Lab 12 : Cucumber

01418471 Introduction to Software Engineering

Fork และ Clone โปรเจคตัวอย่าง

1. ไปที่ repo ของตัวอย่างการทดสอบด้วย Cucumber ที่ https://github.com/ladyusa/cucumber-shop
2. Fork repo นี้ และ clone repo ของตนเองลงมาที่เครื่อง และ open มาใน IntelliJ
3. สังเกตที่ dependency ใน pom.xml ดังนี้ เราต้องใช้ junit, cucumber-java และ cucumber-junit

```
<dependencies>
   <dependency>
       <groupId>org.junit.jupiter</groupId>
       <artifactId>junit-jupiter-api</artifactId>
       <version>5.9.0</version>
       <scope>test</scope>
   </dependency>
   <dependency>
       <groupId>io.cucumber</groupId>
       <artifactId>cucumber-java</artifactId>
       <version>7.8.0</version>
   </dependency>
   <dependency>
       <groupId>io.cucumber</groupId>
       <artifactId>cucumber-junit</artifactId>
       <version>7.8.0</version>
       <scope>test</scope>
   </dependency>
</dependencies>
```

4. สังเกตด้วยว่า เรามี tag build ที่จะช่วยในการ compile/build ปรับเวอร์ชันจาวาให้ตรงกับเวอร์ชันของเครื่องนิสิต

```
<build>
   <plugins>
       <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-compiler-plugin</artifactId>
           <configuration>
               <source>15</source>
               <target>15</target>
           </configuration>
       </plugin>
   </plugins>
</build>
```

3. สังเกตในโฟลเดอร์ src/main/java จะมี package ชื่อ ku.shop จะมีไฟล์ที่เกี่ยวข้องกับการซื้อขายสินค้า

4. สังเกตในโฟลเดอร์ src/test/resources จะมีไฟล์ชื่อ buy.feature ซึ่งเก็บ feature และ (test) scenario ต่าง ๆ ที่เกี่ยวข้องกับ feature นี้

```
Feature: Buy products
   As a customer
   I want to buy products

Background:
   Given the store is ready to service customers
   And a product "Bread" with price 20.50 and stock of 5 exists
   And a product "Jam" with price 80.00 and stock of 10 exists

Scenario: Buy one product
   When I buy "Bread" with quantity 2
   Then total should be 41.00

Scenario: Buy multiple products
   When I buy "Bread" with quantity 2
   And I buy "Jam" with quantity 1
   Then total should be 121.00
```

5. สังเกตในโฟลเดอร์ src/test/java/ku.shop จะมีคลาส BuyStepdefs ที่กำหนดว่า ประโยคต่าง ๆ ใน feature file จะใช้โค้ดใดในการรันและทดสอบ

```
package ku.shop;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

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

public class BuyStepdefs {

   private ProductCatalog catalog;
   private Order order;

   @Given("the store is ready to service customers")
   public void the_store_is_ready_to_service_customers() {
       catalog = new ProductCatalog();
       order = new Order();
   }
```

```java
    @Given("a product {string} with price {float} and stock of {int} exists")
    public void a_product_exists(String name, double price, int stock) {
        catalog.addProduct(name, price, stock);
    }

    @When("I buy {string} with quantity {int}")
    public void i_buy_with_quantity(String name, int quantity) {
        Product prod = catalog.getProduct(name);
        order.addItem(prod, quantity);
    }

    @Then("total should be {float}")
    public void total_should_be(double total) {
        assertEquals(total, order.getTotal());
    }
}
```

6. สังเกตในโฟลเดอร์ src/test/java/ku.shop จะมีคลาส BuyUAT สั้น ๆ ที่ระบุว่า ให้รัน Cucumber โดยใช้ feature file ในโฟลเดอร์ใด และให้แสดงผลแบบใด

```java
package ku.shop;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(features = "classpath:features",
                 plugin = {"pretty", "html:target/cucumber.html"})
public class BuyUAT {
}
```

7. ลองรัน test โดยรันไฟล์ BuyUAT ซึ่งควรได้ผลดังนี้

```
Tests passed: 2 of 2 tests – 174 ms
BuyUAT (ku.shop)                                   174 ms
  Buy products                                     174 ms
    Buy one product                                159 ms
    Buy multiple products                           15 ms

/Library/Java/JavaVirtualMachines/jdk-11.0.13.jdk/Contents/Home/bin/java ...

Scenario: Buy one product                                  # features/buy.feat
  Given the store is ready to service customers            # ku.shop.BuyStepde
  And a product "Bread" with price 20.50 and stock of 5 exists # ku.shop.BuyStepde
  And a product "Jam" with price 80.00 and stock of 10 exists  # ku.shop.BuyStepde
  When I buy "Bread" with quantity 2                       # ku.shop.BuyStepde
  Then total should be 41.00                               # ku.shop.BuyStepde

Scenario: Buy multiple products                            # features/buy.feat
  Given the store is ready to service customers            # ku.shop.BuyStepde
  And a product "Bread" with price 20.50 and stock of 5 exists # ku.shop.BuyStepde
  And a product "Jam" with price 80.00 and stock of 10 exists  # ku.shop.BuyStepde
  When I buy "Bread" with quantity 2                       # ku.shop.BuyStepde
  And I buy "Jam" with quantity 1                          # ku.shop.BuyStepde
  Then total should be 121.00                              # ku.shop.BuyStepde
```

12. ให้ไปที่ไฟล์รายงานการทดสอบ อยู่ที่ target/cucumber.html ให้เปิดไฟล์ด้วย browser จะได้ผลดังนี้

```
classpath:features/buy.feature

Feature: Buy products

  As a customer
  I want to buy products

  Background:

    Given the store is ready to service customers
    And a product "Bread" with price 20.50 and stock of 5 exists
    And a product "Jam" with price 80.00 and stock of 10 exists

  Scenario: Buy one product

    When I buy "Bread" with quantity 2
    Then total should be 41.00

  Scenario: Buy multiple products

    When I buy "Bread" with quantity 2
    And I buy "Jam" with quantity 1
    Then total should be 121.00
```

การใช้ Scenario Outline

Scenario outline ช่วยในการสร้างหลายกรณีทดสอบใน 1 scenario

1. ใน feature เราสามารถเขียน scenario ด้วย `Scenario Outline:` และ `Examples:` เพื่อระบุหลายกรณีทดสอบใน 1 scenario ได้ดังตัวอย่างต่อไปนี้

```
Scenario Outline: Buy one product in table
   When I buy <product> with quantity <quantity>
   Then total should be <total>
   Examples:
       | product   | quantity |  total   |
       | "Bread"   |     1    |   20.50  |
       | "Jam"     |     2    |  160.00  |
```

2. ใช้ step definition เดิมได้

3. Cucumber จะเห็นเป็น 2 scenario ดังผลการรันดังนี้

```
Scenario Outline: Buy one product in table
  Given the store is ready to service customers
  And a product "Bread" with price 20.50 and stock of 5 exists
  And a product "Jam" with price 80.00 and stock of 10 exists
  When I buy "Bread" with quantity 1
  Then total should be 20.50

Scenario Outline: Buy one product in table
  Given the store is ready to service customers
  And a product "Bread" with price 20.50 and stock of 5 exists
  And a product "Jam" with price 80.00 and stock of 10 exists
  When I buy "Jam" with quantity 2
  Then total should be 160.00
```

ตัวอย่างเพิ่มเติม

1. ไปที่ repo ของตัวอย่าง Cucumber เพิ่มเติมที่ https://github.com/ladyusa/cucumber-atm
2. Fork repo นี้ และ clone repo ของตนเองลงมาที่เครื่อง
3. สังเกตในโฟลเดอร์ src/test/resources จะมีไฟล์ชื่อ *.feature หลายไฟล์ ตามบริบทที่ต่างกัน
4. สังเกตในโฟลเดอร์ src/test/java/ku.atm จะมีคลาสเทส ทั้งระดับ UAT และ Unit Test เราสามารถรันพร้อมกันได้ โดยรันทั้งโฟลเดอร์

การบ้าน

โปรเจค Shop

1. เพิ่มสินค้าอะไรก็ได้อีก 1 อย่าง และเพิ่มสินค้านี้เข้าไปใน scenario outline ใน Examples: clause
2. เพิ่ม feature และ step definition เพื่อทดสอบการตัดสต๊อคสินค้า กล่าวคือ ถ้ามีสินค้า 10 ชิ้น และลูกค้าซื้อไป 2 ชิ้น สินค้าต้องเหลือในสต๊อค 8 ชิ้น
3. Commit และ push ขึ้น repo ของตนเองเพื่อส่งงาน

โปรเจค ATM

4. เพิ่ม feature และ step definition เพื่อทดสอบฝากเงิน
5. Commit และ push ขึ้น repo ของตนเองเพื่อส่งงาน