Cucumber

Usa Sammapun

Different levels of test

- **Unit testing** - test individual unit of software such as method, class
- **Integration testing** - test interaction between class/component
- **System testing** - end-to-end test, including UI and database
- **User acceptance testing** - test whether the software meets users' satisfaction

Acceptance test

- Unit tests are necessary but insufficient as verification tools.
  - Verify that the small elements of the system work
  - Do not verify that the system works properly as a whole.
- Unit tests : white box tests to verify individual mechanisms of system.
- Acceptance tests : black box tests to verify customer requirements are being met.

Source : Robert C. Martin, Micah Martin. Agile Principles, Patterns, and Practices in C#. Prentice Hall, 2006.

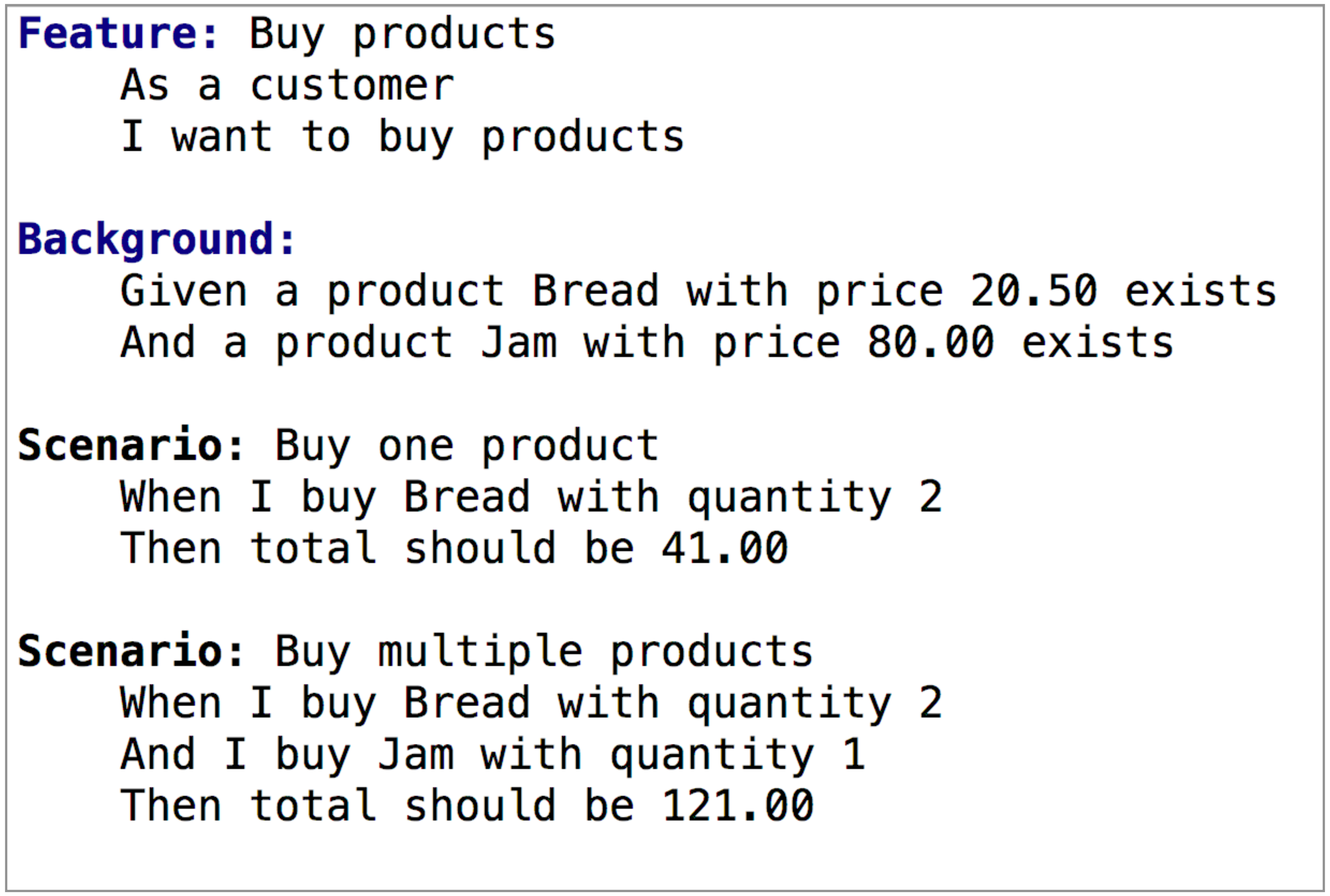
Acceptance test

- Acceptance tests
  - Customers write them to verify a desired feature is correct
  - Programmers read them to truly understand the feature.
- **Unit tests** serve as executable **documentation** for **internal** of system
- UAT serve as executable **documentation** of **feature** of system
- Acceptance tests—ultimate **requirement documentation** of features

Source : Robert C. Martin, Micah Martin. Agile Principles, Patterns, and Practices in C#. Prentice Hall, 2006.

Cucumber

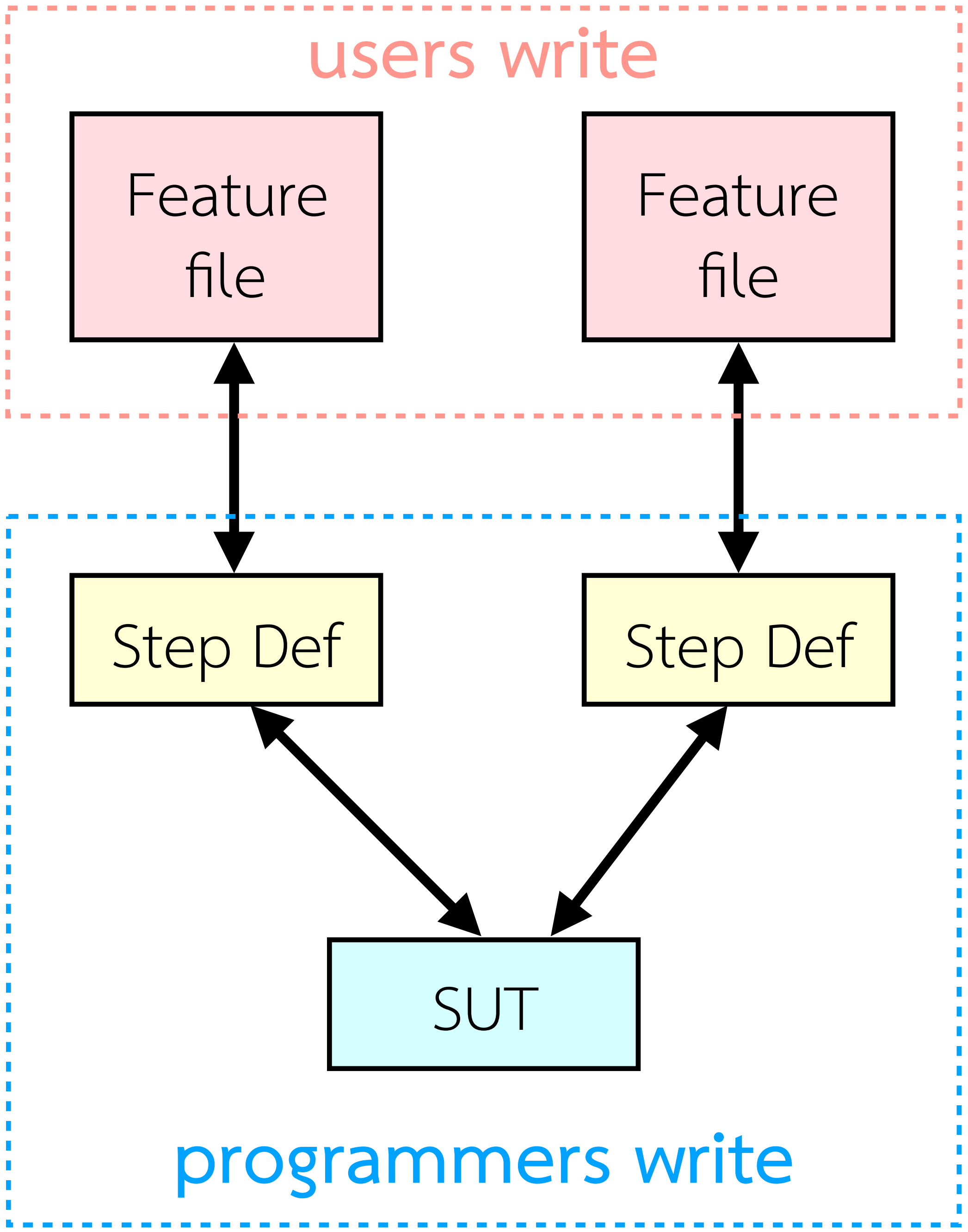
- automated acceptance testing tools
  - Allows users to specify requirement + test case via a “feature” file



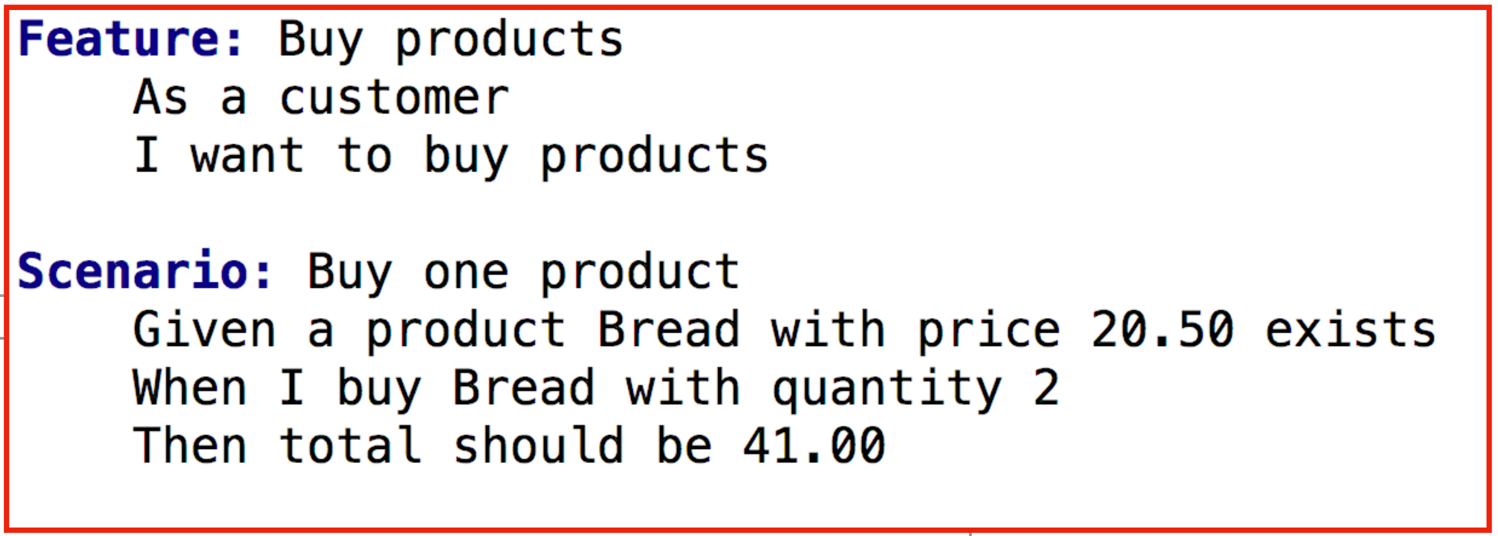
```
Feature: Buy products
    As a customer
    I want to buy products

Background:
    Given a product Bread with price 20.50 exists
    And a product Jam with price 80.00 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
```

Cucumber

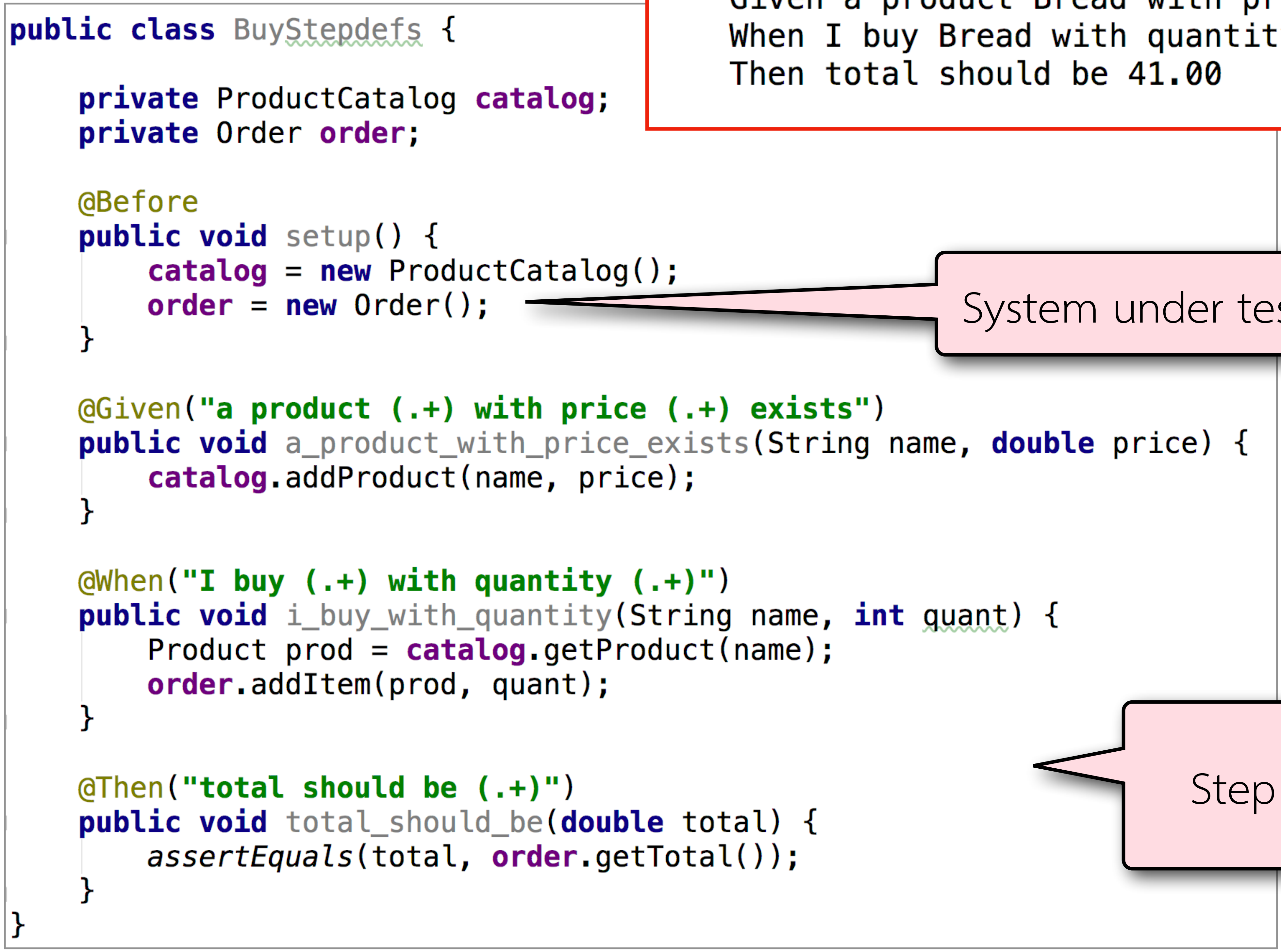
- Feature file : ระบุกรณีทดสอบ
- Testing engine (Cucumber / jUnit)
  - เชื่อมกรณีทดสอบกับ feature และรันเทส
- Step definition file
  - นิยามประโยคใน feature ให้เป็นโค้ดเทส
- System under test (SUT)
  - ซอฟต์แวร์/ระบบที่ต้องการทดสอบ



Simple scenario



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

Scenario: Buy one product
    Given a product Bread with price 20.50 exists
    When I buy Bread with quantity 2
    Then total should be 41.00
```



```
public class BuyStepdefs {

    private ProductCatalog catalog;
    private Order order;

    @Before
    public void setup() {
        catalog = new ProductCatalog();
        order = new Order();
    }

    @Given("a product (.+) with price (.+) exists")
    public void a_product_with_price_exists(String name, double price) {
        catalog.addProduct(name, price);
    }

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

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

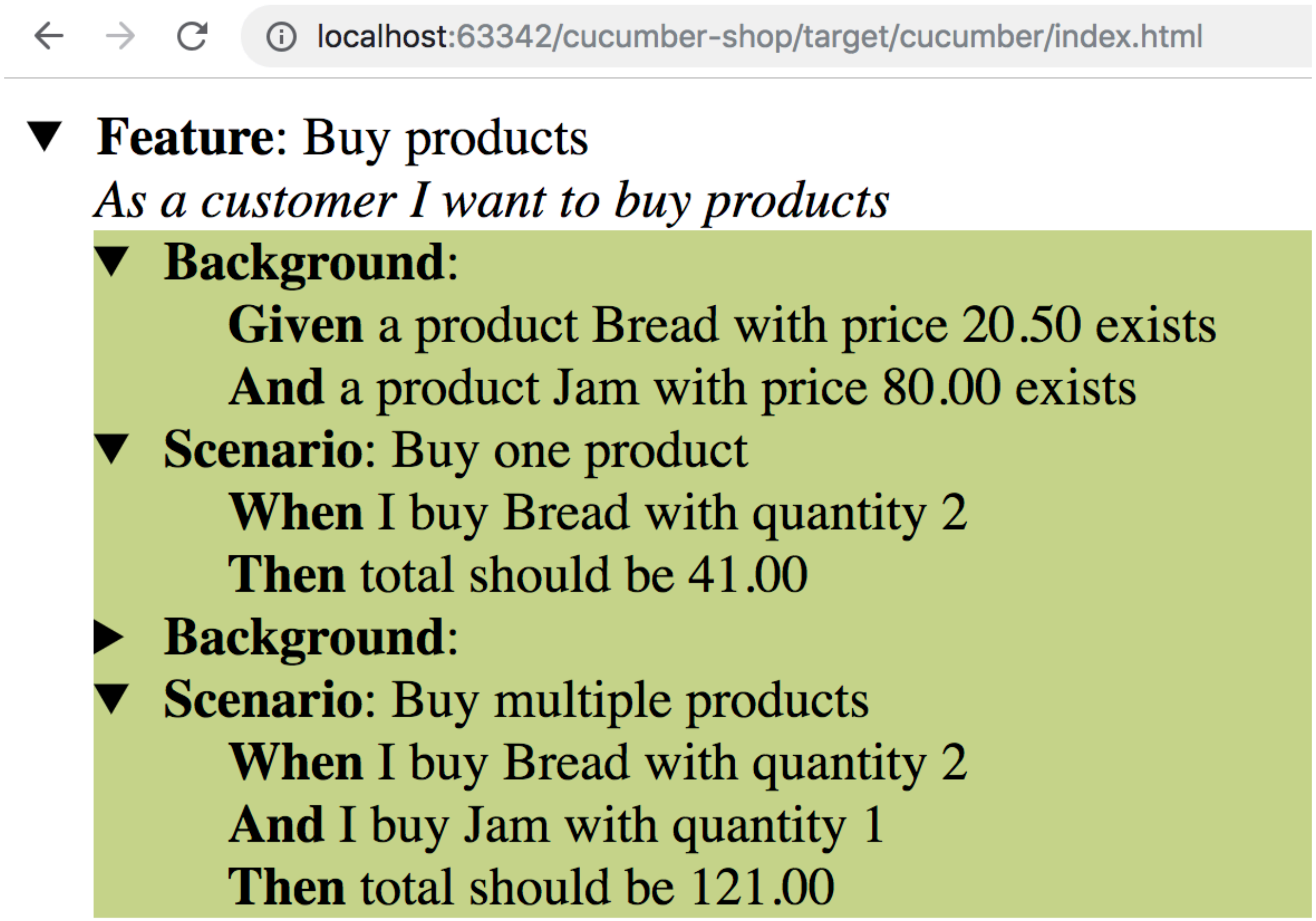
System under test

Step definition file

Cucumber

- Software development collaboration tool
  - Lets customers and analysts write "executable" acceptance tests using simple formats in a "feature" file
  - Developers write "step definition" to link the test cases with the actual system itself
- Cucumber compares these test cases in a "feature" file with actual values from SUT (system under test) and colors results
  - Work with Java, Ruby, JavaScript, etc.

Test results


localhost:63342/cucumber-shop/target/cucumber/index.html

▼ **Feature**: Buy products
As a customer I want to buy products
▼ **Background**:
Given a product Bread with price 20.50 exists
And a product Jam with price 80.00 exists
▼ **Scenario**: Buy one product
When I buy Bread with quantity 2
Then total should be 41.00
▶ **Background**:
▼ **Scenario**: Buy multiple products
When I buy Bread with quantity 2
And I buy Jam with quantity 1
Then total should be 121.00

Writing executable requirement / test case

Given a context

When an event happens

Then an outcome *should* occur

- Keyword เช่น
  - Given
  - When
  - Then
  - And
  - Background

Simple scenario

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

Scenario: Buy one product
    Given a product Bread with price 20.50 exists
    When I buy Bread with quantity 2
    Then total should be 41.00
```

```java
public class BuyStepdefs {

    private ProductCatalog catalog;
    private Order order;

    @Before
    public void setup() {
        catalog = new ProductCatalog();
        order = new Order();
    }

    @Given("a product (.+) with price (.+) exists")
    public void a_product_with_price_exists(String name, double price) {
        catalog.addProduct(name, price);
    }

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

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

Before hook : รันก่อนทดสอบ
ลักษณะเดียวกับ @BeforeEach

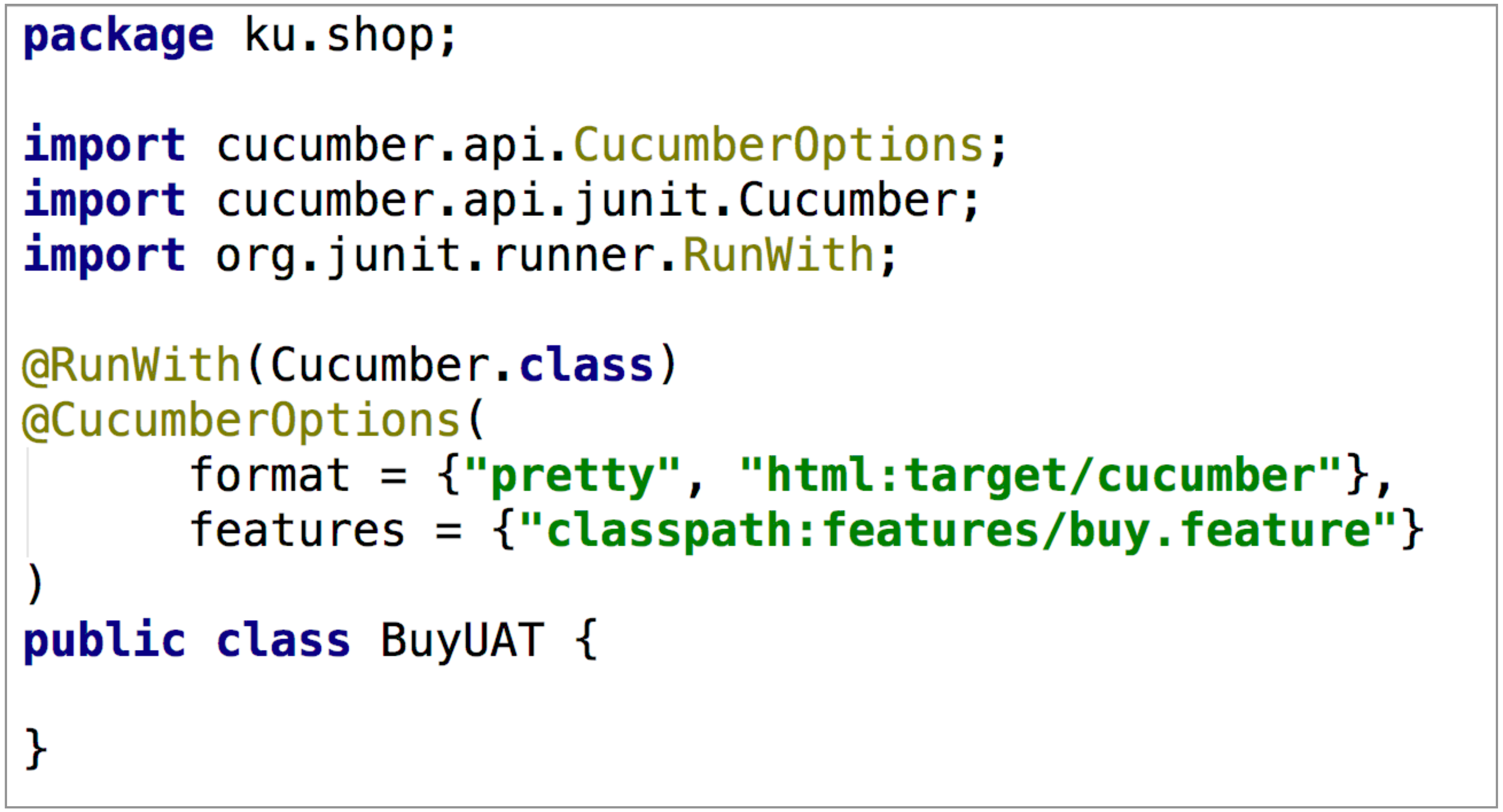
Simple scenario

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

Scenario: Buy one product
    Given a product Bread with price 20.50 exists
    When I buy Bread with quantity 2
    Then total should be 41.00
```

```java
public class BuyStepdefs {

    private ProductCatalog catalog;
    private Order order;

    @Before
    public void setup() {
        catalog = new ProductCatalog();
        order = new Order();
    }

    @Given("a product (.+) with price (.+) exists")
    public void a_product_with_price_exists(String name, double price) {
        catalog.addProduct(name, price);
    }

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

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

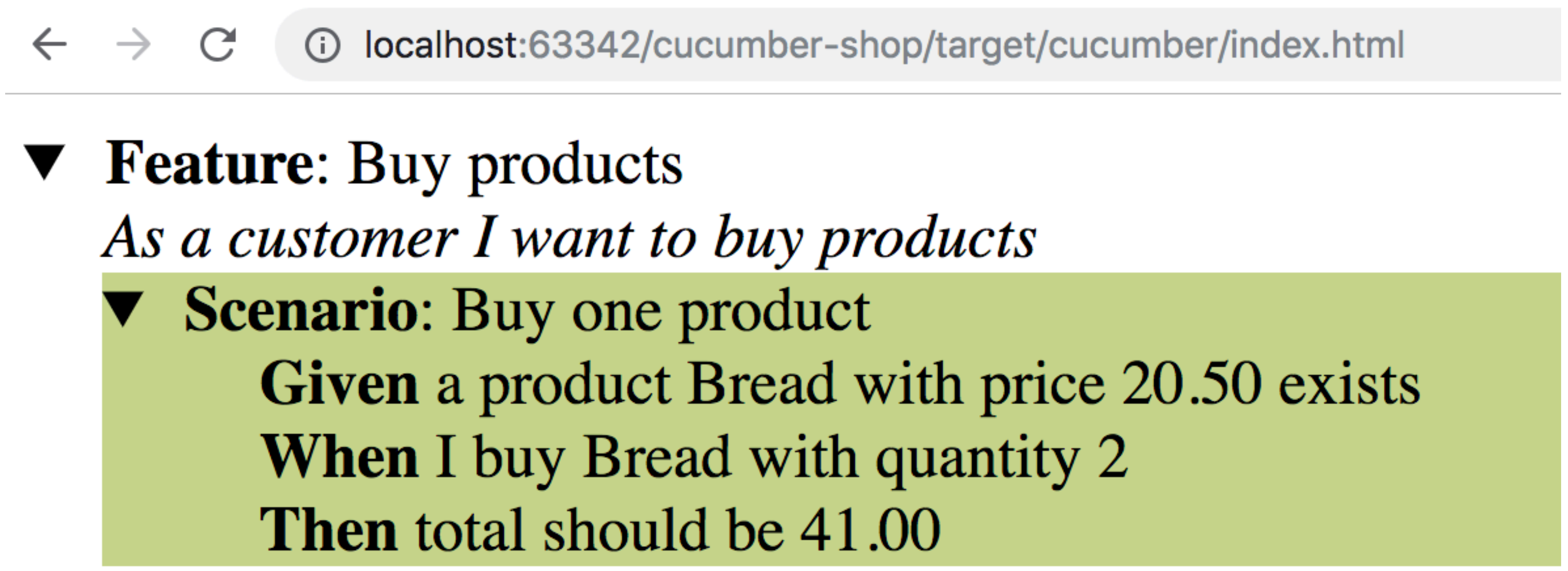
นิยามประโยคใน Given

Test runner in jUnit



```java
package ku.shop;

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

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

}
```

Test results


localhost:63342/cucumber-shop/target/cucumber/index.html

▼ **Feature**: Buy products

As a customer I want to buy products

▼ **Scenario**: Buy one product

Given a product Bread with price 20.50 exists

When I buy Bread with quantity 2

Then total should be 41.00

Example code

- https://github.com/ladyusa/cucumber-shop
- https://github.com/ladyusa/cucumber-atm