаченные данные.
18. В случае, если файл с введённым пользователем названием уже существует и заполнен, система должна записать скаченные данные в него поверх ранее существовавших данных.
19. В случае, если файл с названием, соответствующим URL, уже существует и заполнен, система должна записать скаченные данные в него поверх ранее существовавших данных.
20. Система должна проверять валидность URL.
21. Система должна сохранять данные в формате в зависимости от переданного аргумента типа сохраняемого файла.
22. Система должна дополнить название файла или URL до указания формата файла в соответствии с типом сохраняемого файла.
23. В случае, если системе был передан аргумент -txt, система должна сохранить файл в формате TXT.
24. В случае, если системе был передан аргумент -xml, система должна сохранить файл в формате XML.
25. В случае, если системе был передан аргумент -html, система должна сохранить файл в формате HTML.
26. В случае, если аргумента нет, система должна сохранить данные в файл формата TXT.

# Код с комментариями

Pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>SoftwareTesting</groupId>  
 <artifactId>SoftwareTesting</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <configuration>  
 <source>10</source>  
 <target>10</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
 <dependencies>  
 <!-- https://mvnrepository.com/artifact/org.jsoup/jsoup -->  
 <dependency>  
 <groupId>org.jsoup</groupId>  
 <artifactId>jsoup</artifactId>  
 <version>1.13.1</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>5.5.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.junit.platform</groupId>  
 <artifactId>junit-platform-runner</artifactId>  
 <version>1.5.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>1.18.6</version>  
 </dependency>  
  
 </dependencies>  
  
  
</project>

package model.enums;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
public enum DocumentPartTypes {  
 *ALL*("-all"),  
 *HEAD*("-head"),  
 *BODY*("-body"),  
 *TEXT*("-text");  
  
 private String title;  
  
 DocumentPartTypes(String title) {  
 this.title = title;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public static ArrayList<String> getAll() {  
 return new ArrayList<>(Arrays.*asList*("-all", "-head", "-body", "-text"));  
 }  
  
 public static DocumentPartTypes getByTitle(String title) {  
 if (title == null) return *ALL*;  
 switch (title) {  
 case "-head": return *HEAD*;  
 case "-body": return *BODY*;  
 case "-text": return *TEXT*;  
 case "-all":  
 case "":  
 return *ALL*;  
 default: return null;  
 }  
 }  
}

package model.enums;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
public enum OutputFileTypes {  
 *TXT*("-txt"),  
 *HTML*("-html"),  
 *XML*("-xml");  
  
 private String title;  
  
 OutputFileTypes(String title) {  
 this.title = title;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public String getType() {  
 return "." + title.substring(1, title.length());  
 }  
  
 public static ArrayList<String> getAll() {  
 return new ArrayList<>(Arrays.*asList*("-txt", "-html", "-xml"));  
 }  
  
 public static OutputFileTypes getByTitle(String title) {  
 if (title == null) return *TXT*;  
 switch (title) {  
 case "-html": return *HTML*;  
 case "-xml": return *XML*;  
 case "-txt":  
 case "":  
 return *TXT*;  
 default: return null;  
 }  
 }  
}

package model.enums;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
public enum OutputTypes {  
 *FILE*("-f"),  
 *CONSOLE*("-c");  
  
 private String title;  
  
 OutputTypes(String title) {  
 this.title = title;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public static ArrayList<String> getAll() {  
 return new ArrayList<>(Arrays.*asList*("-f", "-c"));  
 }  
  
 public static OutputTypes getByTitle(String title) {  
 if (title == null) return *CONSOLE*;  
 switch (title) {  
 case "-f": return *FILE*;  
 case "-c":  
 case "":  
 return *CONSOLE*;  
 default: return null;  
 }  
 }  
}

package model;  
  
import lombok.Data;  
  
@Data  
public class ScrappedPage {  
  
 private final String content;  
  
 private final String filename;  
  
 public ScrappedPage(String content) {  
 this.content = content;  
 this.filename = null;  
 }  
  
 public ScrappedPage(String content, String filename) {  
 this.content = content;  
 this.filename = filename;  
 }  
}

package model;  
  
import lombok.Data;  
import model.enums.DocumentPartTypes;  
import model.enums.OutputFileTypes;  
import model.enums.OutputTypes;  
  
@Data  
public class WebScrapperState {  
  
 private final DocumentPartTypes documentPartType;  
  
 private final OutputTypes outputType;  
  
 private final OutputFileTypes outputFileType;  
  
 private final String filename;  
  
 private final String url;  
}

package services;  
  
import java.util.List;  
import model.WebScrapperState;  
import model.enums.DocumentPartTypes;  
import model.enums.OutputFileTypes;  
import model.enums.OutputTypes;  
  
import java.util.regex.Pattern;  
import java.util.stream.Collectors;  
  
public class InputArgsService {  
  
 public static WebScrapperState parse(String[] args) {  
 DocumentPartTypes documentPartType = DocumentPartTypes.*getByTitle*(  
 *nullSafeGet*(  
 List.*of*(args).stream()  
 .filter(x -> DocumentPartTypes.*getAll*().contains(x))  
 .collect(Collectors.*toList*()), 0));  
  
 OutputFileTypes outputFileType = OutputFileTypes.*getByTitle*(  
 *nullSafeGet*(  
 List.*of*(args).stream()  
 .filter(x -> OutputFileTypes.*getAll*().contains(x))  
 .collect(Collectors.*toList*()), 0));  
  
 OutputTypes outputType = OutputTypes.*getByTitle*(  
 *nullSafeGet*(  
 List.*of*(args).stream()  
 .filter(x -> OutputTypes.*getAll*().contains(x))  
 .collect(Collectors.*toList*()), 0));  
  
 String url = *parseUrl*(args[0]);  
 String filename;  
  
 switch (args.length) {  
  
 case 1:  
 filename = null;  
 break;  
  
 case 2:  
 case 3:  
 case 4:  
 case 5:  
 if (OutputTypes.*getAll*().contains(args[1]) || DocumentPartTypes.*getAll*().contains(args[1])) { // -c / -f / -text / -body / -head / -all  
 if (args.length == 2) {  
 filename = null;  
 } else {  
 if (OutputTypes.*getAll*().contains(args[1])) { // -c / -f  
 if (args[1].equals(OutputTypes.*CONSOLE*.getTitle())) { //CONSOLE -c  
 if (DocumentPartTypes.*getAll*().contains(args[2])) { // -text / -body / -head / -all  
 filename = null;  
 } else {  
 throw new IllegalArgumentException("After -c output type you can write only "  
 + "scrapping document part argument! (-text / -body / -head / -all) "  
 + "For help use command -h.");  
 }  
  
 } else { //FILE -f  
 if (args.length == 3) { // (-text / -body / -head / -all) OR (-txt / -html / -xml)  
 if (DocumentPartTypes.*getAll*().contains(args[2]) || OutputFileTypes.*getAll*().contains(args[2])) {  
 filename = null;  
 } else {  
 filename = args[2]; // filename  
 }  
  
 } else if (args.length == 4) {  
 if (DocumentPartTypes.*getAll*().contains(args[2])) { // -text / -body / -head / -all  
 if (OutputFileTypes.*getAll*().contains(args[3])) {  
 filename = null; // -txt / -html / -xml  
 } else {  
 filename = args[3]; // filename  
 }  
 } else if (OutputFileTypes.*getAll*().contains(args[2])) { // -txt / -html / -xml  
 filename = args[3]; // filename  
 } else {  
 throw new IllegalArgumentException("After -f must be part of scrapping page "  
 + "or file type argument before filename! For help use command -h.");  
 }  
  
 } else { // args.length == 5  
 if (DocumentPartTypes.*getAll*().contains(args[2])) { // -text / -body / -head / -all  
 if (OutputFileTypes.*getAll*().contains(args[3])) { // -txt / -html / -xml  
 filename = args[4]; // filename  
 } else {  
 throw new IllegalArgumentException("You must write arguments in order: "  
 + "url, -f, part of scrapping page argument, file type argument, " +  
 "filename! For help use command -h.");  
 }  
 } else {  
 throw new IllegalArgumentException("You must write arguments in order: "  
 + "url, -f, part of scrapping page argument, file type argument, " +  
 "filename! For help use command -h.");  
 }  
 }  
 }  
  
 } else {  
 throw new IllegalArgumentException("You can write other args only after "  
 + "output type command (-c or -f)! For help use command -h.");  
 }  
 }  
 } else {  
 throw new IllegalArgumentException("You must write output type command or part of scrapping page "  
 + "after URL! For help use command -h.");  
 }  
 break;  
  
 default:  
 throw new IllegalArgumentException("You can’t input more than 5 parameters! For help use command -h.");  
 }  
  
 if (outputType.equals(OutputTypes.*FILE*)) {  
 filename = *parseFilename*(filename, url, outputFileType);  
 }  
  
 return new WebScrapperState(documentPartType, outputType, outputFileType, filename, url);  
 }  
  
 private static String parseUrl(String url) {  
 String regex = "^(https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~\_|!:,.;]\*[-a-zA-Z0-9+&@#/%=~\_|]";  
 if (Pattern.*matches*(regex, url)) {  
 return url;  
 } else {  
 throw new IllegalArgumentException("Invalid URL! Please, write URL correctly. For help use command -h.");  
 }  
 }  
  
 private static String parseFilename(String filename, String url, OutputFileTypes outputFileType) {  
 if (filename == null) {  
 filename = url.replace("\\", "").replace("/", "");  
 }  
 String regex = ".\*\\." + outputFileType.getType().replace(".", "") + "$";  
 if (Pattern.*matches*(regex, filename)) {  
 return filename;  
 } else {  
 return filename + outputFileType.getType();  
 }  
 }  
  
 private static String nullSafeGet(List<String> list, int i) {  
 if (list.isEmpty()) return null;  
 return list.get(0);  
 }  
}

package services;  
  
import model.ScrappedPage;  
import model.WebScrapperState;  
import model.enums.OutputTypes;  
  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class OutputService {  
  
 public static void Output(WebScrapperState state, ScrappedPage page) {  
 if (state.getOutputType().equals(OutputTypes.*FILE*)) {  
 *FileOutput*(page);  
 } else {  
 *ConsoleOutput*(page);  
 }  
 }  
  
 private static void FileOutput(ScrappedPage page) {  
 try {  
 FileWriter myWriter = new FileWriter(page.getFilename());  
 myWriter.write(page.getContent());  
 myWriter.close();  
 System.*out*.println("Successfully wrote to the file: " + page.getFilename());  
 } catch (IOException e) {  
 System.*out*.println("An error occurred.");  
 e.printStackTrace();  
 }  
 }  
  
 private static void ConsoleOutput(ScrappedPage page) {  
 System.*out*.println("\nContent: \n" + page.getContent());  
 }  
  
 public static void ConsoleHelp() {  
 System.*out*.println("\nUsage of Dana's web scraping program:\n" +  
 "\n" +  
 "[-h]\n" +  
 "[<URL> (-text / -body / -head / -all)]\n" +  
 "[<URL> -c (-text / -body / -head / -all)]\n" +  
 "[<URL> -f (-text / -body / -head / -all) (-txt / -html / -xml) (<output file name>)]\n" +  
 "\n" +  
 "-h - call this info\n" +  
 "<URL> - download content from resource by this URL\n" +  
 "\n" +  
 " (Output type commands)\n" +  
 "-c - output in console (by default)\n" +  
 "-f - save downloaded content to file with filename equals to URL\n" +  
 "\n" +  
 " (Part type of downloading content arguments)\n" +  
 "-text - downloading text data without html tags\n" +  
 "-body - downloading html data from body\n" +  
 "-head - downloading html data from head\n" +  
 "-all - downloading whole html page\n" +  
 "\n" +  
 " (Output file type arguments)\n" +  
 "-txt - saving data in .txt file\n" +  
 "-html - saving data in .html file\n" +  
 "-xml - saving data in .xml file\n" +  
 "\n" +  
 "<output file name> - save downloaded data to file with this name\n" +  
 "\n" +  
 "REMEMBER! The order is necessary!\n");  
 }  
}

package services;  
  
import model.ScrappedPage;  
import model.WebScrapperState;  
import org.jsoup.Jsoup;  
import org.jsoup.nodes.Document;  
  
import java.io.IOException;  
  
public class WebScrapingService {  
  
 public static ScrappedPage scrap(WebScrapperState state) {  
 try {  
  
 Document doc = Jsoup.*connect*(state.getUrl()).get();  
  
 String content = "";  
 switch (state.getDocumentPartType()) {  
  
 case *ALL*:  
 content = doc.html();  
 break;  
  
 case *BODY*:  
 content = doc.body().html();  
 break;  
  
 case *HEAD*:  
 content = doc.head().html();  
 break;  
  
 case *TEXT*:  
 content = doc.text();  
 break;  
  
 }  
  
 if (state.getFilename() == null) {  
 return new ScrappedPage(content);  
 } else {  
 return new ScrappedPage(content, state.getFilename());  
 }  
  
 } catch (IOException e) {  
 System.*out*.println("An error occurred.");  
 e.printStackTrace();  
 throw new IllegalArgumentException("This URL address doesn’t exist!");  
 }  
 }  
}

import model.ScrappedPage;  
import model.WebScrapperState;  
import services.InputArgsService;  
import services.OutputService;  
import services.WebScrapingService;  
  
public class Main {  
 public static void main(String[] args) {  
  
 switch (args.length) {  
  
 case 0: throw new IllegalArgumentException("Please provide URL of downloading resource. " +  
 "For help use command -h.");  
  
 case 1: {  
 if (args[0].equals("-h")) {  
 OutputService.*ConsoleHelp*();  
 } else {  
 *callScrapper*(args);  
 }  
 break;  
 }  
  
 case 2:  
 case 3:  
 case 4:  
 case 5:{  
 if (args[0].equals("-h")) {  
 throw new IllegalArgumentException("-h command has no arguments! For help use command -h.");  
 } else {  
 *callScrapper*(args);  
 }  
 break;  
 }  
  
 default:  
 throw new IllegalArgumentException("You can’t input more than 5 parameters! For help use command -h.");  
  
 }  
 }  
  
 private static void callScrapper(String[] args) {  
 WebScrapperState state = InputArgsService.*parse*(args);  
 ScrappedPage page = WebScrapingService.*scrap*(state);  
 OutputService.*Output*(state, page);  
 }  
}

package tests.modelTests.enumsTests;  
  
import model.enums.DocumentPartTypes;  
import org.junit.jupiter.api.Test;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
  
public class DocumentPartTypesTests {  
 @Test  
 public void documentPartTypes\_All\_Test() {  
 *assertEquals*(DocumentPartTypes.*ALL*, DocumentPartTypes.*valueOf*("ALL"));  
 }  
  
 @Test  
 public void documentPartTypes\_Head\_Test() {  
 *assertEquals*(DocumentPartTypes.*HEAD*, DocumentPartTypes.*valueOf*("HEAD"));  
 }  
  
 @Test  
 public void documentPartTypes\_Body\_Test() {  
 *assertEquals*(DocumentPartTypes.*BODY*, DocumentPartTypes.*valueOf*("BODY"));  
 }  
  
 @Test  
 public void documentPartTypes\_Text\_Test() {  
 *assertEquals*(DocumentPartTypes.*TEXT*, DocumentPartTypes.*valueOf*("TEXT"));  
 }  
  
 @Test  
 public void getTitle\_All\_Test() {  
 *assertEquals*(DocumentPartTypes.*ALL*.getTitle(), "-all");  
 }  
  
 @Test  
 public void getTitle\_Head\_Test() {  
 *assertEquals*(DocumentPartTypes.*HEAD*.getTitle(), "-head");  
 }  
  
 @Test  
 public void getTitle\_Body\_Test() {  
 *assertEquals*(DocumentPartTypes.*BODY*.getTitle(), "-body");  
 }  
  
 @Test  
 public void getTitle\_Text\_Test() {  
 *assertEquals*(DocumentPartTypes.*TEXT*.getTitle(), "-text");  
 }  
  
 @Test  
 public void getByTitle\_All\_Test() {  
 *assertEquals*(DocumentPartTypes.*ALL*, DocumentPartTypes.*getByTitle*("-all"));  
 }  
  
 @Test  
 public void getByTitle\_All\_Null\_Test() {  
 *assertEquals*(DocumentPartTypes.*ALL*, DocumentPartTypes.*getByTitle*(null));  
 }  
  
 @Test  
 public void getByTitle\_All\_Empty\_Test() {  
 *assertEquals*(DocumentPartTypes.*ALL*, DocumentPartTypes.*getByTitle*(""));  
 }  
  
 @Test  
 public void getByTitle\_Head\_Test() {  
 *assertEquals*(DocumentPartTypes.*HEAD*, DocumentPartTypes.*getByTitle*("-head"));  
 }  
  
 @Test  
 public void getByTitle\_Body\_Test() {  
 *assertEquals*(DocumentPartTypes.*BODY*, DocumentPartTypes.*getByTitle*("-body"));  
 }  
  
 @Test  
 public void getByTitle\_Text\_Test() {  
 *assertEquals*(DocumentPartTypes.*TEXT*, DocumentPartTypes.*getByTitle*("-text"));  
 }  
  
 @Test  
 public void getAllTest() {  
 *assertEquals*(DocumentPartTypes.*getAll*(), new ArrayList<>(Arrays.*asList*("-all", "-head", "-body", "-text")));  
 }  
}

package tests.modelTests.enumsTests;  
  
import model.enums.OutputFileTypes;  
import org.junit.jupiter.api.Test;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class OutputFileTypesTests {  
 @Test  
 public void OutputFileTypes\_Txt\_Test() {  
 *assertEquals*(OutputFileTypes.*TXT*, OutputFileTypes.*valueOf*("TXT"));  
 }  
  
 @Test  
 public void OutputFileTypes\_Xml\_Test() {  
 *assertEquals*(OutputFileTypes.*XML*, OutputFileTypes.*valueOf*("XML"));  
 }  
  
 @Test  
 public void OutputFileTypes\_Html\_Test() {  
 *assertEquals*(OutputFileTypes.*HTML*, OutputFileTypes.*valueOf*("HTML"));  
 }  
  
 @Test  
 public void getTitle\_Txt\_Test() {  
 *assertEquals*(OutputFileTypes.*TXT*.getTitle(), "-txt");  
 }  
  
 @Test  
 public void getTitle\_Xml\_Test() {  
 *assertEquals*(OutputFileTypes.*XML*.getTitle(), "-xml");  
 }  
  
 @Test  
 public void getTitle\_Html\_Test() {  
 *assertEquals*(OutputFileTypes.*HTML*.getTitle(), "-html");  
 }  
  
 @Test  
 public void getByTitle\_Txt\_Test() {  
 *assertEquals*(OutputFileTypes.*TXT*, OutputFileTypes.*getByTitle*("-txt"));  
 }  
  
 @Test  
 public void getByTitle\_Txt\_Null\_Test() {  
 *assertEquals*(OutputFileTypes.*TXT*, OutputFileTypes.*getByTitle*(null));  
 }  
  
 @Test  
 public void getByTitle\_Txt\_Empty\_Test() {  
 *assertEquals*(OutputFileTypes.*TXT*, OutputFileTypes.*getByTitle*(""));  
 }  
  
 @Test  
 public void getByTitle\_Xml\_Test() {  
 *assertEquals*(OutputFileTypes.*XML*, OutputFileTypes.*getByTitle*("-xml"));  
 }  
  
 @Test  
 public void getByTitle\_Html\_Test() {  
 *assertEquals*(OutputFileTypes.*HTML*, OutputFileTypes.*getByTitle*("-html"));  
 }  
  
 @Test  
 public void getAllTest() {  
 *assertEquals*(OutputFileTypes.*getAll*(), new ArrayList<>(Arrays.*asList*("-txt", "-html", "-xml")));  
 }  
}

package tests.modelTests.enumsTests;  
  
import model.enums.OutputTypes;  
import org.junit.jupiter.api.Test;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class OutputTypesTypesTests {  
 @Test  
 public void OutputTypes\_File\_Test() {  
 *assertEquals*(OutputTypes.*FILE*, OutputTypes.*valueOf*("FILE"));  
 }  
  
 @Test  
 public void OutputTypes\_Console\_Test() {  
 *assertEquals*(OutputTypes.*CONSOLE*, OutputTypes.*valueOf*("CONSOLE"));  
 }  
  
 @Test  
 public void getTitle\_File\_Test() {  
 *assertEquals*(OutputTypes.*FILE*.getTitle(), "-f");  
 }  
  
 @Test  
 public void getTitle\_Console\_Test() {  
 *assertEquals*(OutputTypes.*CONSOLE*.getTitle(), "-c");  
 }  
  
 @Test  
 public void getByTitle\_Console\_Test() {  
 *assertEquals*(OutputTypes.*CONSOLE*, OutputTypes.*getByTitle*("-c"));  
 }  
  
 @Test  
 public void getByTitle\_Console\_Null\_Test() {  
 *assertEquals*(OutputTypes.*CONSOLE*, OutputTypes.*getByTitle*(null));  
 }  
  
 @Test  
 public void getByTitle\_Console\_Empty\_Test() {  
 *assertEquals*(OutputTypes.*CONSOLE*, OutputTypes.*getByTitle*(""));  
 }  
  
 @Test  
 public void getByTitle\_File\_Test() {  
 *assertEquals*(OutputTypes.*FILE*, OutputTypes.*getByTitle*("-f"));  
 }  
  
 @Test  
 public void getAllTest() {  
 *assertEquals*(OutputTypes.*getAll*(), new ArrayList<>(Arrays.*asList*("-f", "-c")));  
 }  
}

package tests.modelTests;  
  
import model.ScrappedPage;  
import org.junit.Test;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.junit.jupiter.api.Assertions.*assertNull*;  
  
public class ScrappedPageTests {  
 private static final ScrappedPage *scrapedPage* = new ScrappedPage("content", "filename.txt");  
  
 private static final ScrappedPage *emptyScrapedPage* = new ScrappedPage("");  
  
 @Test  
 public void getFilenameTest() {  
 *assertEquals*(*scrapedPage*.getFilename(), "filename.txt");  
 }  
  
 @Test  
 public void getContentTest() {  
 *assertEquals*(*scrapedPage*.getContent(), "content");  
 }  
  
 @Test  
 public void getEmptyFilenameTest() {  
 *assertNull*(*emptyScrapedPage*.getFilename());  
 }  
  
 @Test  
 public void getEmptyContentTest() {  
 *assertEquals*(*emptyScrapedPage*.getContent(), "");  
 }  
}

package tests.modelTests;  
  
import model.WebScrapperState;  
import model.enums.DocumentPartTypes;  
import model.enums.OutputFileTypes;  
import model.enums.OutputTypes;  
import org.junit.Test;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class WebScrapperStateTests {  
  
 private static final WebScrapperState *webScrapperState* = new WebScrapperState(DocumentPartTypes.*ALL*,  
 OutputTypes.*FILE*, OutputFileTypes.*HTML*, "myFile.html", "https://myUrl.com");  
  
 @Test  
 public void getUrlTest() {  
 *assertEquals*(*webScrapperState*.getUrl(), "https://myUrl.com");  
 }  
  
 @Test  
 public void getFilenameTest() {  
 *assertEquals*(*webScrapperState*.getFilename(), "myFile.html");  
 }  
  
 @Test  
 public void getDocumentPartTypeTest() {  
 *assertEquals*(*webScrapperState*.getDocumentPartType(), DocumentPartTypes.*ALL*);  
 }  
  
 @Test  
 public void getOutputTypeTest() {  
 *assertEquals*(*webScrapperState*.getOutputType(), OutputTypes.*FILE*);  
 }  
  
 @Test  
 public void getOutputFileTypeTest() {  
 *assertEquals*(*webScrapperState*.getOutputFileType(), OutputFileTypes.*HTML*);  
 }  
}

package tests.servicesTests;  
  
import model.WebScrapperState;  
import model.enums.DocumentPartTypes;  
import model.enums.OutputFileTypes;  
import model.enums.OutputTypes;  
import org.junit.Test;  
import services.InputArgsService;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.junit.jupiter.api.Assertions.*assertThrows*;  
  
public class InputArgsServiceTest {  
  
 @Test  
 public void parseTest\_Url() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://google.com"}),  
 new WebScrapperState(DocumentPartTypes.*ALL*, OutputTypes.*CONSOLE*, OutputFileTypes.*TXT*, null, "https://google.com"));  
 }  
  
 @Test  
 public void parseTest\_Url\_negative() {  
 Throwable exception = *assertThrows*(IllegalArgumentException.class, () -> InputArgsService.*parse*(new String[]{"wrong url"}));  
 *assertEquals*(exception.getMessage(), "Invalid URL! Please, write URL correctly. For help use command -h.");  
 }  
  
 @Test  
 public void parseTest\_Url\_f() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://google.com", "-f"}),  
 new WebScrapperState(DocumentPartTypes.*ALL*, OutputTypes.*FILE*, OutputFileTypes.*TXT*, "https:google.com.txt", "https://google.com"));  
 }  
  
 @Test  
 public void parseTest\_Url\_body() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://google.com", "-body"}),  
 new WebScrapperState(DocumentPartTypes.*BODY*, OutputTypes.*CONSOLE*, OutputFileTypes.*TXT*, null, "https://google.com"));  
 }  
  
 @Test  
 public void parseTest\_Url\_2nd\_arg\_negative() {  
 Throwable exception = *assertThrows*(IllegalArgumentException.class, () -> InputArgsService.*parse*(new String[]{"https://google.com", "something"}));  
 *assertEquals*(exception.getMessage(), "You must write output type command or part of scrapping page after URL! For help use command -h.");  
 }  
  
 @Test  
 public void parseTest\_Url\_3rd\_arg\_negative() {  
 Throwable exception = *assertThrows*(IllegalArgumentException.class, () -> InputArgsService.*parse*(new String[]{"https://google.com", "-all", "something"}));  
 *assertEquals*(exception.getMessage(), "You can write other args only after output type command (-c or -f)! For help use command -h.");  
 }  
  
 @Test  
 public void parseTest\_Url\_c\_all() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://google.com", "-c", "-all"}),  
 new WebScrapperState(DocumentPartTypes.*ALL*, OutputTypes.*CONSOLE*, OutputFileTypes.*TXT*, null, "https://google.com"));  
 }  
  
 @Test  
 public void parseTest\_Url\_c\_3rd\_arg\_negative() {  
 Throwable exception = *assertThrows*(IllegalArgumentException.class, () -> InputArgsService.*parse*(new String[]{"https://google.com", "-c", "something"}));  
 *assertEquals*(exception.getMessage(), "After -c output type you can write only scrapping document part argument! (-text / -body / -head / -all) For help use command -h.");  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_text() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://google.com", "-f", "-text"}),  
 new WebScrapperState(DocumentPartTypes.*TEXT*, OutputTypes.*FILE*, OutputFileTypes.*TXT*, "https:google.com.txt", "https://google.com"));  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_html() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "-html"}),  
 new WebScrapperState(DocumentPartTypes.*ALL*, OutputTypes.*FILE*, OutputFileTypes.*HTML*, "https:mysite.html", "https://mysite.html"));  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_filename() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "something"}),  
 new WebScrapperState(DocumentPartTypes.*ALL*, OutputTypes.*FILE*, OutputFileTypes.*TXT*, "something.txt", "https://mysite.html"));  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_head\_html() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "-head", "-html"}),  
 new WebScrapperState(DocumentPartTypes.*HEAD*, OutputTypes.*FILE*, OutputFileTypes.*HTML*, "https:mysite.html", "https://mysite.html"));  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_head\_filename() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "-head", "something"}),  
 new WebScrapperState(DocumentPartTypes.*HEAD*, OutputTypes.*FILE*, OutputFileTypes.*TXT*, "something.txt", "https://mysite.html"));  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_xml\_filename() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "-xml", "something"}),  
 new WebScrapperState(DocumentPartTypes.*ALL*, OutputTypes.*FILE*, OutputFileTypes.*XML*, "something.xml", "https://mysite.html"));  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_3rd\_arg\_negative\_filename() {  
 Throwable exception = *assertThrows*(IllegalArgumentException.class, () -> InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "negative", "something"}));  
 *assertEquals*(exception.getMessage(), "After -f must be part of scrapping page or file type argument before filename! For help use command -h.");  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_body\_xml\_filename() {  
 *assertEquals*(InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "-body", "-xml", "something"}),  
 new WebScrapperState(DocumentPartTypes.*BODY*, OutputTypes.*FILE*, OutputFileTypes.*XML*, "something.xml", "https://mysite.html"));  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_3rd\_arg\_negative\_something\_something() {  
 Throwable exception = *assertThrows*(IllegalArgumentException.class, () -> InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "-txt", "something", "something"}));  
 *assertEquals*(exception.getMessage(), "You must write arguments in order: url, -f, part of scrapping page argument, file type argument, filename! For help use command -h.");  
 }  
  
 @Test  
 public void parseTest\_Url\_f\_body\_4rth\_arg\_negative\_something() {  
 Throwable exception = *assertThrows*(IllegalArgumentException.class, () -> InputArgsService.*parse*(new String[]{"https://mysite.html", "-f", "-body", "negative", "something"}));  
 *assertEquals*(exception.getMessage(), "You must write arguments in order: url, -f, part of scrapping page argument, file type argument, filename! For help use command -h.");  
 }  
}

package tests.servicesTests;  
  
import model.ScrappedPage;  
import model.WebScrapperState;  
import model.enums.DocumentPartTypes;  
import model.enums.OutputFileTypes;  
import model.enums.OutputTypes;  
import org.junit.Test;  
import services.WebScrapingService;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.junit.jupiter.api.Assertions.*assertThrows*;  
  
public class WebScrappingServiceTest {  
 @Test  
 public void scrapTest\_All\_Console() {  
 *assertEquals*(WebScrapingService.*scrap*(new WebScrapperState(DocumentPartTypes.*ALL*, OutputTypes.*CONSOLE*, OutputFileTypes.*TXT*, null, "http://www.example.org/")),  
 new ScrappedPage("<!doctype html>\n" +  
 "<html> \n" +  
 " <head> \n" +  
 " <title>Example Domain</title> \n" +  
 " <meta charset=\"utf-8\"> \n" +  
 " <meta http-equiv=\"Content-type\" content=\"text/html; charset=utf-8\"> \n" +  
 " <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\"> \n" +  
 " <style type=\"text/css\">\n" +  
 " body {\n" +  
 " background-color: #f0f0f2;\n" +  
 " margin: 0;\n" +  
 " padding: 0;\n" +  
 " font-family: -apple-system, system-ui, BlinkMacSystemFont, \"Segoe UI\", \"Open Sans\", \"Helvetica Neue\", Helvetica, Arial, sans-serif;\n" +  
 " \n" +  
 " }\n" +  
 " div {\n" +  
 " width: 600px;\n" +  
 " margin: 5em auto;\n" +  
 " padding: 2em;\n" +  
 " background-color: #fdfdff;\n" +  
 " border-radius: 0.5em;\n" +  
 " box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);\n" +  
 " }\n" +  
 " a:link, a:visited {\n" +  
 " color: #38488f;\n" +  
 " text-decoration: none;\n" +  
 " }\n" +  
 " @media (max-width: 700px) {\n" +  
 " div {\n" +  
 " margin: 0 auto;\n" +  
 " width: auto;\n" +  
 " }\n" +  
 " }\n" +  
 " </style> \n" +  
 " </head> \n" +  
 " <body> \n" +  
 " <div> \n" +  
 " <h1>Example Domain</h1> \n" +  
 " <p>This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.</p> \n" +  
 " <p><a href=\"https://www.iana.org/domains/example\">More information...</a></p> \n" +  
 " </div> \n" +  
 " </body>\n" +  
 "</html>"));  
 }  
  
 @Test  
 public void scrapTest\_Head\_Console() {  
 *assertEquals*(WebScrapingService.*scrap*(new WebScrapperState(DocumentPartTypes.*HEAD*, OutputTypes.*CONSOLE*, OutputFileTypes.*TXT*, null, "http://www.example.org/")),  
 new ScrappedPage("<title>Example Domain</title> \n" +  
 "<meta charset=\"utf-8\"> \n" +  
 "<meta http-equiv=\"Content-type\" content=\"text/html; charset=utf-8\"> \n" +  
 "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\"> \n" +  
 "<style type=\"text/css\">\n" +  
 " body {\n" +  
 " background-color: #f0f0f2;\n" +  
 " margin: 0;\n" +  
 " padding: 0;\n" +  
 " font-family: -apple-system, system-ui, BlinkMacSystemFont, \"Segoe UI\", \"Open Sans\", \"Helvetica Neue\", Helvetica, Arial, sans-serif;\n" +  
 " \n" +  
 " }\n" +  
 " div {\n" +  
 " width: 600px;\n" +  
 " margin: 5em auto;\n" +  
 " padding: 2em;\n" +  
 " background-color: #fdfdff;\n" +  
 " border-radius: 0.5em;\n" +  
 " box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);\n" +  
 " }\n" +  
 " a:link, a:visited {\n" +  
 " color: #38488f;\n" +  
 " text-decoration: none;\n" +  
 " }\n" +  
 " @media (max-width: 700px) {\n" +  
 " div {\n" +  
 " margin: 0 auto;\n" +  
 " width: auto;\n" +  
 " }\n" +  
 " }\n" +  
 " </style>"));  
 }  
  
 @Test  
 public void scrapTest\_Body\_Console() {  
 *assertEquals*(WebScrapingService.*scrap*(new WebScrapperState(DocumentPartTypes.*BODY*, OutputTypes.*CONSOLE*, OutputFileTypes.*TXT*, null, "http://www.example.org/")),  
 new ScrappedPage("<div> \n" +  
 " <h1>Example Domain</h1> \n" +  
 " <p>This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.</p> \n" +  
 " <p><a href=\"https://www.iana.org/domains/example\">More information...</a></p> \n" +  
 "</div>"));  
 }  
  
 @Test  
 public void scrapTest\_Text\_Console() {  
 *assertEquals*(WebScrapingService.*scrap*(new WebScrapperState(DocumentPartTypes.*TEXT*, OutputTypes.*CONSOLE*, OutputFileTypes.*TXT*, null, "http://www.example.org/")),  
 new ScrappedPage("Example Domain Example Domain This domain is for use in " +  
 "illustrative examples in documents. You may use this domain in literature without prior " +  
 "coordination or asking for permission. More information..."));  
 }  
  
 @Test  
 public void scrapTest\_Text\_File() {  
 *assertEquals*(WebScrapingService.*scrap*(new WebScrapperState(DocumentPartTypes.*TEXT*, OutputTypes.*FILE*, OutputFileTypes.*TXT*, "filename.txt", "http://www.example.org/")),  
 new ScrappedPage("Example Domain Example Domain This domain is for use in " +  
 "illustrative examples in documents. You may use this domain in literature without prior " +  
 "coordination or asking for permission. More information...", "filename.txt"));  
 }  
  
 @Test  
 public void scrapTest\_negative\_url() {  
 Throwable exception = *assertThrows*(IllegalArgumentException.class, () -> WebScrapingService.*scrap*(new WebScrapperState(DocumentPartTypes.*TEXT*, OutputTypes.*FILE*, OutputFileTypes.*TXT*, "filename.txt", "http://www.exampler.org/")));  
 *assertEquals*(exception.getMessage(), "This URL address doesn’t exist!");  
 }  
}

package testSuits;  
  
import org.junit.platform.runner.JUnitPlatform;  
import org.junit.platform.suite.api.SelectPackages;  
import org.junit.runner.RunWith;  
  
@RunWith(JUnitPlatform.class)  
@SelectPackages("tests.modelTests.enumsTests")  
public class EnumsTestsSuite {  
}

package testSuits;  
  
import org.junit.platform.runner.JUnitPlatform;  
import org.junit.platform.suite.api.SelectPackages;  
import org.junit.runner.RunWith;  
  
@RunWith(JUnitPlatform.class)  
@SelectPackages("tests.modelTests")  
public class ModelTestsSuite {  
}

package testSuits;  
  
import org.junit.platform.runner.JUnitPlatform;  
import org.junit.platform.suite.api.SelectPackages;  
import org.junit.runner.RunWith;  
  
@RunWith(JUnitPlatform.class)  
@SelectPackages("tests.servicesTests")  
public class ServicesTestsSuite {  
}

package testSuits;  
  
import org.junit.platform.runner.JUnitPlatform;  
import org.junit.platform.suite.api.SelectPackages;  
import org.junit.runner.RunWith;  
  
@RunWith(JUnitPlatform.class)  
@SelectPackages("tests")  
public class AllTestSuite {  
}

# Найденные баги

Неправильно построенное регулярное выражение в

InputArgsService.parseFilename

Было

String regex = "\*" + outputFileType.getType() + "$";

Стало

String regex = ".\*\\." + outputFileType.getType().replace(".", "") + "$";

Неправильно записанные несколько раз названия классов в

InputArgsService.parseFilename

Это

OutputTypes.*getAll*()

Вместо

OutputFileTypes.*getAll*()

Вылетало NPE в 3х одинаковых местах в том же классе и использование неправильного метода в enum-ах (valueOf вместо getByTitle)

InputArgsService.parseFilename

Было

DocumentPartTypes documentPartType = DocumentPartTypes.*valueOf*(  
 List.*from*(args).stream()  
 .filter(x -> DocumentPartTypes.*getAll*().contains(x))  
 .collect(Collectors.*toList*()).get(0));

Стало

DocumentPartTypes documentPartType = DocumentPartTypes.*getByTitle*(  
 *nullSafeGet*(  
 List.*of*(args).stream()  
 .filter(x -> DocumentPartTypes.*getAll*().contains(x))  
 .collect(Collectors.*toList*()), 0));

private static String nullSafeGet(List<String> list, int i) {  
 if (list.isEmpty()) return null;  
 return list.get(0);  
}

# Выводы и заключение

Я написала работающий сервис по собственным функциональным требованиям. Было очень сложно придумать достаточное количество деталей и дополнительных функций, чтобы нашлось 30 функциональных требований, это довольно много для описания довольно простой задачи, пришлось усложнять несколько раз. В ходе написания сразу старалась придерживаться правильной архитектуры, паттернов, использовать всякие полезные улучшения (вроде Lombok, Stream API и др.), применять знания и паттерны, используемые на рабочем проекте. В ходе написания тестов писала все кейсы, какие могла придумать и негативные тесты тоже все написала, какие есть исключения в проекте. Баг было немного и проект сразу хорошо заработал, наверное это говорит обо мне что-то хорошее. Создание тестов и процесс дебага заняли 3,5 часа, рассчитывала в лучшем случае на 2, в худшем на 5, создание остального – 7 часов, рассчитывала на 8. В принципе, значения в пределах нормальной погрешности. Время полезно засекать для улучшения самостоятельной эстимации.