Behaviour-Driven Development

If the system's not behaving, what is it doing?

Luigi Libero Lucio STARACE
luigiliberolucio.starace@unina.it

May 5, 2020
University of Naples, Federico II

Behaviour-Driven Development (BDD)

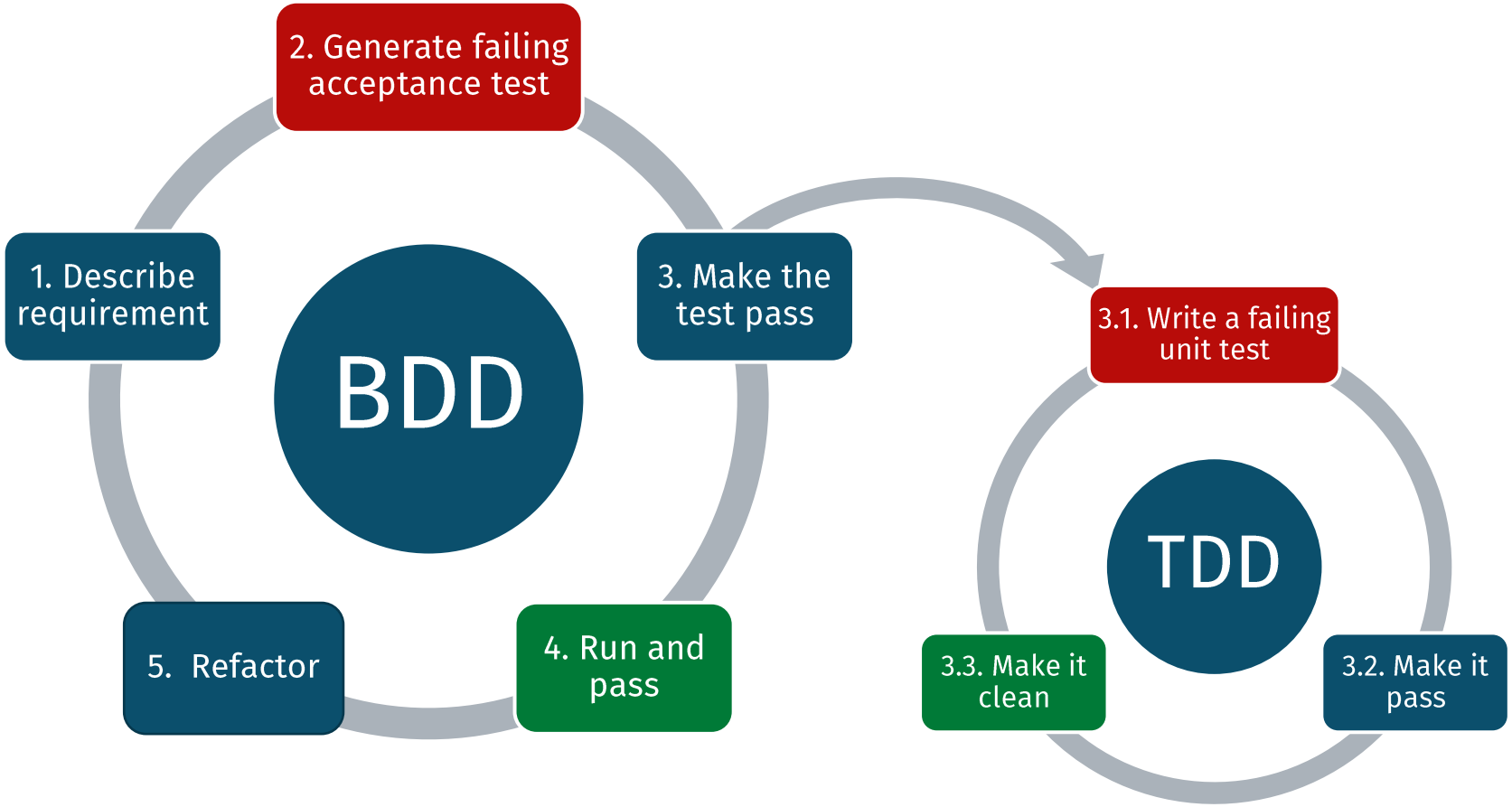
Like TDD, Behaviour-Driven Development is a *test-first* approach

- The focus is on testing the actual behaviour of the system from the end users perspective

BDD – Motivations

- Sometimes TDD can make developers too detail-focused, losing the broader picture of the business goals that really matter.
- Writing acceptance tests is not easy
  - Where do we start with TDD? When are we done?
- Risks of misunderstandings between stakeholders

The BDD process



Behaviour-Driven Development

Requirements are expressed using an ubiquitous, shared language that

- Is easily understood by business owners as well as engineers
- Can unambiguously define requirements, by defining example scenarios
- Can also be easily transformed into executable acceptance tests
- … thus creating living, executable specifications

BDD – Describing Requirements

- Three amigos meet (Customer, Developers and QA)
- Often, some form of structured natural language is used
- Given, When, Then
  - Given I am a registered user
  - When I insert my credentials correctly in the login form
  - Then I should be redirected to my personal page
- Many tools help describing requirements and generating acceptance tests:
  - Cucumber, Gauge, SpecFlow, JBehave, PHPSpec, …
  - Often supported by automation libraries like Selenium, Espresso, …

TDD vs BDD

TDD

- Doing the thing right
- Involves developers
- Unit tests

BDD

- Doing the right thing
- Involves developers, customers, and QA
- Acceptance tests

Introducing Cucumber

Cucumber

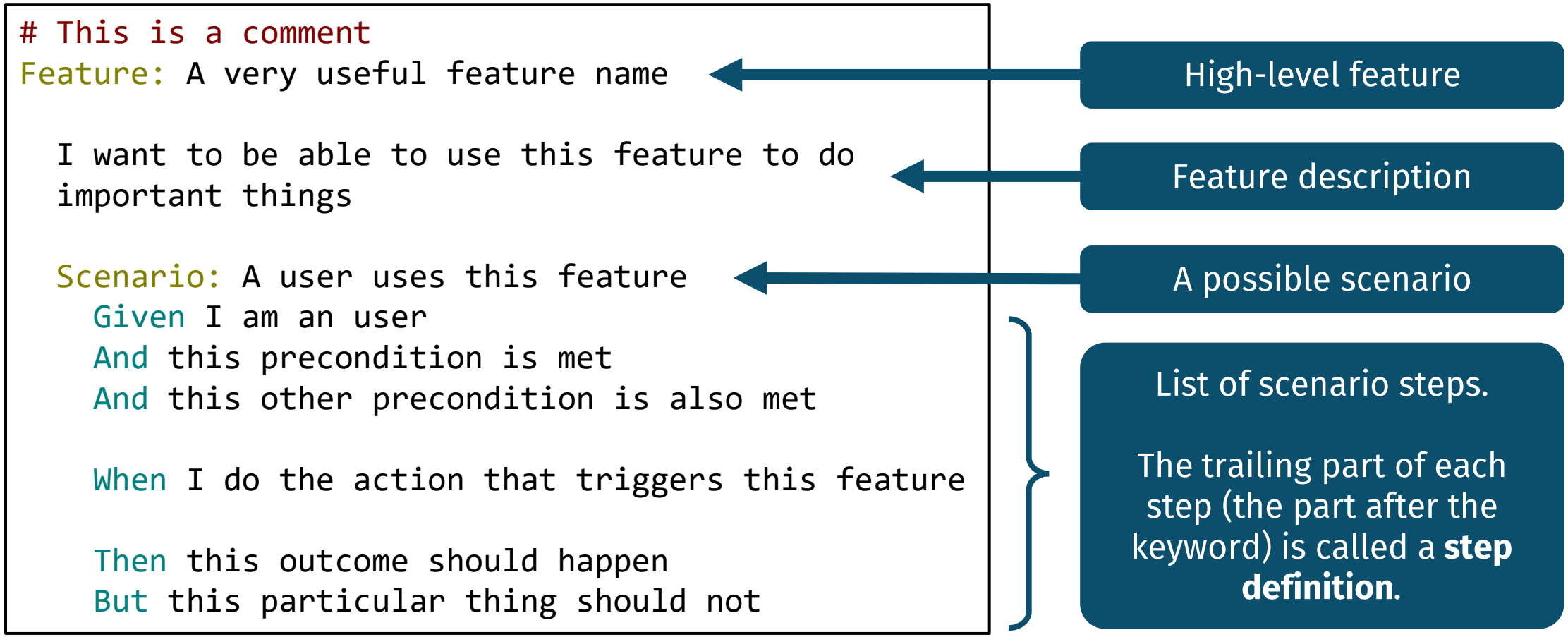
- A tool supporting BDD
- Originally for Ruby, now supports Java, Javascript, Kotlin
- Introduced **Gherkin**, a set of grammar rules that make plain text structured enough for testing automation

Gherkin

- A language to write specifications
- A Gherkin specification is a plain text file, called feature file
- A set of keywords gives structure and meaning to natural language specifications

Gherkin – A first example

```
# This is a comment
Feature: A very useful feature name

  I want to be able to use this feature to do
  important things

  Scenario: A user uses this feature
    Given I am an user
    And this precondition is met
    And this other precondition is also met

    When I do the action that triggers this feature

    Then this outcome should happen
    But this particular thing should not
```



Gherkin – A first example

```
# This is a comment
Feature: A very useful feature name

  I want to be able to use this feature to do
  important things

  Scenario: A user uses this feature
    Given I am an user
    * this precondition is met
    * this other precondition is also met

    When I do the action that triggers this feature

    Then this outcome should happen
    * this particular thing should not
```

High-level feature

Feature description

A possible scenario

List of scenario steps.

The trailing part of each step (the part after the keyword) is called a **step definition**.

Gherkin – A realistic example

```
Feature: After login redirection
  I want to be redirected to my homepage after login

  Scenario: A user is redirected to his/her homepage
    Given I am a registered user
    When I visit the login page
    And I insert the correct credentials in the form
    Then I should be redirected to my personal page

  Scenario: A VIP user is redirected to his/her VIP homepage
    Given I am a registered user
    And I purchased a VIP pass
    When I visit the login page
    And I insert the correct credentials in the form
    Then I should be redirected to my VIP personal page
    But I should see no advertisements
```

Gherkin – Parametric scenarios

```
Feature: Generate greetings

  Scenario Outline: Get an appropriate greeting
    Given that my name is <name>
    When I request a greeting for my name
    Then I should get a response with HTTP status <status>
    And the response should contain the message <message>

  Examples:
    | name         | status | message               |
    | "Luigi"      | 200    | "Hello, Mr. Luigi!"   |
    | "Ada"        | 200    | "Hello, Mrs. Ada!"    |
    | "Liz Swann"  | 200    | "Hello, Miss Swann!"  |
    | "George W."  | 500    | "Invalid input!"      |
```

Gherkin – Spoken Languages

The language you choose for Gherkin should be the same your users and domain experts use when they talk about the domain

- Gherkin has been translated to more than 70 languages!
- A `# language:` header in the first line of a feature file tells Gherkin which language to use.

```
# language: it
Funzionalità: Prelievo contanti
  Voglio prelevare contanti dal mio conto

Scenario: Prelievo riuscito
  Dato che ho un conto
  E il conto ha un saldo di 75.0 €
  E non ho raggiunto il limite prelievi
  Quando richiedo di prelevare 50.0 €
  Allora il saldo del conto diventa 25.0 €
  E ricevo un SMS di avviso
```

Gherkin – Matching steps to code

- When executing Gherkin specifications, Cucumber tries to match each step definition to actual code (functions).
- The matched functions are then executed in the same order and with the same parameters as the corresponding steps in each scenario.
- This matching is done by annotating methods with regular expressions or Cucumber expressions that match one or more step definitions.

Gherkin – Matching steps to code

```
Feature: Calculator adds two integers
  I want to compute sums with my Calc

  Scenario Outline: Compute sums
    When I add <a> and <b>
    Then I verify that I get <sum>

    Examples:
      | a | b | sum |
      | 5 | 0 |  5  |
      | 0 | 7 |  7  |
      | 5 | 6 |  11 |
```

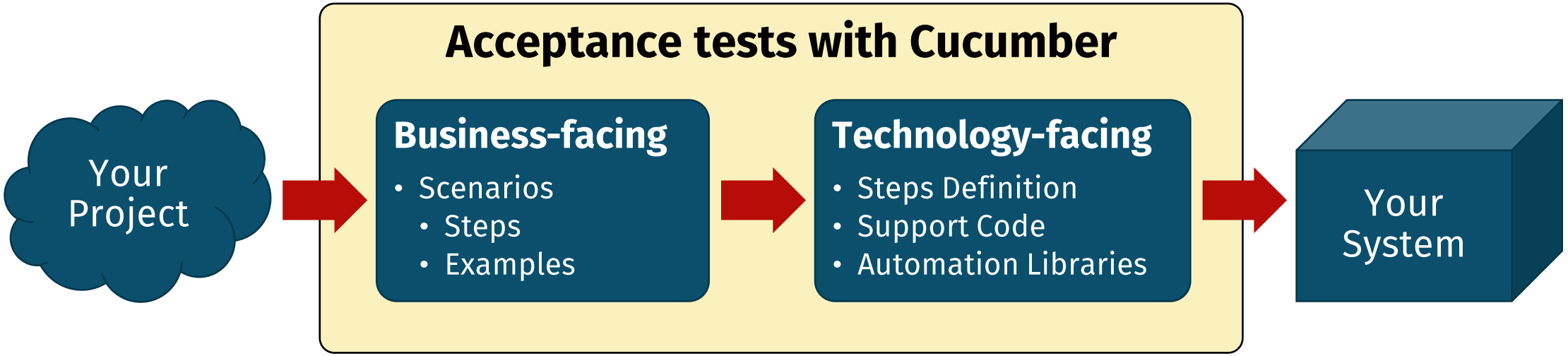
```
public class StepDefinitions {

  int sum;

  @When("I add {int} and {int}")
  public void i_add(int a, int b) {
    sum = Calculator.sum(a, b);
  }

  @Then("I verify that I get {int}")
  public void i_get(int expected) {
    assertEquals(expected,sum);
  }
}
```

Gherkin – Matching steps to code

```
Scenario: Withdrawal succeeds
  Given I own an Account
  And the account balance is 100.0 €
  When I withdraw 50.0 €
  Then I see that the balance is 50.0 €
  And I receive an SMS containing
    "You just withdrew 50 €"

Scenario: Withdrawal fails
  Given I have an Account
  And the account has 100.0 €
  When I withdraw 150.0 €
  Then I see an error message
    containing "Insufficient funds!"
```

```
@Given("I own/have an Account")
public void i_own_an_Account() {}

@Given("the account (balance )is/has {double} €")
public void the_account_balance_is_€(double b) {}

@When("I withdraw {double} €")
public void i_withdraw_€(double w) {}

@Then("I receive an SMS containing {string}")
public void i_receive_SMS_containing(String s) {}

@Then("I see an error message containing {string}")
public void i_see_error_containing(String s) {}

@Then("I see that balance is {double} €")
public void i_see_that_balance_is_€(double b) {}
```

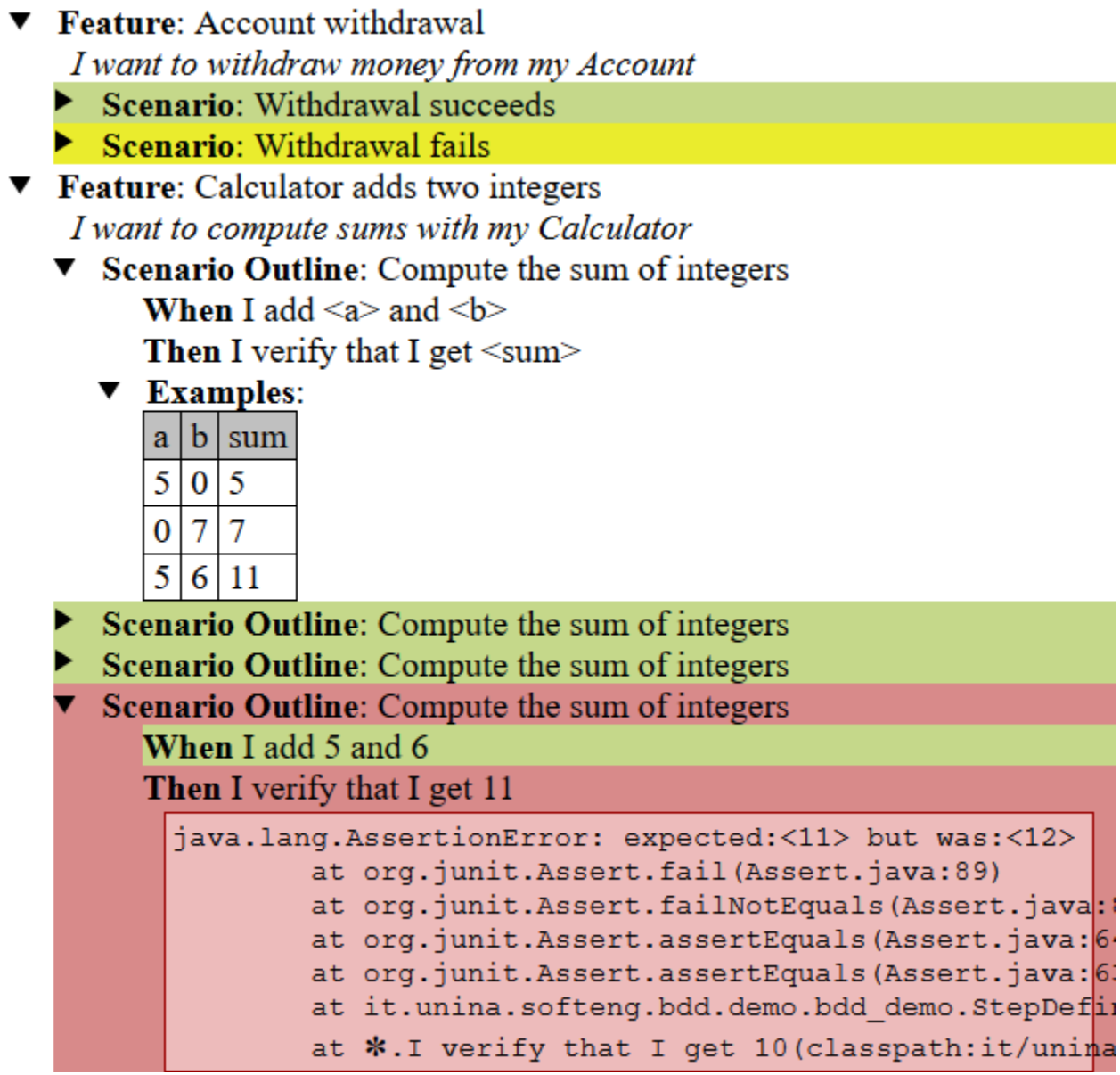
Cucumber – What does it do?



Cucumber – What does it do?

- You can run Cucumber from JUnit 4 (or JUnit 5 with the Vintage module), or from build tools like Maven.
- Cucumber automatically discovers .feature files from `/src/test/resources/` and subdirectories, and tries to run them.
- If no matching step definition is found for some steps, Cucumber can automatically generate stubs for these steps.

```
@Then("I see an error message containing {string}")
public void i_see_an_error_message_containing(String string) {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
```



Cucumber – Report

- Scenarios with missing steps are highlighted in yellow
- Passing scenarios in green
- And failing ones in red

References

- Freeman, Steve, and Nat Pryce. "*Growing object-oriented software, guided by tests*". Pearson Education, 2009.
- Martin, Robert C. "*Professionalism and test-driven development.*" *Ieee Software* 24.3 (2007): 32-36.
- Smart, John Ferguson. "*BDD in Action: Behavior-driven development for the whole software lifecycle*". Manning, 2015.
- Beck, Kent. "*Test-Driven Development: By Example*". Addison-Wesley Professional, 2002.