**Инструменты BDD для Java: Cucumber-JVM, Cukes-rest, JBehave**

*(заполняется спикером урока)*

**Что нужно знать для проверки ДЗ**

1. Язык Gherkin: Язык написания спецификаций и сценариев используя ключевые слов, таких как Feature, Scenario, Given, When и Then. Можно посмотреть в модуле “Спецификация и язык Gherkin” прошлого урока “Основы поведенческого тестирования”.
2. Посмотреть 1 раз воркшоп “Воркшоп: Пример поведенческого теста” и “Воркшоп: Правила написания хорошего BDD теста” на странице урока “Основы поведенческого тестирования”.
3. Cucumber expression или даже регулярные выражения. Можно посмотреть примеры в модуле “Воркшоп: Определения шагов сценария” текущего урока. Также может помочь инструмент <https://cucumber.github.io/try-cucumber-expressions/> для проверки Cucumber выражений - он также генерирует соответствующие регулярные выражения и содержит подсказки.
4. Посмотреть 1 раз “Воркшоп: Расширенный функционал Cucumber” текущего урока, чтобы понимать что такое Scenario Outline и Examples; Background; Hooks; Теги.
5. Посмотреть 1 раз “Использование библиотек проверки утверждений JUnit 5” текущего урока и посмотреть статью “Примеры утверждений JUnit 5” <https://habr.com/ru/articles/591587/> чтобы представлять какие проверки могут добавить ученики в своих @Then методах.
6. Посмотреть 1 раз “Воркшоп: Преобразование типов в Cucumber” - это не является обязательным для студента, но если он будет использовать таблицы или пользовательские типы - за это можно похвалить.
7. Посмотреть 1 раз “Воркшоп: Отчеты о тестировании в Cucumber”, чтобы понимать как выглядит отчет о тестировании, который должен быть приложен в домашней работе.

**Чек-лист:**

1. Описание функциональности: Удостоверьтесь, что описание функций в Feature четко сформулировано и понятно.
2. Описание роли в формате: как (в роли) / чтобы достичь / я хочу, чтобы
3. Сценарии: Оцените наличие разнообразных сценариев (минимум 3) использования
4. Конкретные шаги: Убедитесь, что каждый сценарий включает в себя конкретные шаги (Given, When, Then), описывающие поведение системы.
5. Читаемость: Проверьте, насколько легко читаются сценарии и понятны ли они. Понятность сценария для неподготовленного члена команды одно из качеств использования Gherkin.
6. Использование ключевых слов: Проверьте, что студенты используют ключевые слова Gherkin в соответствии с их назначением.
7. В Given должны быть шаги, которые приводят систему к нужному состоянию, например, чтобы купить товар как зарегистрированный пользователь можно использовать шаг “Given пользователь успешно создан”.
8. В When должна быть действия: открыл, добавил, нажал и т.д.
9. В Then должны быть проверки: пользователь на нужной странице, товар успешно добавлен, цена правильная, деньги списались и т.д.
10. Как минимум один шаг (Given, When или Then) должен быть параметризованным используя либо Cucumber Expression, либо регулярные выражения. Можно похвалить студента, если он также реализует использование таблиц с данными или пользовательский тип данных, хотя это и не обязательно.
11. Как минимум один сценарий должен быть обобщенным, то есть иметь Scenario Outline и Examples. Соответственно в Examples должно быть минимум 3 строки: заголовок и 2 строки для данных сценариев.
12. Feature должно иметь Background с предусловием, то есть тот шаг, который применяется ко всем сценариям в самом начале. Если Background содержит действительно шаг, который было бы хорошо вынести в предусловие - это плюс, если вынесен тот шаг, который не обязательно выносить - так как он нигде и не повторяется и у студента всего 1 сценарий - то это не ошибка, но можно про это упомянуть.
13. В проекте должен быть класс с хуками (Hooks) и использованием хотя бы одной аннотации на выбор из: @Before, @After, @BeforeStep, @AfterStep, @BeforeAll, @AfterAll. Использование хука с условием у студента будет дополнительным плюсом.
14. В Feature должны использоваться теги на любом уровне (Feature, Scenario, Scenario Outline). Можно похвалить, если было несколько тегов и есть сценарии @smoke, @positive, @negative и это можно использовать для фильтрации тестов по смыслу, виду тестирования и т.д.
15. В StepDefinitions классе есть хотя бы один реализованный @Then метод, в котором используется Assert (проверка утверждений). Можно похвалить, если будут более сложные проверки, чем в уроке Assertions.assertEquals(expectedValue, actualResult);
16. Финальное решение прислано в виде GitHub репозитория.
17. Также присутствует скриншот отчета, на котором видно запущенные тесты из репозитория. Можно похвалить, если это будет не стандартный отчет Cucumber, а отчет Allure Report или что-то аналогичное.

**Итого, минимум, чтобы принять ДЗ**

1. Описание функции.
2. Наличие минимум 1 сценария использования для функции.
3. Корректное использование ключевых слов Gherkin.
4. Читаемость и понимаемость сценариев файлов.
5. Каждый шаг сценария реализован в шагах сценария (Step Definitions).
6. Хотя бы 1 шаг использует Cucumber expressions или регулярные выражения.
7. Используется как минимум 1 тег.
8. Используется как минимум 1 хук.
9. Хотя бы 1 сценарий использует Scenario Outline и Examples.
10. В Feature используется Background.
11. Хотя бы один реализованный @Then шаг содержит Assert.
12. Код выложен на github.
13. Приложен скриншот с отчетом о тестировании.