

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №3 по дисциплине «Автоматизация тестирования»

Обучающийся группы 6232-020402D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Александров А.А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанков А.А.

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc150372992)

[2 Результаты работы 4](#_Toc150372993)

[2.1 Результаты задания 1 4](#_Toc150372994)

[2.2 Результаты задания 2 7](#_Toc150372995)

[3 Вывод 10](#_Toc150372996)

[ПРИЛОЖЕНИЕ А Листинг программы 11](#_Toc150372997)

1. Постановка задачи

**Цель и постановка задания**

В процессе выполнения заданий познакомиться с языком Gherkin и с фреймворком Cucumber.

**Задание 1**

Разработать несколько user story для программы, реализованной в процессе 2 лабораторной работы**Задание 2**

На основе первой лабораторной работы создать Spring Boot приложение, в котором реализовать следующие рест енд-поинты:

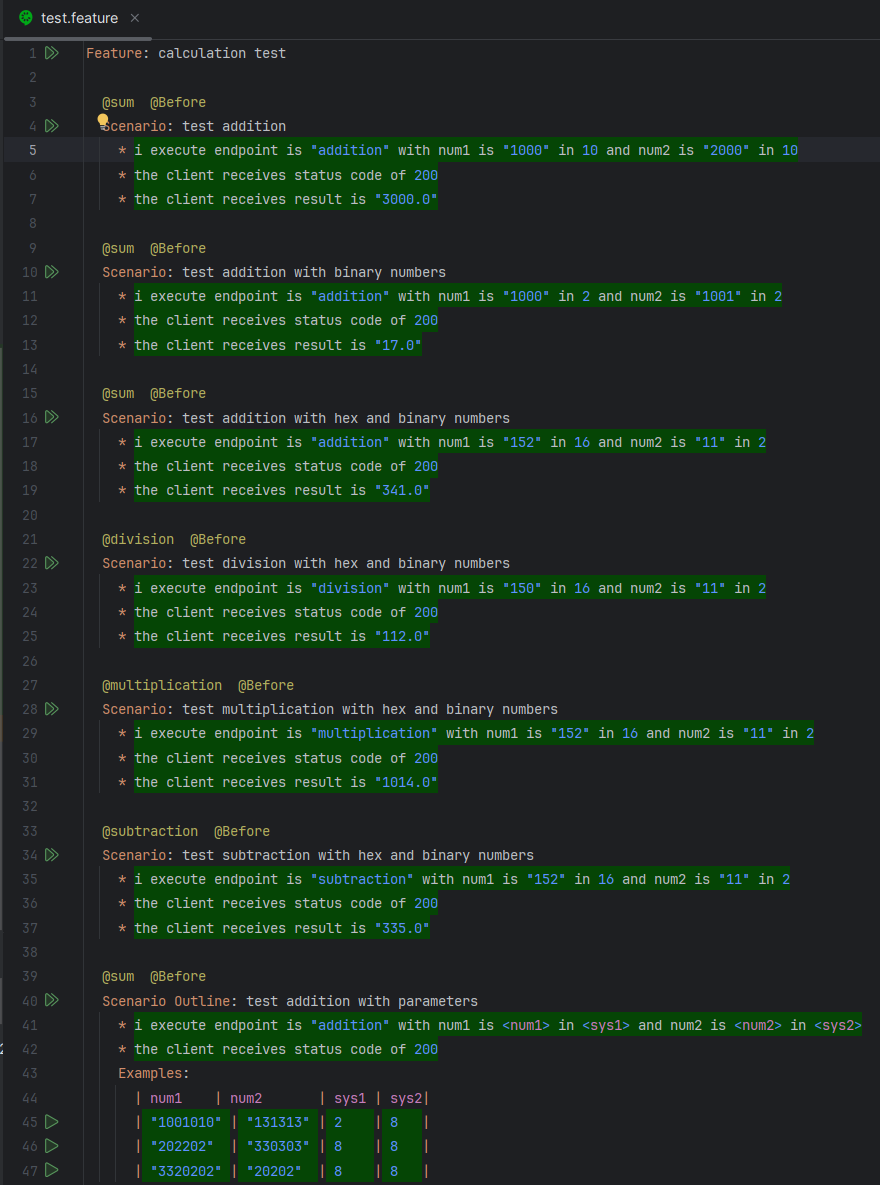
1. Описать созданные user story в нотации Gherkin.
2. Исключить миграции по накатыванию данных в БД. Перед запуском тестов БД должна быть пустая.
3. Реализовать интеграционное тестирование по разработанным user story с помощью фреймворка Cucumber. Перед запуском сценариев в feature файле устанавливается стартовое состояние БД (background). После прохода сценария, БД должна возвращаться в исходное состояние (для этого можно использовать хуки).
4. В user story долен присутствовать минимум по одному сценарию для передачи коллекций с помощью таблиц, с помощью класса, с помощью кастомного разделителя. Сценарий с использованием даты и просто обычный сценарии с несколькими аргументами
5. Результаты работы
   1. Результаты задания 1

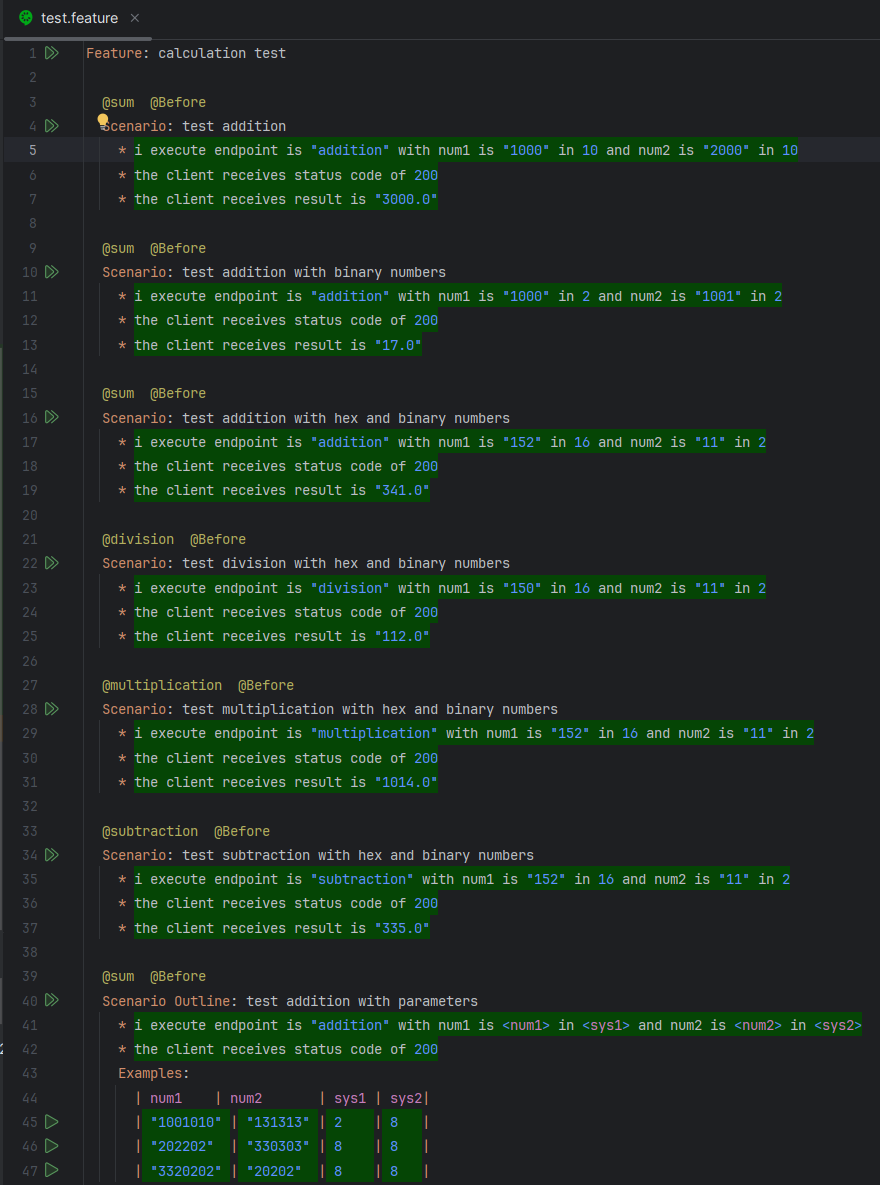
На языке Gherkin были реализованы различные user story с использованием заданных параметров, таблиц в качестве подставляемых параметров и таблиц в качестве входных данных, а так же их комбинации.

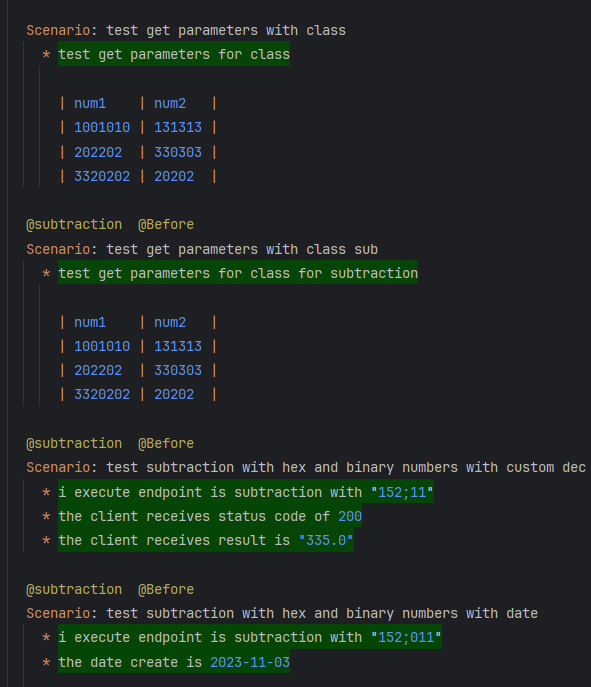
Подключим Cucumber в pom.xml:



Примеры сценариев:



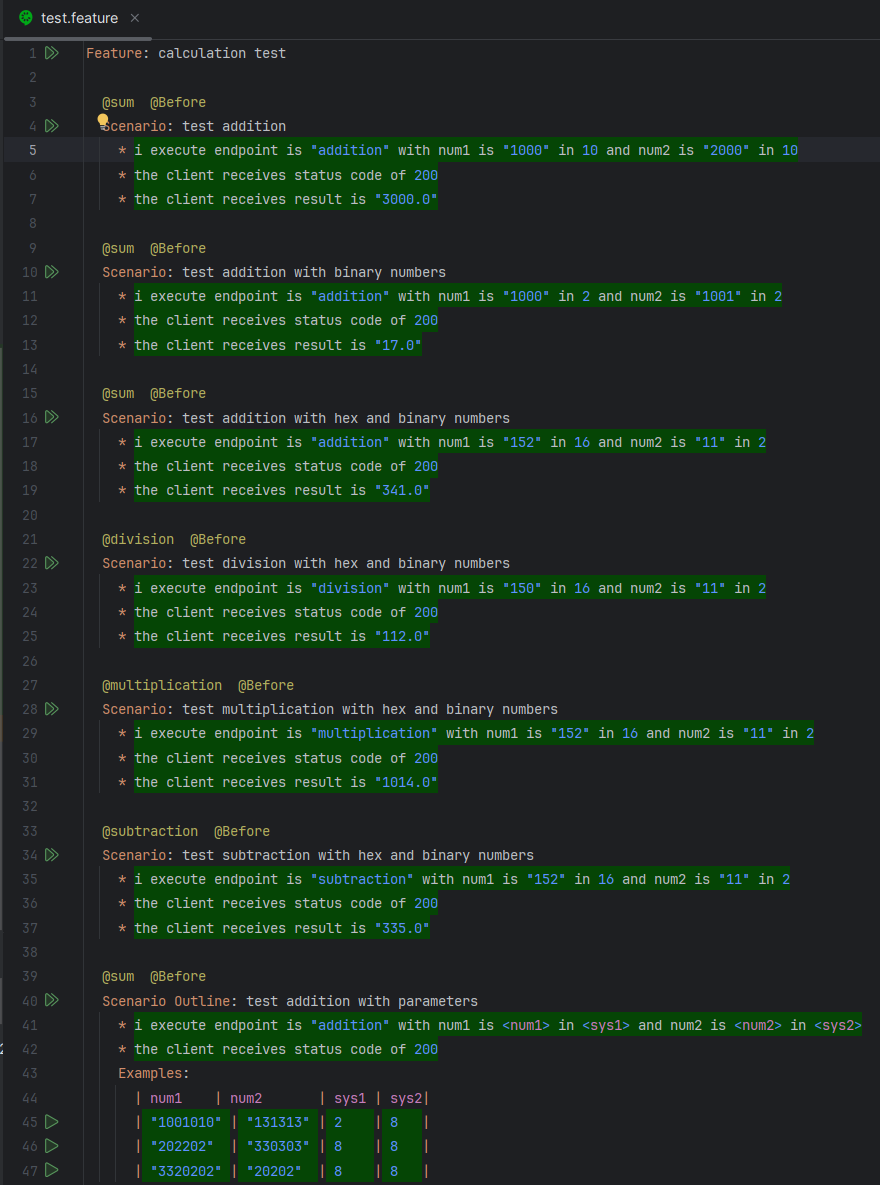




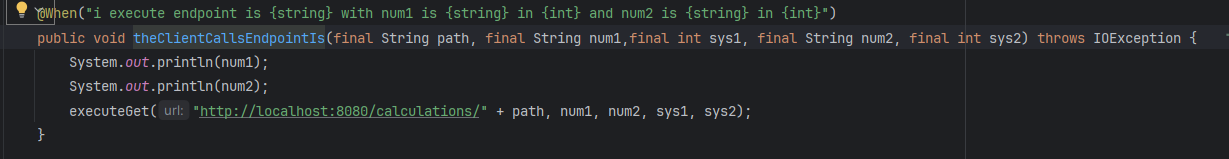
* 1. Результаты задания 2

Каждое действие в описанном сценарии имеет свою соответствующую кодовую реализацию. Для каждого шага может быть написана только одна реализация.

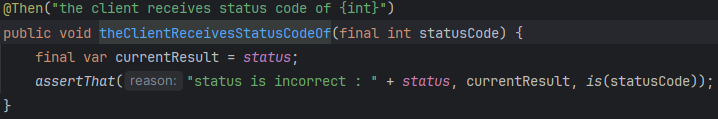
К примеру выражение:

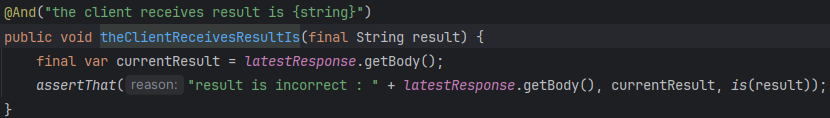


Обрабатывается:



Каждый шаг заполняет соответствующее свойство в объекте запроса, который в последствии будет отправлен в качестве параметра на выполнение сервису:

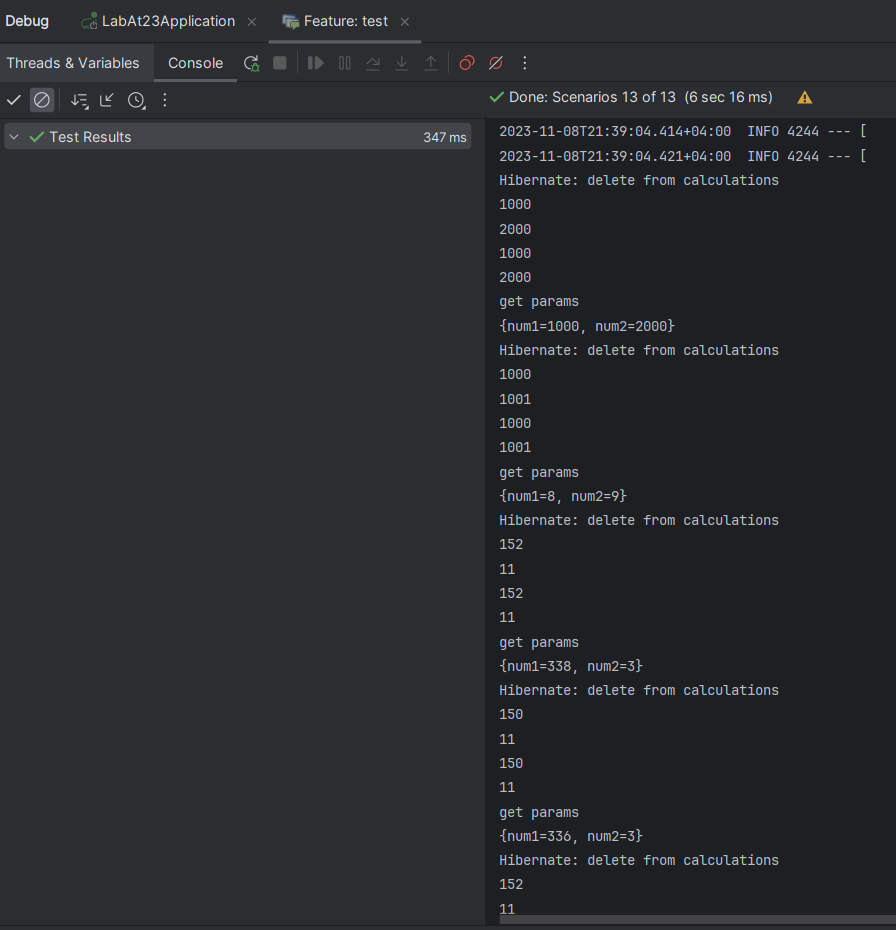




Для всех тестов используется единый конвейер, в котором регистрируются все необходимые сервисы, а также контекст базы данных, создаваемый единожды перед запуском тестов. База данных создаётся динамически средствами фреймворка Testcontainers, которая реализована в лабораторной работе №2.



Результаты тестов:



1. Вывод

В результате выполнения лабораторной работы были реализованы пользовательские сценарии на языке Gherkin, написана их кодовая реализация.

Выполнены unit-тесты, с динамическим окружением, включающим создаваемую базу данных в контейнере силами фреймворка Testcontainers.

Написан отчёт.

[Исходный код доступен по ссылке](https://github.com/Allexeey/autotests-lab1/tree/lab-23)

ПРИЛОЖЕНИЕ А  
Листинг программы

Feature: calculation test

@sum @Before

Scenario: test addition

\* i execute endpoint is "addition" with num1 is "1000" in 10 and num2 is "2000" in 10

\* the client receives status code of 200

\* the client receives result is "3000.0"

@sum @Before

Scenario: test addition with binary numbers

\* i execute endpoint is "addition" with num1 is "1000" in 2 and num2 is "1001" in 2

\* the client receives status code of 200

\* the client receives result is "17.0"

@sum @Before

Scenario: test addition with hex and binary numbers

\* i execute endpoint is "addition" with num1 is "152" in 16 and num2 is "11" in 2

\* the client receives status code of 200

\* the client receives result is "341.0"

@division @Before

Scenario: test division with hex and binary numbers

\* i execute endpoint is "division" with num1 is "150" in 16 and num2 is "11" in 2

\* the client receives status code of 200

\* the client receives result is "112.0"

@multiplication @Before

Scenario: test multiplication with hex and binary numbers

\* i execute endpoint is "multiplication" with num1 is "152" in 16 and num2 is "11" in 2

\* the client receives status code of 200

\* the client receives result is "1014.0"

@subtraction @Before

Scenario: test subtraction with hex and binary numbers

\* i execute endpoint is "subtraction" with num1 is "152" in 16 and num2 is "11" in 2

\* the client receives status code of 200

\* the client receives result is "335.0"

@sum @Before

Scenario Outline: test addition with parameters

\* i execute endpoint is "addition" with num1 is <num1> in <sys1> and num2 is <num2> in <sys2>

\* the client receives status code of 200

Examples:

| num1 | num2 | sys1 | sys2|

| "1001010" | "131313" | 2 | 8 |

| "202202" | "330303" | 8 | 8 |

| "3320202" | "20202" | 8 | 8 |

Scenario: test get parameters with class

\* test get parameters for class

| num1 | num2 |

| 1001010 | 131313 |

| 202202 | 330303 |

| 3320202 | 20202 |

@subtraction @Before

Scenario: test get parameters with class sub

\* test get parameters for class for subtraction

| num1 | num2 |

| 1001010 | 131313 |

| 202202 | 330303 |

| 3320202 | 20202 |

@subtraction @Before

Scenario: test subtraction with hex and binary numbers with custom dec

\* i execute endpoint is subtraction with "152;11"

\* the client receives status code of 200

\* the client receives result is "335.0"

@subtraction @Before

Scenario: test subtraction with hex and binary numbers with date

\* i execute endpoint is subtraction with "152;011"

\* the date create is 2023-11-03

package com.uni.labat23.definitions;

import com.fasterxml.jackson.databind.JavaType;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.datatype.jsr310.JSR310Module;

import com.uni.labat23.entity.Calculations;

import com.uni.labat23.entity.models.Numbers;

import com.uni.labat23.repository.CalculationsRepository;

import io.cucumber.java.\*;

import io.cucumber.java.en.And;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.convert.Delimiter;

import java.io.IOException;

import java.lang.reflect.Type;

import java.util.Arrays;

import java.util.Date;

import java.util.List;

import static org.hamcrest.MatcherAssert.assertThat;

import static org.hamcrest.Matchers.is;

public class StepDefsIntegrationTest extends SpringIntegrationTest {

@Autowired

CalculationsRepository calculationsRepository;

private final ObjectMapper objectMapper =

new ObjectMapper().registerModule(new JSR310Module());

@DefaultParameterTransformer

@DefaultDataTableEntryTransformer

@DefaultDataTableCellTransformer

public Object defaultTransformer(Object fromValue, Type toValueType) {

JavaType javaType = objectMapper.constructType(toValueType);

return objectMapper.convertValue(fromValue, javaType);

}

@ParameterType("(?:.+;)+.+")

public List<String> stringList(final String raw) {

final String[] values = raw.split(";");

return Arrays.asList(values);

}

@When("i execute endpoint is {string} with num1 is {string} in {int} and num2 is {string} in {int}")

public void theClientCallsEndpointIs(final String path, final String num1,final int sys1, final String num2, final int sys2) throws IOException {

System.out.println(num1);

System.out.println(num2);

executeGet("http://localhost:8080/calculations/" + path, num1, num2, sys1, sys2);

}

@When("i execute endpoint is subtraction with \"{stringList}\"")

public void theClientCallsEndpointIs(@Delimiter("; ") final List<String> arg) throws IOException {

System.out.println("theClientCallsEndpointIs");

executeGet("http://localhost:8080/calculations/subtraction", arg.get(0), arg.get(1), 16, 2);

}

@Then("the client receives status code of {int}")

public void theClientReceivesStatusCodeOf(final int statusCode) {

final var currentResult = status;

assertThat("status is incorrect : " + status, currentResult, is(statusCode));

}

@And("the client receives result is {string}")

public void theClientReceivesResultIs(final String result) {

final var currentResult = latestResponse.getBody();

assertThat("result is incorrect : " + latestResponse.getBody(), currentResult, is(result));

}

@And("^the date create is (.+)$")

public boolean theClientReceivesResultIs(final Date result) {

final var currentResult = calculationsRepository.findDate("338");

List<Calculations> calculations1 = calculationsRepository.findAll();

for (Calculations calculation : calculations1) {

System.out.println(calculation);

}

return currentResult.after(result);

}

@Given("test get parameters for class")

public void testGetParametersForClass(final List<Numbers> arg) throws Throwable {

for (final Numbers numbers : arg) {

System.out.print(numbers.getNum1());

System.out.print(numbers.getNum2());

}

}

@Given("test get parameters for class for subtraction")

public void testGetParametersForClassSub(final List<Numbers> arg) throws Throwable {

for (final Numbers numbers : arg) {

executeGet("http://localhost:8080/calculations/subtraction", String.valueOf(numbers.getNum1()), String.valueOf(numbers.getNum2()), 8, 8);

}

}

@Before

public void cleanDatabase() {

calculationsRepository.clear();

}

}