ENDLICH DIESELBE SPRACHE SPRECHEN!

MIT JGIVEN DAS SYSTEMVERHALTEN FÜR **ALLE** VERSTÄNDLICH DOKUMENTIEREN

NooK Lübeck, 04.11.2023

Johannes Thorn

JOHANNES THORN

- Senior Softwareentwickler bei eCube
- Kreativer Problemlöser im Projekt

Schwerpunkte

- Java-Technologien
- Tools für EntwickeInde
- Leichtgewichtige Architekturarbeit

SOFTWAREENTWICKLUNG HEUTE

1. Anforderungen sammeln
2. Code schreiben
3. ????
4. **PROFIT!!**

KOMPLEXE FACH-LOGIK

Wähle das Produkt aus, das nach allen bekannten Regeln in diesem Auftrag das richtige ist.

PROBLEM

> *Wir trauen dem System nicht.*
>
> — Eine ungenannte Fachabteilung

ZIEL

- Gemeinsame Sprache zwischen EntwickelInden und Fachmenschen
- Verständliche Testergebnisse für den Fachbereich

FEEDBACK VON DER FACHABTEILUNG

> *Ihr habt da einen Fehler im Produkt-Entscheider in Code-Zeile 217.*

REALISTISCHES FEEDBACK

> *Wenn ich da drücke, funktioniert das Programm nicht.*

TATSÄCHLICHES FEEDBACK

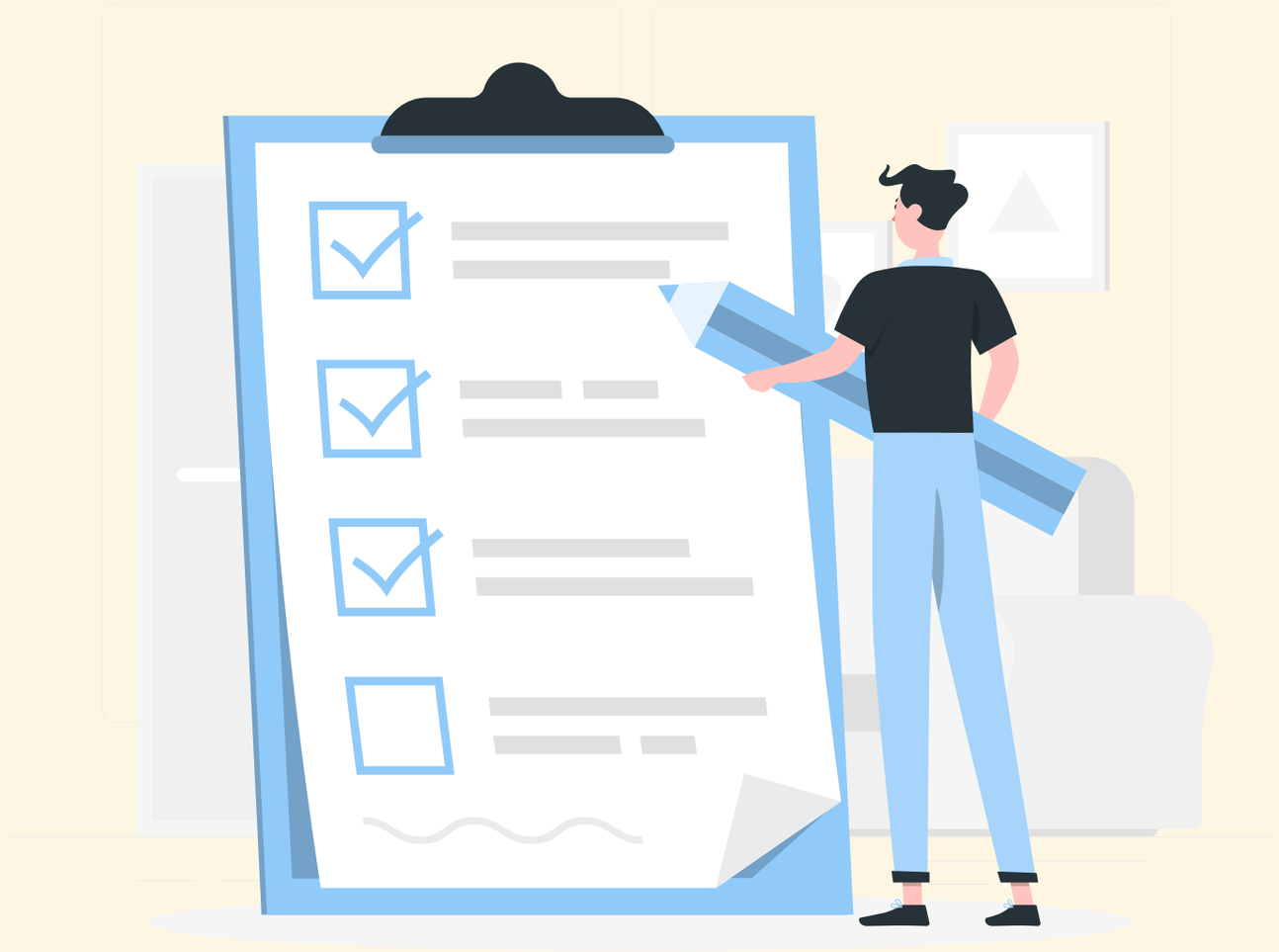
In Beispiel P-17 wird das falsche Produkt X als Ergebnis ausgewählt. Stattdessen soll das Produkt Y mit dem günstigsten Preis gewählt werden.

BEHAVIOR DRIVEN DEVELOPMENT* (BDD)

vs. Test Driven Development

ZIELE

1. Enge Zusammenarbeit
    - zwischen (Fach-)Menschen
    - und (Entwicklungs-)Menschen
2. gemeinsames Verständnis
3. Fallbeispiele als lebendige Dokumentation



MENGE VON BEISPIELEN/SZENARIEN

```
Gegeben eine Kundin
Wenn die Kundin eine Bestellung aufgibt
Dann wird ein Paket versendet

Gegeben eine Kundin
Und eine Bestellung
Wenn die Kundin die Bestellung storniert
Dann wird kein Paket versendet
```

GHERKIN-SYNTAX

- Given
- When
- Then
- And
- …

- Gegeben
- Wenn
- Dann
- Und
- …

CUCUMBER

- Szenarien in Plaintext `.feature` Files
- Glue-Code verknüpft Szenarien und Software

JGIVEN

Pragmatisches BDD in Java

JGIVEN BASICS

- Einfaches BDD in Java
- Sprechende API für Szenarien
- (Fach-)Menschenlesbare Testberichte

EINFACHES SZENARIO

```
1  class OrderManagementTest extends SimpleScenarioTest<OrderSteps> {
2      @Test
3      void an_order_is_placed() {
4          given().a_customer("Sandra");
5
6          when().the_customer_places_an_order();
7
8          then().a_package_is_sent();
9      }
10 }
```

PLAINTEXT BERICHT

```
Scenario: an order is placed

  Given a customer Sandra
   When the customer places an order
   Then a package is sent
```

GLUE-CODE???

```
1  public class OrderSteps extends Stage<OrderSteps> {
2
3      private Customer customer;
4
5      public OrderSteps a_customer(String name) {
6          this.customer = new Customer(name);
7          return self();
8      }
9
10     // ...
11
12 }
```

GLUE-CODE???

```
 1 public class OrderSteps extends Stage<OrderSteps> {
 2
 3     private Customer customer;
 4
 5     // ...
 6
 7     public OrderSteps a_package_is_sent() {
 8         final Set<Package> sentPackages =
 9                 shippingManager.getSentPackages(customer);
10         assertFalse(sentPackages.isEmpty());
11         return self();
12     }
13 }
```

STEPS

Definition

```
public class OrderSteps extends Stage<OrderSteps> {
    public OrderSteps there_are_$_packages_in_stock(int count) {
        return self();
    }
}
```

Verwendung

```
given().there_are_$_packages_in_stock(8);
```

Ergebnis

```
Given there are 8 packages in stock
```

FORMATIERUNG DER PARAMETER

1. Anpassen von `toString()`
2. Wrapper-Klassen mit eigener String-Repräsentation
3. Spezielle Annotationen am Parameter
   - `@Format(value=BooleanFormatter.class, args={…})`
   - `@POJOFormat`
   - …

TABELLEN ALS PARAMETER

```
1 public class GivenProducts extends Stage<GivenProducts> {
2     public GivenProducts the_products_in_stock(
3             @Table ProductWithPrice... products ) {
4     }
5 }
```

Given the products in stock

Product	Price
carrots	5
potatoes	10

STRUKTURIERTE SZENARIEN

```
1 class PanCakeExamples extends
2         ScenarioTest<GivenIngredients, WhenCook, ThenMeal> {
3     // ...
4 }
```

STRUKTURIERTE STAGES

```java
public class GivenIngredients extends Stage<GivenIngredients> {
    @ScenarioState
    List<String> ingredients = new ArrayList<String>();

    public GivenIngredients some_ingredients(String... ingredient) {
        ingredients.addAll( ingredient );
        return self();
    }
}
```

STRUKTURIERTE STAGES

```
1 public class WhenCook extends Stage<WhenCook> {
2     @ScenarioState
3     List<String> ingredients;
4
5     public WhenCook the_cook_uses_the_ingredients() {
6         assertThat(ingredients).isNotEmpty();
7         return self();
8     }
9 }
```

DEUTSCH UND ANDERE SPRACHEN

```
1  class DeSzenarioTest extends EinfacherSzenarioTest<DeutscheTestStufe> {
2
3      @Test
4      void Szenarien_können_in_deutsch_geschrieben_werden() {
5          gegeben().ein_deutsches_Projekt();
6          wenn().JGiven_verwendet_wird()
7              .und().die_Szenarien_in_deutsch_geschrieben_werden();
8          dann().generiert_JGiven_deutsche_Berichte();
9      }
10 }
```

MENSCHENLESBARE BERICHTE

HTML / AsciiDoc / Plaintext (/ JSON ☺)

PLAINTEXT U. JSON

automatisch während der Testausführung

ANDERE FORMATE

Report-Generator

- Standalone
- Maven-Plugin
- Gradle-Plugin

HTML REPORT

- dynamische HTML-Seite
- viele Interaktionsmöglichkeiten

82
7
3
Value
gory
pose
res
s
tep
les
Annotation
CustomStyle

Data Tables

In order to get a better overview over the different cases of a scenario
As a human,
I want to have different cases represented as a data table

2 Successful, 0 Failed, 0 Pending, 2 Total (24ms)

Group By Sort By Status Tags

Serve Coffee Correct messages are shown 5 (13ms)

Data Tables

Given a coffee machine
And there are **<coffees left>** coffees left in the machine
When I insert **<number of coins>** one euro coins
And I press the coffee button
Then the message **<message>** is shown

Cases

Group by

#	coffees left	number of coins	message	Status
1	0	0	Error: No coffees left	✓
2	0	1	Error: No coffees left	✓
3	1	0	Error: Insufficient money	✓
4	0	5	Error: No coffees left	✓
5	1	5	Enjoy your coffee!	✓

com.tngtech.jgiven.examples.coffeemachine.ServeCoffeeTest

Serve Coffee Serving a coffee reduces the number of available coffees by one 3 (10ms)

Data Tables

FUNKTIONEN

- Teilmengen für Szenarien
  - nach erfolgreich / fehlerhaft / in Vorbereitung
  - nach Tags
- Filtern / Sortieren / Gruppieren
- Attachments (Bilder, Hyperlinks, …)
- …

ASCIIDOC REPORT

- statisches Dokument
- AsciiDoc als Basis für **DocsAsCode**

Correct messages are shown

✅ (13ms)

Tags: *Data Tables*

Given a coffee machine
➕ *An empty coffee machine that is already turned on.*
The coffee price is set to 2 EUR.

And there are <**coffees left**> coffees left in the machine
➕ *The number of coffees in the machine is set to the given value.*

When I insert <**number of coins**> one euro coins

And I press the coffee button

Then the message <**message**> is shown

Table 14. Cases

#	coffees left	number of coins	message	Sta
1	0	0	Error: No coffees left	✅ (
2	0	1	Error: No coffees left	✅ (
3	1	0	Error: Insufficient money	✅ (

FORTGESCHRITTENE VERWENDUNG

UNVOLLSTÄNDIGE IMPLEMENTIERUNG

```
@Pending
public ThenResult the_result_is_$( int result ) {
    // add assertions
    return self();
}
```

MEHR BESCHREIBUNG

Definition

```
@As( "$ % are added" )
public WhenCalculator $_percent_are_added( int percent ) {
    return self();
}
```

Verwendung

```
when().$_percent_are_added(10);
```

Ergebnis

```
When 10 % are added
```

NOCH MEHR BESCHREIBUNG

Definition

```
@ExtendedDescription( "Der Kontext ist komplex, weil ..." )
public OrderSteps ein_komplexer_Kontext() {
    return self();
}
```

Verwendung

```
gegeben().ein_komplexer_Kontext();
```

Ergebnis

```
Gegeben ein komplexer Kontext [Der Kontext ist komplexe, weil ...]
```

TECHNISCHE DETAILS

```
public GivenRocket a_rocket(@Hidden int rocketId) {
    rocket = Rocket.create(rocketId);
    return self();
}
```

```
@Hidden
public GivenRocket prepareRocketSimulator() {
    rocketSimulator = createRocketSimulator();
    return self();
}
```

ATTACHMENTS

```
1  public class PageDesignSteps extends Stage<PageDesignSteps> {
2      @ExpectedScenarioState
3      CurrentStep currentStep;
4
5      public SELF the_page_adheres_to_the_design() {
6          String base64 = // ...
7
8          currentStep.addAttachment(
9              Attachment.fromBase64( base64, MediaType.PNG )
10                 .withTitle("Screenshot")
11         );
12
13         // ...
14         return self();
15     }
16 }
```

PARAMETRISIERTE SZENARIEN

```
Coffee is not served

   Given a coffee machine
     And the coffee costs 2 euros
     And there are <coffees> coffees left in the machine
    When I insert <euros> one euro coins
     And I press the coffee button
    Then I should not be served a coffee

  Cases:

   | # | coffees | euros | Status  |
   +---+---------+-------+---------+
   | 1 |       1 |     1 | Success |
   | 2 |       0 |     2 | Success |
   | 3 |       1 |     0 | Success |
```

LIFECYCLE-METHODEN

- `@BeforeScenario` vor allen Steps
- `@BeforeStage` vor Steps einer Stage
- `@AfterStage` nach Steps einer Stage
- `@AfterScenario` nach allen Steps

WEITERE FEATURES

- Verschachtelte Steps gruppiert `@NestedSteps`
- Hierarchische `@Tags`
- Neue Hilfswörter
  - Intro-Words
  - Filler-Words
- Vererbung zwischen Stages
- Mehr als drei Stages durch Injection
  - `@ScenarioStage`
  - dynamisch `addStage`

WEITERE INTEGRATIONEN

Spock / Spring / Selenium / Android

SPOCK

```
class SpockSpec extends ScenarioSpec<Given, When, Then> {
    def "my scenario"() {
        expect:

        given().some_context()
        when().some_action()
        then().some_outcome()
    }
}
```

SPRING

- `@EnableJGiven`
- `SpringScenarioTest` und `SimpleSpringScenarioTest`
- `@JGivenStage`

SELENIUM

`PageObjects` ohne zusätzliche Abhängigkeiten

ANDROID

- Testausführung auf dem Gerät selbst
- JSON-Reports werden heruntergeladen

VIELEN DANK.

Ich freue mich auf eure Fragen!

Kontakt und Fragen

★ @johthor@social.chaotikum.org
✉ Johannes.Thorn@eCube.de

FEEDBACK

Zum Vortrag / Zur Nook

https://2023.nook-luebeck.de/feedback/

APPENDIX A: QUELLEN

- JGiven User Guide - https://jgiven.org/userguide/
- Font Awesome by Dave Gandy - http://fontawesome.io
- People illustrations by Storyset - https://storyset.com/people
- Work illustrations by Storyset - https://storyset.com/work