**Основы интеграционного тестирования**

*(заполняется спикером урока)*

**Что нужно знать для проверки ДЗ**

Посмотреть 1 раз воркшоп. Для ДЗ у студентов должно получиться ровно то, что я делал онлайн в воркшопе, так что смотреть весь урок для проверки ДЗ не нужно.

**Сам ответ - файл, где корректно сделаны все тесты - доступен в конце данного чек-листа.**

**Чек-лист:**

1. студент дал ссылку на свой репозиторий, или предоставил полностью код файла src/test/java/org/mycompany/MyTranslationServiceTest\_TODO.java
2. в файле написан код для всех трех тестовых методов
3. тесты запускаются и все три теста зеленые
4. ниже будут детали кода, ожидаемого в тестах. Также для примера можно посмотреть файл с ответами, упомянутый выше
   1. для теста translateWithGoogle\_anySentenceAndTargetLanguageIsRu\_success:
      1. метод translateWithGoogle вызван с языком “ru”
      2. присутствует мок when метода translate.translate
      3. присутствует мок when метода translation.getTranslatedText
      4. присутствует assertEquals, проверяющий возвращаемый результат и ожидаемый
      5. присутствует verify вызова translate.translate
      6. присутствует verify вызова translation.getTranslatedText
      7. (опционально) присутствует verifyNoMoreInteractions объекта translate
      8. (опционально) присутствует verifyNoMoreInteractions объекта translation
   2. для теста translateWithGoogle\_anySentenceAndTargetLanguageIsNotRu\_failure:
      1. метод translateWithGoogle вызван с языком не “ru”
      2. присутствует assertThrows (или проверка другим способом) IllegalArgumentException при вызове translateWithGoogle
      3. (опционально) присутствует verifyNoInteractions объекта translate
      4. (опционально) присутствует verifyNoInteractions объекта translation
   3. для теста translateWithGoogle\_googleTranslateThrowsException\_failure:
      1. метод translateWithGoogle вызван с языком “ru”
      2. присутствует мок when метода translate.translate, кидающий RuntimeException
      3. присутствует assertThrows (или проверка другим способом) MyTranslationServiceException при вызове translateWithGoogle
      4. присутствует verify вызова translate.translate
      5. (опционально) присутствует verifyNoMoreInteractions объекта translate
      6. (опционально) присутствует verifyNoInteractions объекта translation

**Итого, минимум, чтобы принять ДЗ**

- см список выше - все, за исключением пунктов с пометкой (опционально)

package org.mycompany;

import com.google.cloud.translate.Translate;

import com.google.cloud.translate.Translation;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.ArgumentMatchers;

import org.mockito.Mock;

import org.mockito.Mockito;

import org.mockito.junit.jupiter.MockitoExtension;

import static org.junit.jupiter.api.Assertions.*assertEquals*;

import static org.junit.jupiter.api.Assertions.*assertThrows*;

import static org.mockito.ArgumentMatchers.*eq*;

@ExtendWith(MockitoExtension.class)

class MyTranslationServiceTest\_Answers {

*/\*\**

*\* Mocked dependency instead of actual Translate object for Google API.*

*\*/*

@Mock

private Translate googleTranslate;

*/\*\**

*\* Mocked result of API call instead of real one.*

*\*/*

@Mock

private Translation googleTranslateResult;

*/\*\**

*\* 1. Happy case test.*

*\* <p>*

*\* When `MyTranslationService::translateWithGoogle` method is called with any sentence and target language is equal to "ru",*

*\* `googleTranslate` dependency should be called and `translation.getTranslatedText()` returned.*

*\* No other interactions with `googleTranslate` dependency should be invoked apart from a single call to `googleTranslate.translate()`.*

*\*/*

@Test

void translateWithGoogle\_anySentenceAndTargetLanguageIsRu\_success() {

// given

var myTranslationService = new MyTranslationService(googleTranslate);

var sentence = "Some sentence";

var targetLanguage = "ru";

var expectedTranslation = "Некое предложение";

// we tell our mocked object of type `translate` to return our another mocked object of type `translation`

Mockito.*when*(googleTranslate.translate(*eq*(sentence), ArgumentMatchers.*any*())).thenReturn(googleTranslateResult);

// and here we tell our mocked object of type `translation` to return our expected result when `getTranslatedText` is called

Mockito.*when*(googleTranslateResult.getTranslatedText()).thenReturn(expectedTranslation);

// when

String result = myTranslationService.translateWithGoogle(sentence, targetLanguage);

// then

*assertEquals*(expectedTranslation, result);

// verify that `translate` method was actually called on our mocked `googleTranslate` object

Mockito.*verify*(googleTranslate).translate(*eq*(sentence), ArgumentMatchers.*any*());

// verify that nothing else was called on it

Mockito.*verifyNoMoreInteractions*(googleTranslate);

// verify that `getTranslatedText` was called on our mocked `googleTranslateResult` object

Mockito.*verify*(googleTranslateResult).getTranslatedText();

// verify that nothing else was called on it

Mockito.*verifyNoMoreInteractions*(googleTranslateResult);

}

*/\*\**

*\* 2. Unhappy case test when target language is not supported.*

*\* <p>*

*\* When `MyTranslationService::translateWithGoogle` method is called with any sentence and target language is not equal to "ru",*

*\* `IllegalArgumentException` should be thrown. `googleTranslate` dependency should not be called at all.*

*\*/*

@Test

void translateWithGoogle\_anySentenceAndTargetLanguageIsNotRu\_failure() {

// given

var myTranslationService = new MyTranslationService(googleTranslate);

var sentence = "Some sentence";

var targetLanguage = "es";

// when, then

// assert that exception is thrown:

*assertThrows*(

// of this type

IllegalArgumentException.class,

// as a result of this method call

() -> myTranslationService.translateWithGoogle(sentence, targetLanguage)

);

// verify that `googleTranslate` dependency wasn't used at all, no methods were called on it

Mockito.*verifyNoInteractions*(googleTranslate);

// verify that `googleTranslateResult` dependency wasn't used at all, no methods were called on it

Mockito.*verifyNoInteractions*(googleTranslateResult);

}

*/\*\**

*\* 3. Unhappy case test when Google Translate call throws exception.*

*\* <p>*

*\* When `MyTranslationService::translateWithGoogle` method is called with any sentence and target language is equal to "ru",*

*\* `googleTranslate` dependency should be called. When `googleTranslate` dependency throws exception, it should be*

*\* wrapped into `MyTranslationServiceException` and the latter should be thrown from our method.*

*\*/*

@Test

void translateWithGoogle\_googleTranslateThrowsException\_failure() {

// given

var myTranslationService = new MyTranslationService(googleTranslate);

var sentence = "Some sentence";

var targetLanguage = "ru";

// we tell our mocked object of type `translate` to throw an exception

Mockito.*when*(googleTranslate.translate(*eq*(sentence), ArgumentMatchers.*any*())).thenThrow(new RuntimeException());

// when, then

// assert that exception is thrown:

*assertThrows*(

// of this type

MyTranslationServiceException.class,

// as a result of this method call

() -> myTranslationService.translateWithGoogle(sentence, targetLanguage)

);

// verify that `translate` method was actually called on our mocked `googleTranslate` object

Mockito.*verify*(googleTranslate).translate(*eq*(sentence), ArgumentMatchers.*any*());

// verify that nothing else was called on it

Mockito.*verifyNoMoreInteractions*(googleTranslate);

// verify that `googleTranslateResult` dependency wasn't used at all, no methods were called on it

Mockito.*verifyNoInteractions*(googleTranslateResult);

}

}