CSC
4104

Information System - BDD - Gherkin and Cucumber

Dr J Paul Gibson

Dept. INF

Office D311

paul.gibson@telecom-sudparis.eu

http://jpaulgibson.synology.me/~jpaulgibson/TSP/Teaching/CSC4104/CSC4104-BDD-GherkinCucumber.pdf

Behaviour-driven development (BDD)

What is **BDD** ?

What is **Gherkin**?

How to use/install a BDD tool (like **Cucumber**)

Use BDD for your “Presence en cours” case study

Behaviour-driven development

Introducing BDD

Test method names should be sentences ¶

A simple sentence template keeps test methods focused ¶

An expressive test name is helpful when a test fails ¶

“Behaviour” is a more useful word than “test” ¶

BDD tools emphasizes behaviour over testing ¶

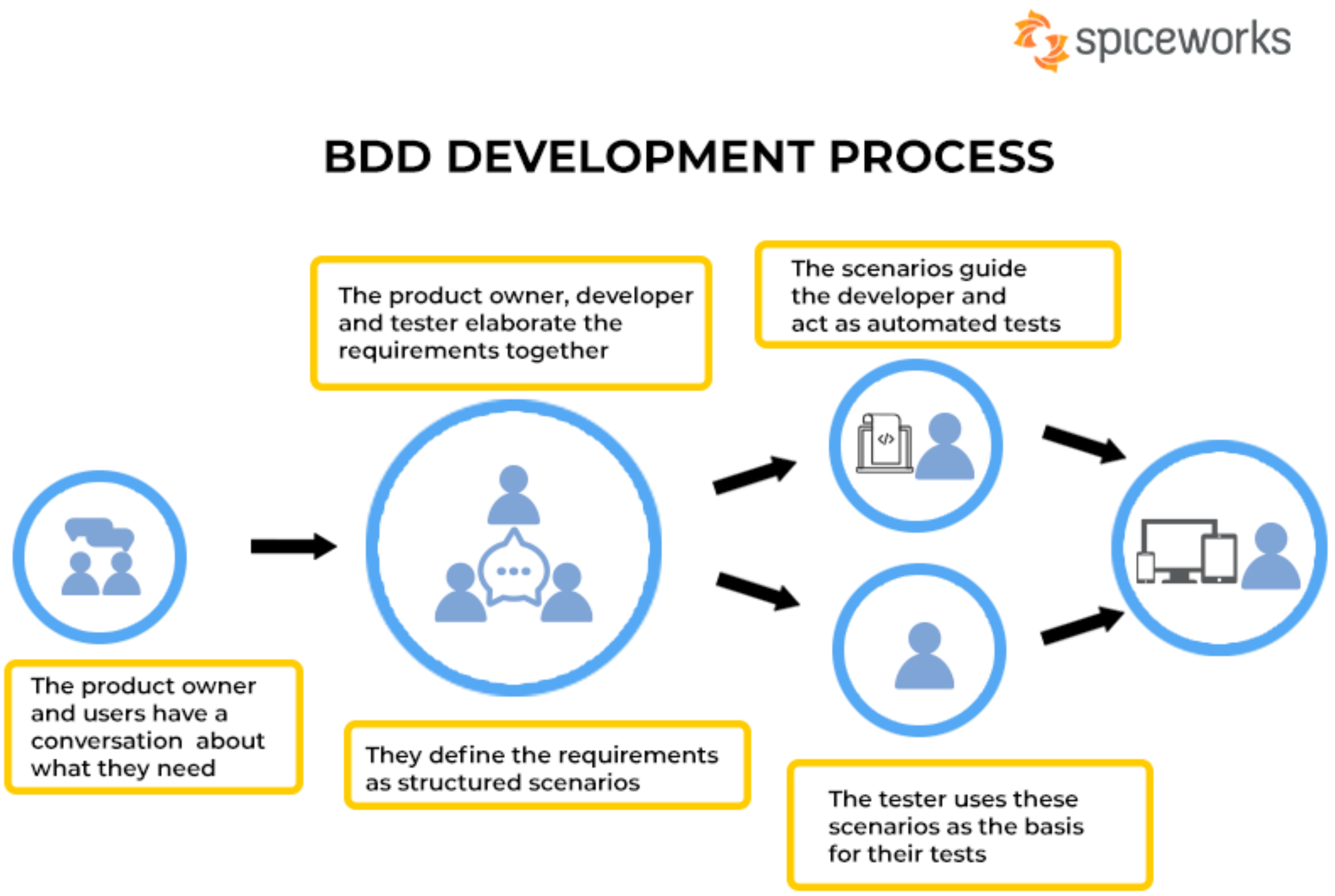
Determine the next most important behaviour ¶

Requirements are behaviour, too ¶

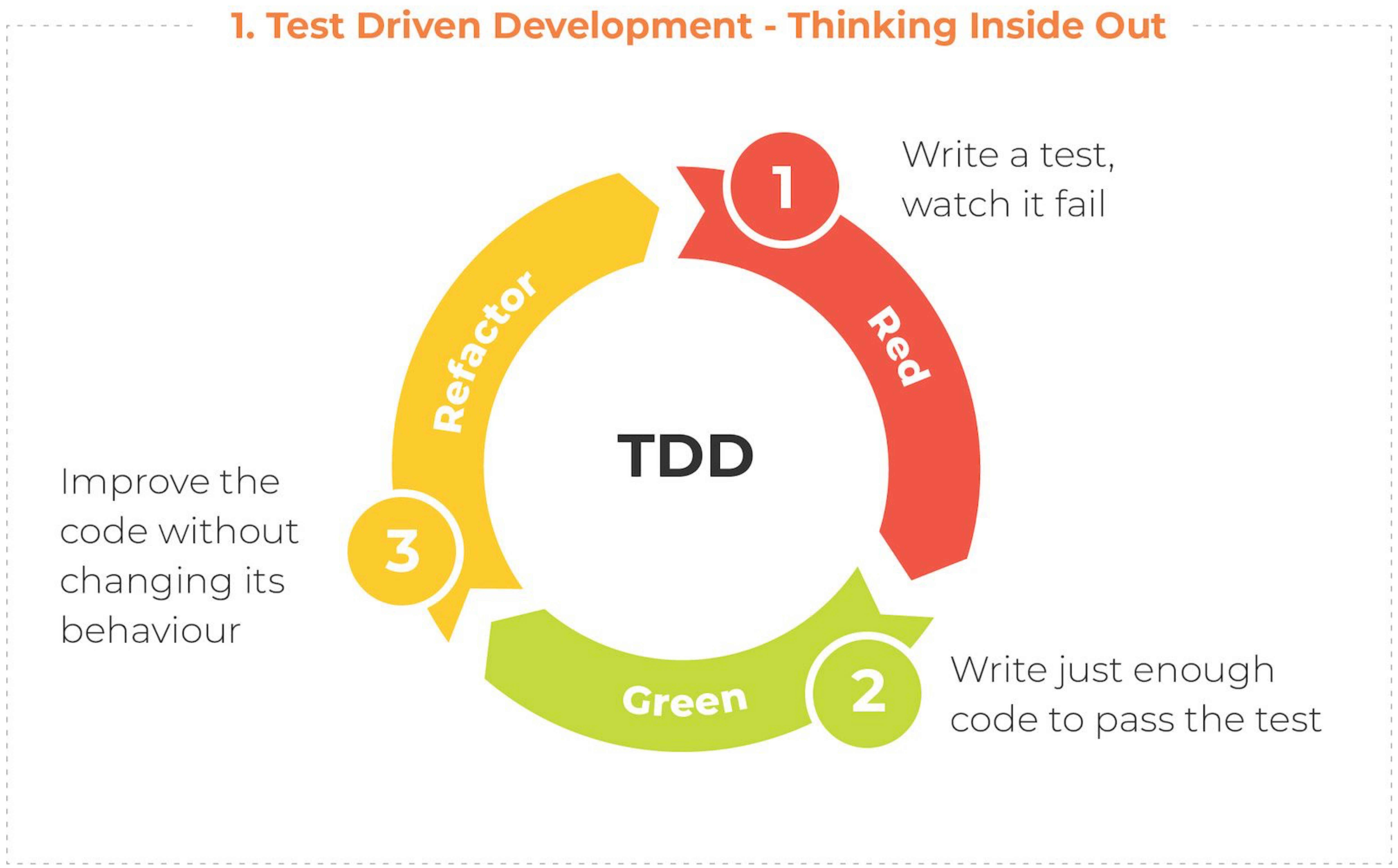
BDD provides a “ubiquitous language” for analysis ¶

Acceptance criteria should be executable ¶

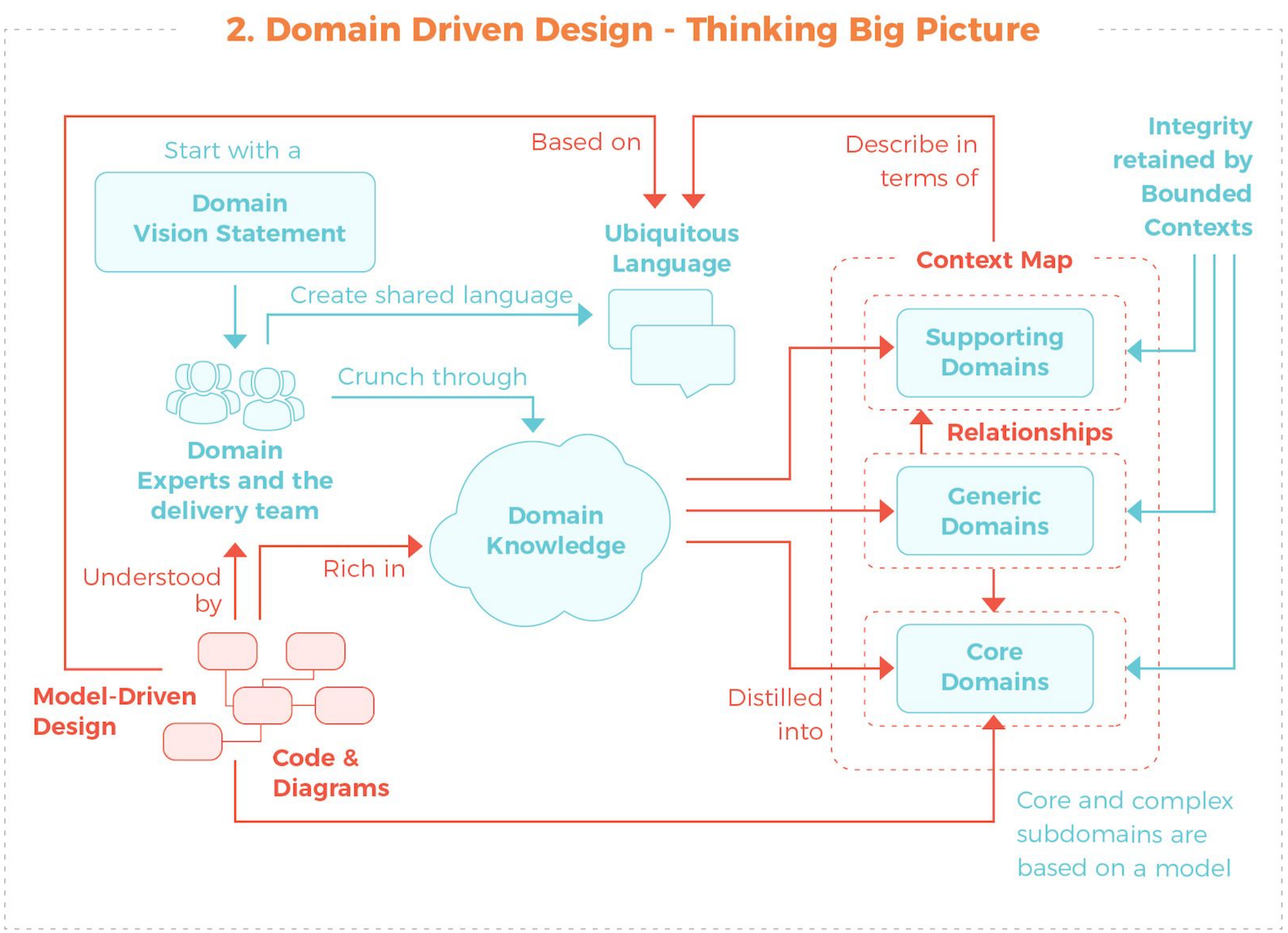
https://dannorth.net/introducing-bdd/


spiceworks
BDD DEVELOPMENT PROCESS
The product owner and users have a conversation about what they need
The product owner, developer and tester elaborate the requirements together
They define the requirements as structured scenarios
The scenarios guide the developer and act as automated tests
The tester uses these scenarios as the basis for their tests

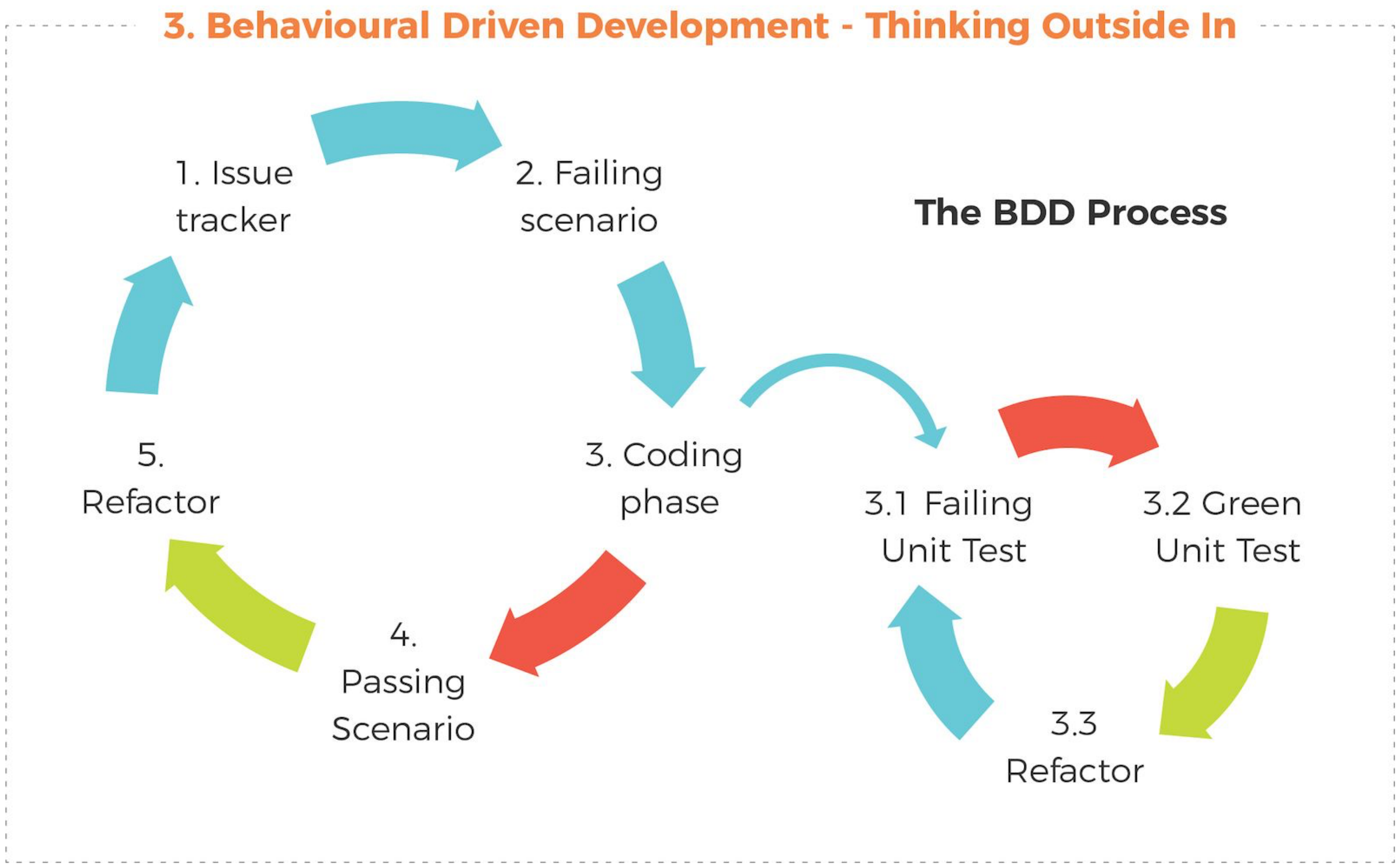
https://www.spiceworks.com/tech/devops/articles/what-is-bdd/


1. Test Driven Development - Thinking Inside Out
TDD
1
Red
Write a test, watch it fail
2
Green
Write just enough code to pass the test
3
Refactor
Improve the code without changing its behaviour

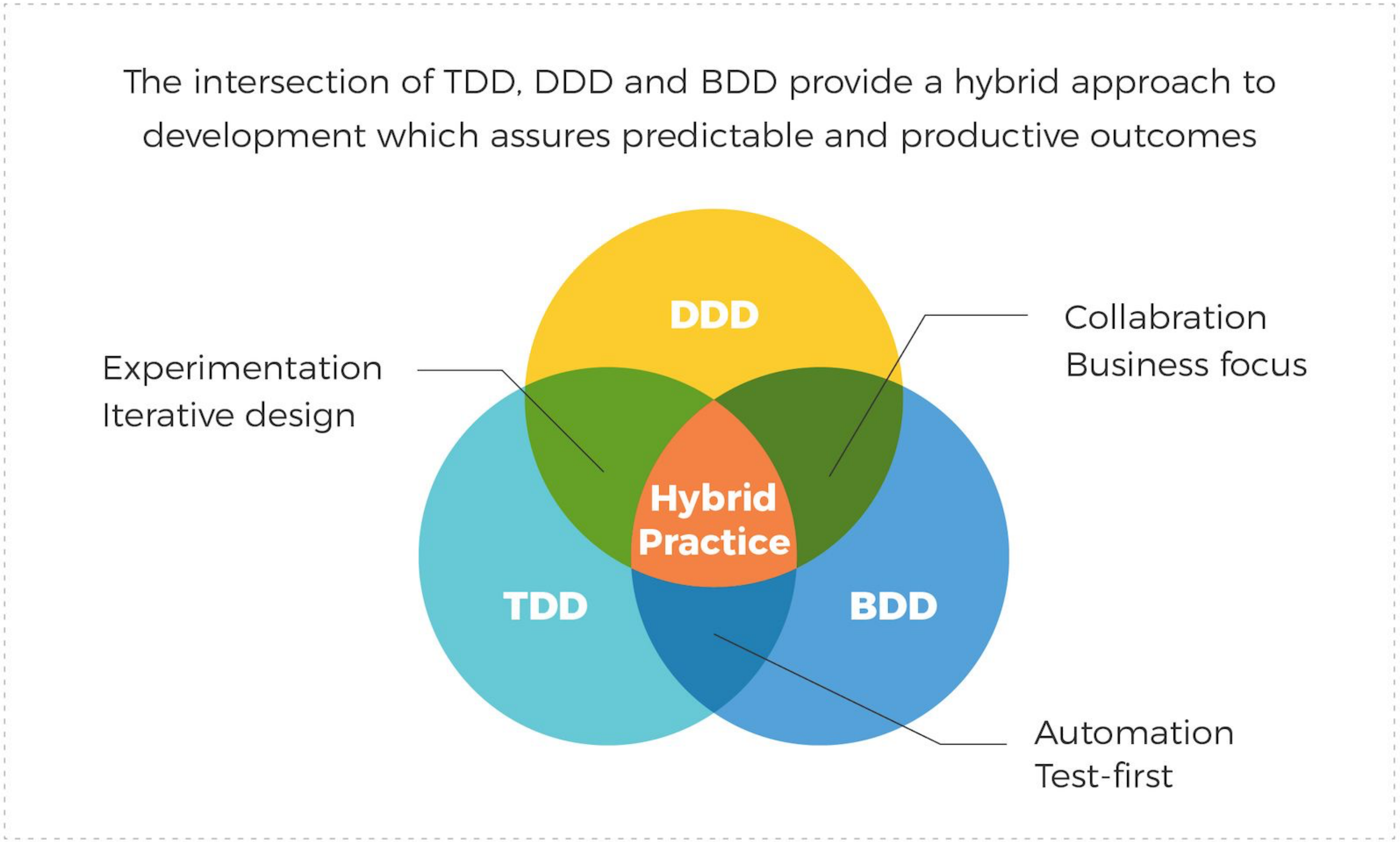
https://www.mobilelive.ca/blog/value-of-tdd-bdd-ddd



https://www.mobilelive.ca/blog/value-of-tdd-bdd-ddd



https://www.mobilelive.ca/blog/value-of-tdd-bdd-ddd


The intersection of TDD, DDD and BDD provide a hybrid approach to development which assures predictable and productive outcomes
DDD
Experimentation
Iterative design
Collabration
Business focus
Hybrid Practice
TDD
BDD
Automation
Test-first

https://www.mobilelive.ca/blog/value-of-tdd-bdd-ddd

Behaviour-driven development

Further reading

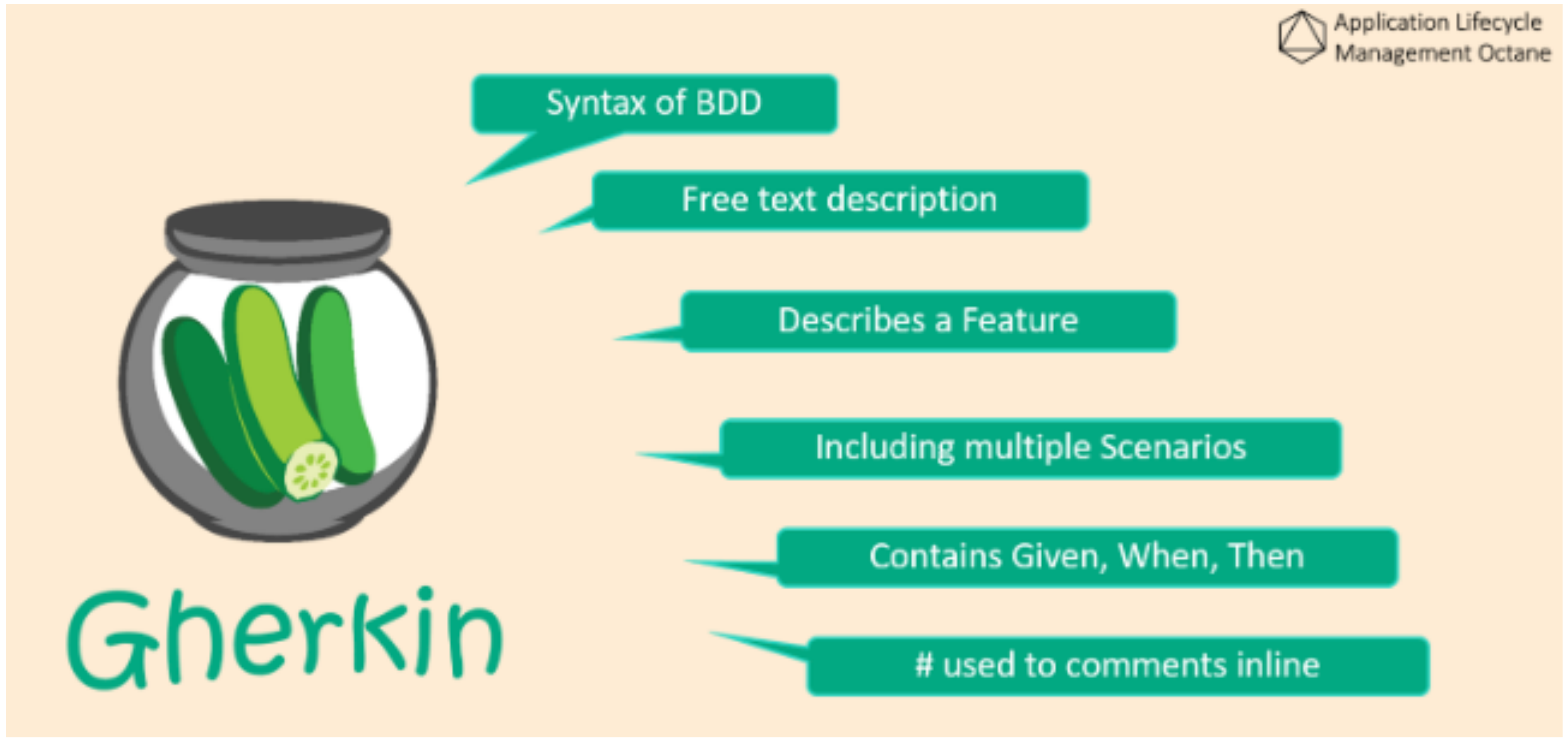
Solis, Carlos, and Xiaofeng Wang. "A study of the characteristics of behaviour driven development." *2011 37th EUROMICRO conference on software engineering and advanced applications*. IEEE, 2011.

Bruschi, Stefania, L. Xiao, and M. Kavatkar. "Behavior driven development (BDD): a case study in healthtech." Pacific NW Software Quality Conference. 2019.

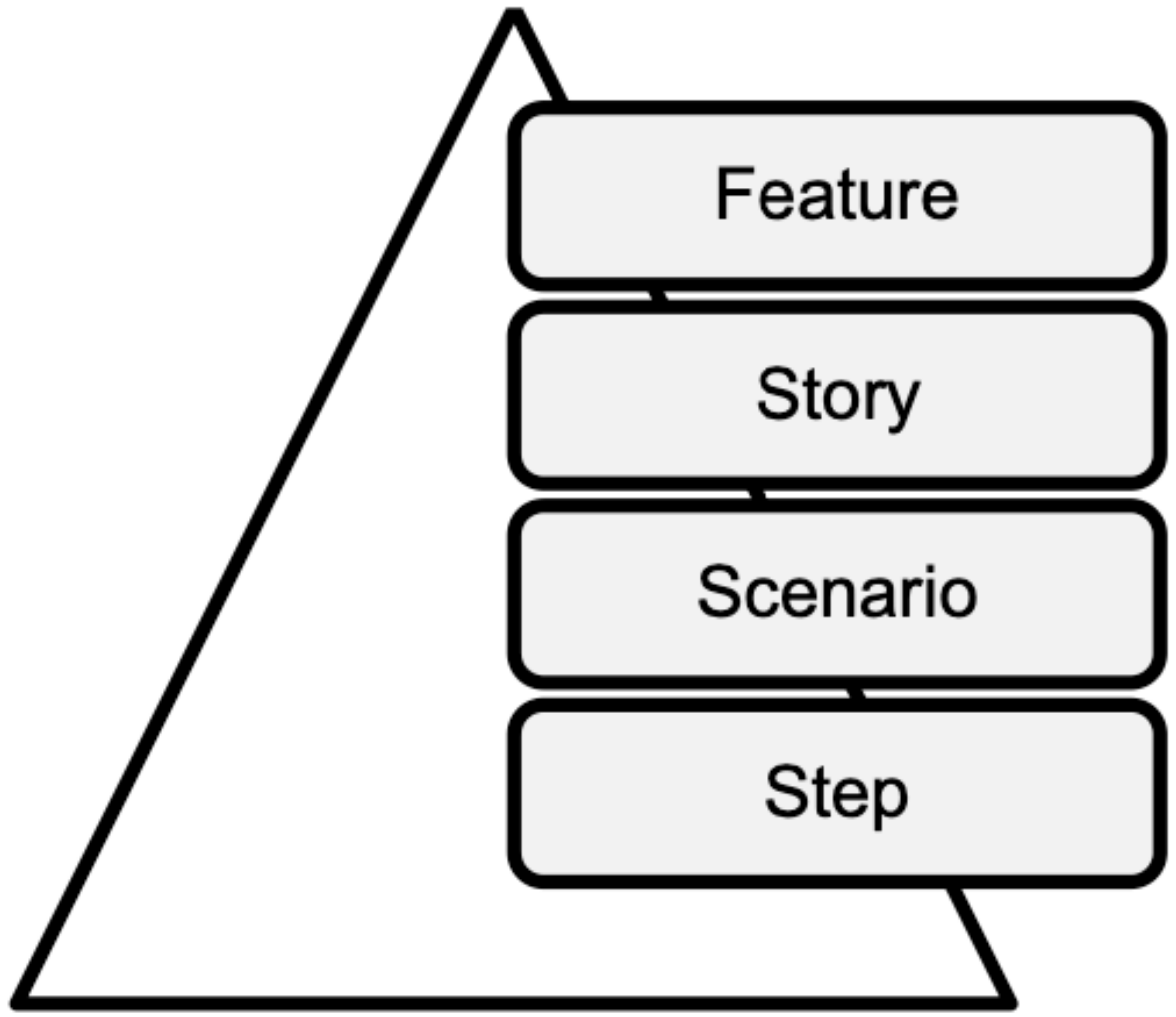
What is Gherkin?

Gherkin is a well-known syntax for writing tests using behaviour-driven development (BDD). This syntax lets you test from the user perspective by using use-case scenarios.

Gherkin syntax uses plain-text with a specific structure. Gherkin syntax is easy to learn but structured enough to allow specific examples.


Application Lifecycle Management Octane
Syntax of BDD
Free text description
Describes a Feature
Including multiple Scenarios
Contains Given, When, Then
# used to comments inline
Gherkin

What are Gherkin Features?



- Feature is a summary of new functionality to be developed and tested.
- Story is the end-user use case that needs to be addressed and the expected outcome is interpreted from the software or system requirements.
- Scenario is the basic unit of functionality which scopes out the user behaviours. A story could have many scenarios to test different outcomes.
- Step in a scenario is a simulation of user behavior which is verified against the expected behavior. There can be many steps to satisfy a test scenario.

Behavior driven development (BDD): a case study in healthtech, Stefania Bruschi, L. Xiao, and M. Kavatkar, 2019

Gherkin Scenarios and Use Cases

A scenario, in the context of use cases, represents one specific path or flow through a use case.

It narrates a sequence of events that unfold during a particular execution of the system.

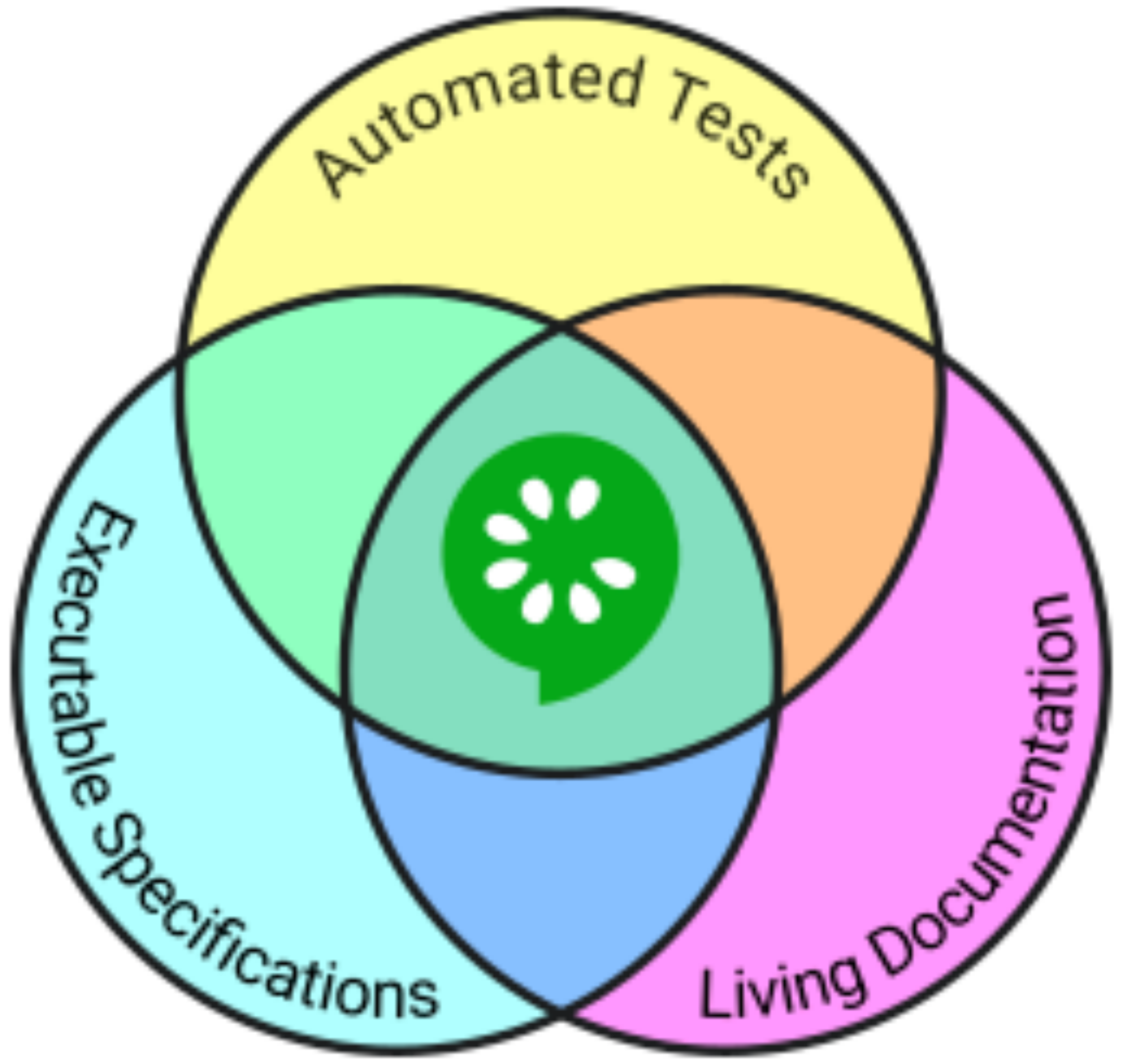
Scenarios provide a granular view of how the system behaves under different conditions, offering insights into the nuanced facets of its functionality.

What is Gherkin?

Gherkin is a set of grammar rules that makes plain text structured enough for a BDD tool like Cucumber) to understand.

Gherkin serves multiple purposes:

- Unambiguous executable specification
- Automated testing using Cucumber
- Document how the system *actually* behaves



https://cucumber.io/docs/guides/overview/

What is Gherkin?

The Cucumber grammar exists in different flavours for many spoken languages so that your team can use the keywords in your own language.

Gherkin documents are stored in `.feature` text files and are typically versioned in source control alongside the software.

See the Gherkin reference for more details.

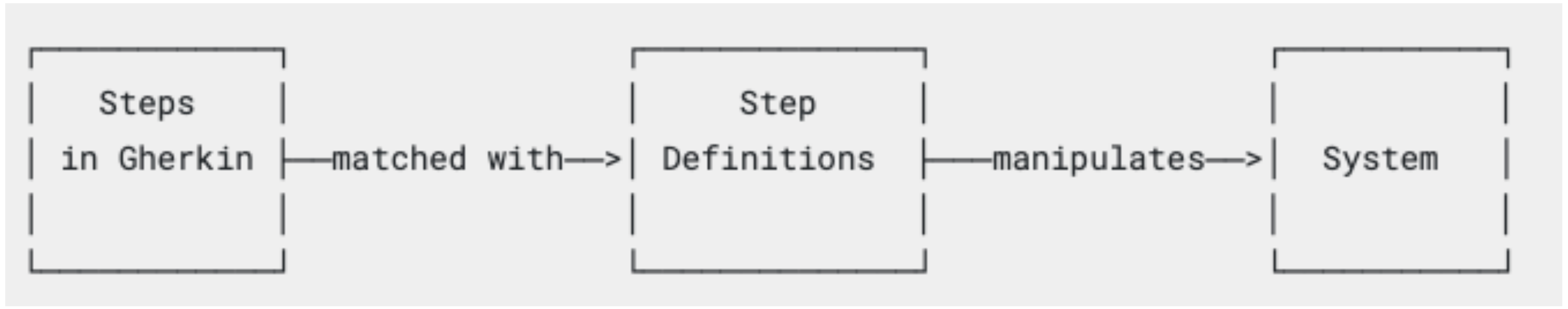
https://cucumber.io/docs/guides/overview/

What is Gherkin?

CSC 4104

Step definitions connect Gherkin steps to programming code. A step definition carries out the action that should be performed by the step.

So step definitions hard-wire the specification to the implementation.

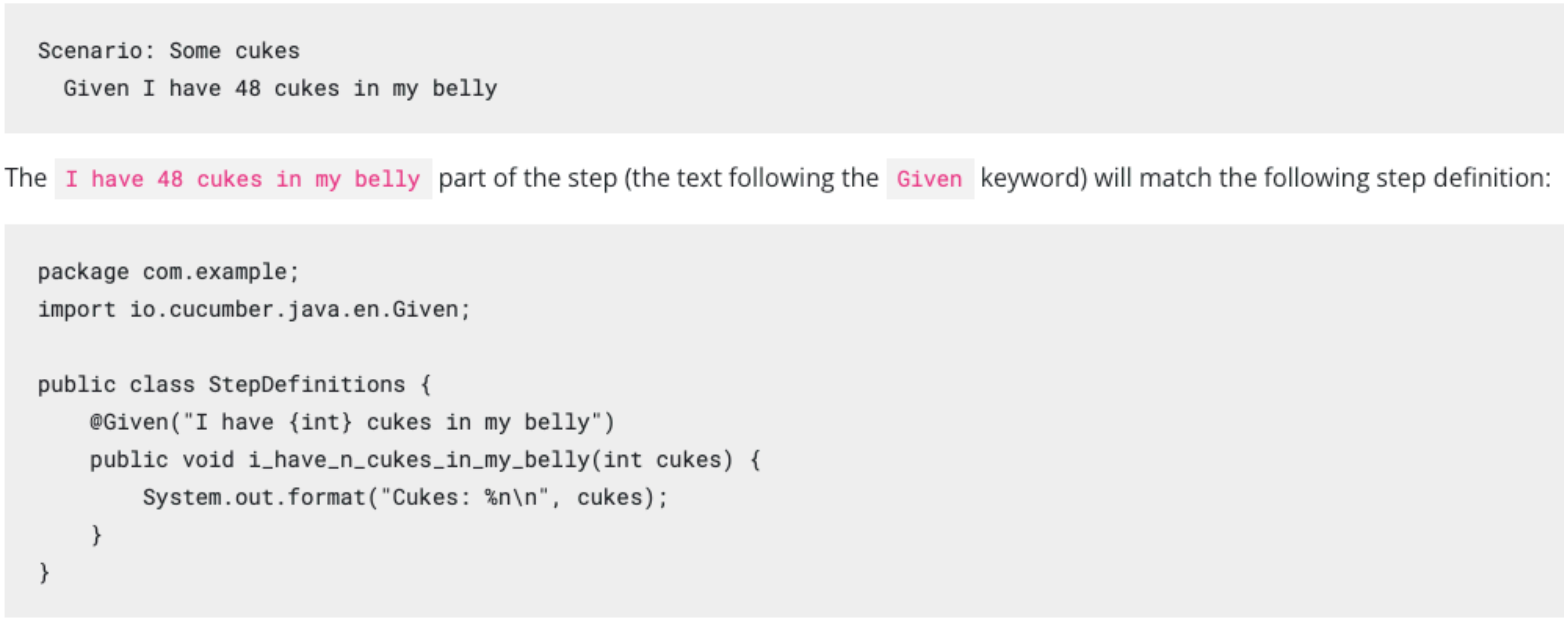


```
┌───────────────┐                 ┌───────────────┐                 ┌───────────────┐
│   Steps       │                 │     Step      │                 │               │
│ in Gherkin    ├──matched with──>│ Definitions   ├───manipulates──>│    System     │
│               │                 │               │                 │               │
└───────────────┘                 └───────────────┘                 └───────────────┘
```

https://cucumber.io/docs/guides/overview/

What is Gherkin?

Example step definition



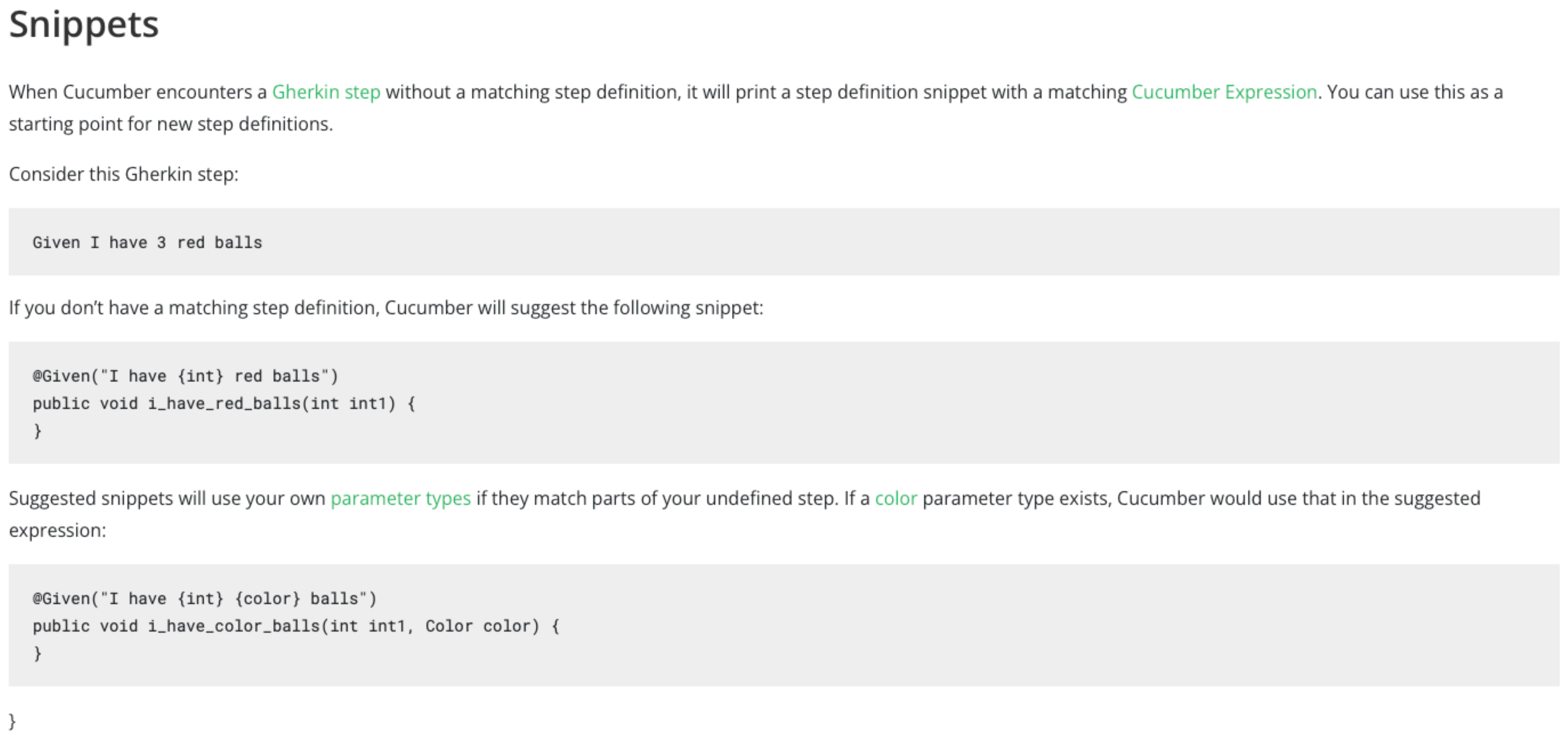
```
Scenario: Some cukes
  Given I have 48 cukes in my belly
```

The `I have 48 cukes in my belly` part of the step (the text following the `Given` keyword) will match the following step definition:

```
package com.example;
import io.cucumber.java.en.Given;

public class StepDefinitions {
    @Given("I have {int} cukes in my belly")
    public void i_have_n_cukes_in_my_belly(int cukes) {
        System.out.format("Cukes: %n\n", cukes);
    }
}
```

https://cucumber.io/docs/cucumber/step-definitions/?lang=java

What is Gherkin?



Snippets

When Cucumber encounters a Gherkin step without a matching step definition, it will print a step definition snippet with a matching Cucumber Expression. You can use this as a starting point for new step definitions.

Consider this Gherkin step:

```
Given I have 3 red balls
```

If you don't have a matching step definition, Cucumber will suggest the following snippet:

```
@Given("I have {int} red balls")
public void i_have_red_balls(int int1) {
}
```

Suggested snippets will use your own parameter types if they match parts of your undefined step. If a color parameter type exists, Cucumber would use that in the suggested expression:

```
@Given("I have {int} {color} balls")
public void i_have_color_balls(int int1, Color color) {
}
```

}

https://cucumber.io/docs/cucumber/step-definitions/?lang=java

Gherkin Reference

Gherkin uses a set of special keywords to give structure and meaning to executable specifications. Each keyword is translated to many spoken languages; in this reference we'll use English.

Most lines in a Gherkin document start with one of the keywords.

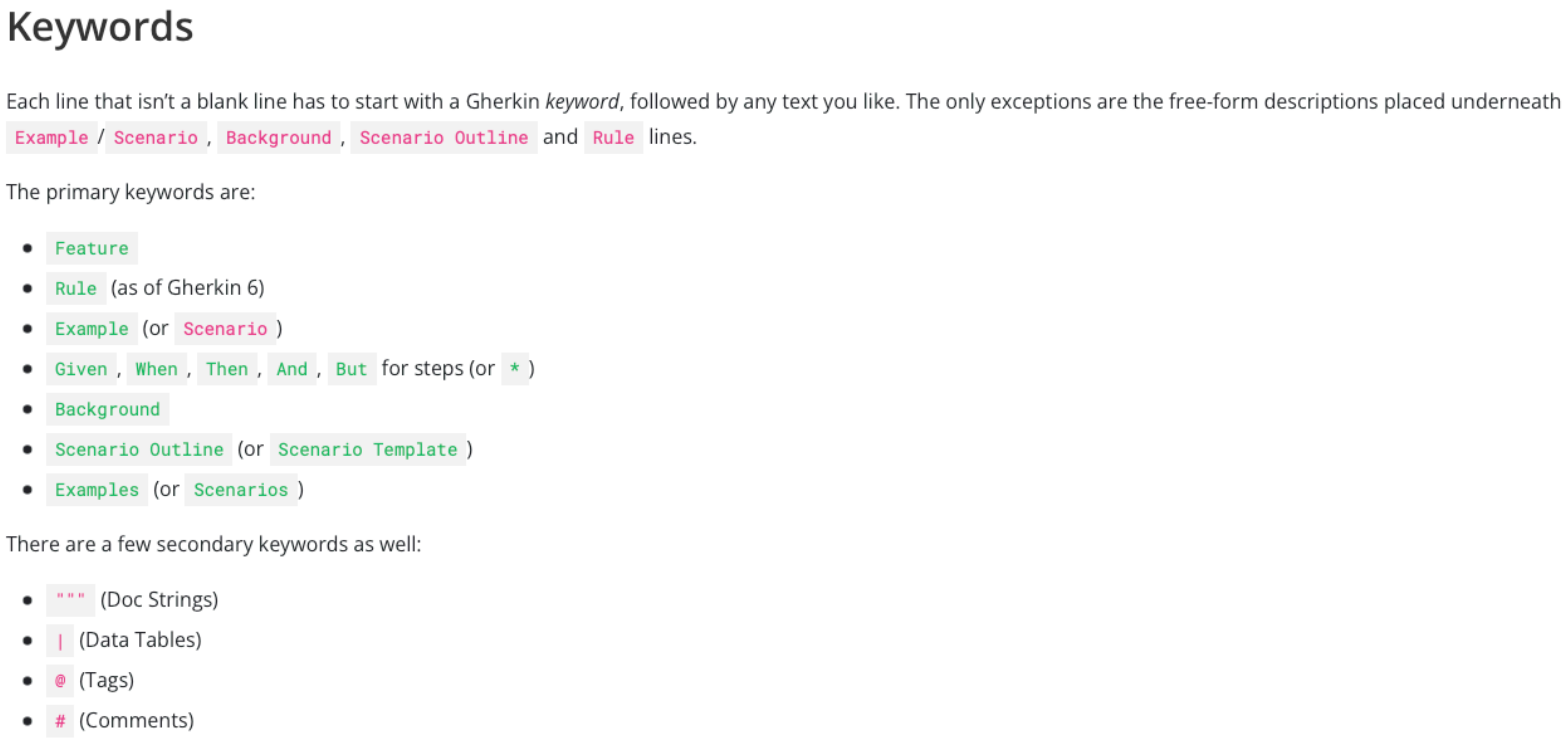
Comments are only permitted at the start of a new line, anywhere in the feature file. They begin with zero or more spaces, followed by a hash sign (`#`) and some text.

Block comments are currently not supported by Gherkin.

Either spaces or tabs may be used for indentation. The recommended indentation level is two spaces.

https://cucumber.io/docs/gherkin/reference/

What is Gherkin?



Keywords

Each line that isn't a blank line has to start with a Gherkin *keyword*, followed by any text you like. The only exceptions are the free-form descriptions placed underneath `Example` / `Scenario` , `Background` , `Scenario Outline` and `Rule` lines.

The primary keywords are:

- `Feature`
- `Rule` (as of Gherkin 6)
- `Example` (or `Scenario`)
- `Given` , `When` , `Then` , `And` , `But` for steps (or `*`)
- `Background`
- `Scenario Outline` (or `Scenario Template`)
- `Examples` (or `Scenarios`)

There are a few secondary keywords as well:

- `"""` (Doc Strings)
- `|` (Data Tables)
- `@` (Tags)
- `#` (Comments)

What is Gherkin?

Given

Given steps are used to describe the initial context of the system - the *scene* of the scenario. It is typically something that happened in the *past*.

When Cucumber executes a Given step, it will configure the system to be in a well-defined state, such as creating and configuring objects or adding data to a test database.

The purpose of Given steps is to **put the system in a known state** before the user (or external system) starts interacting with the system (in the When steps). Avoid talking about user interaction in Given's. If you were creating use cases, Given's would be your preconditions.

It's okay to have several Given steps (use And or But for number 2 and upwards to make it more readable).

Examples:

- Mickey and Minnie have started a game
- I am logged in
- Joe has a balance of £42

What is Gherkin?

When

When steps are used to describe an event, or an *action*. This can be a person interacting with the system, or it can be an event triggered by another system.

Examples:

- Guess a word
- Invite a friend
- Withdraw money

What is Gherkin?

Then

Then steps are used to describe an *expected* outcome, or result.

The step definition of a Then step should use an *assertion* to compare the *actual* outcome (what the system actually does) to the *expected* outcome (what the step says the system is supposed to do).

An outcome *should* be on an **observable** output. That is, something that comes *out* of the system (report, user interface, message), and not a behaviour deeply buried inside the system (like a record in a database).

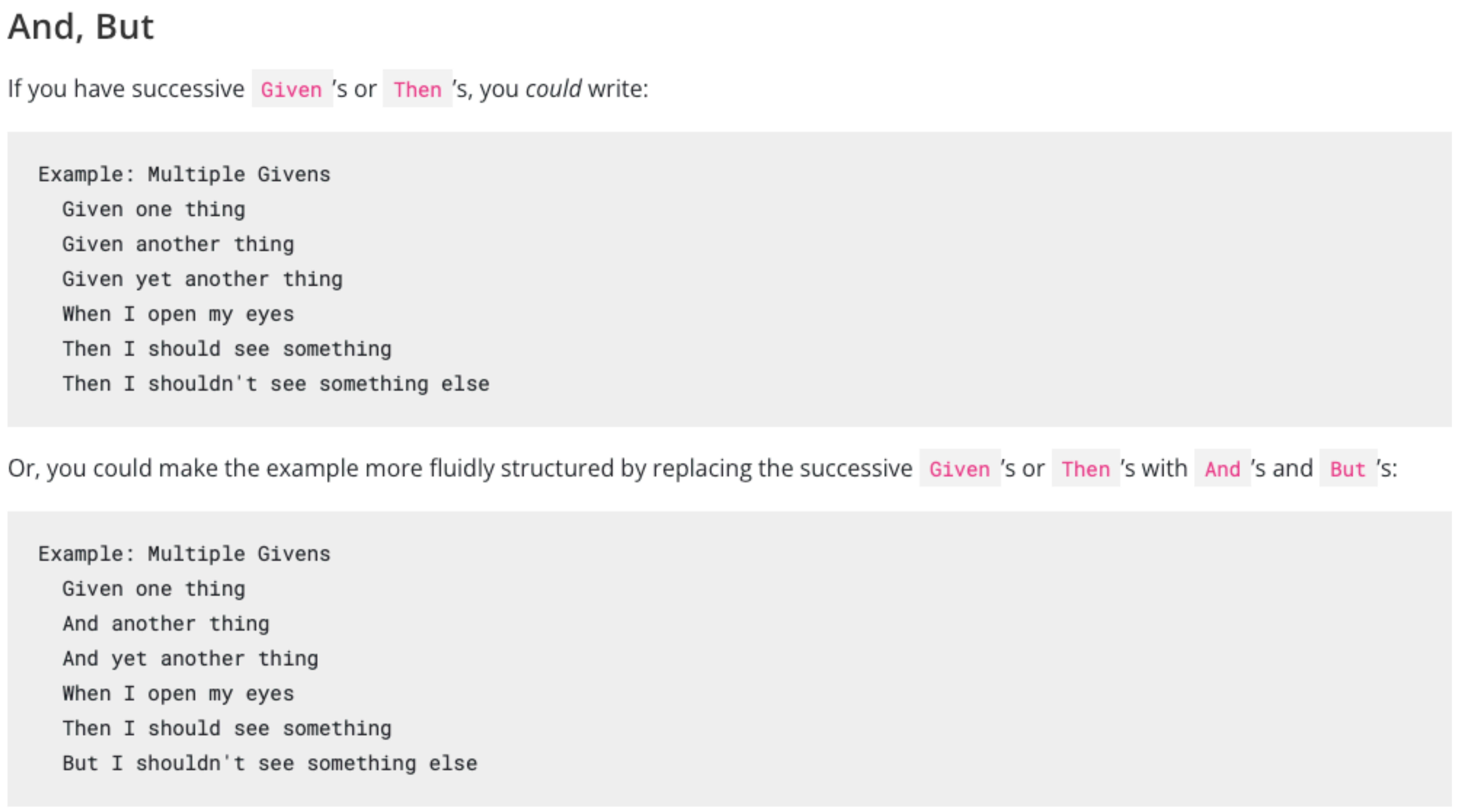
Examples:

- See that the guessed word was wrong
- Receive an invitation
- Card should be swallowed

While it might be tempting to implement Then steps to look in the database - resist that temptation!

You should only verify an outcome that is observable for the user (or external system), and changes to a database are usually not.

What is Gherkin?



And, But

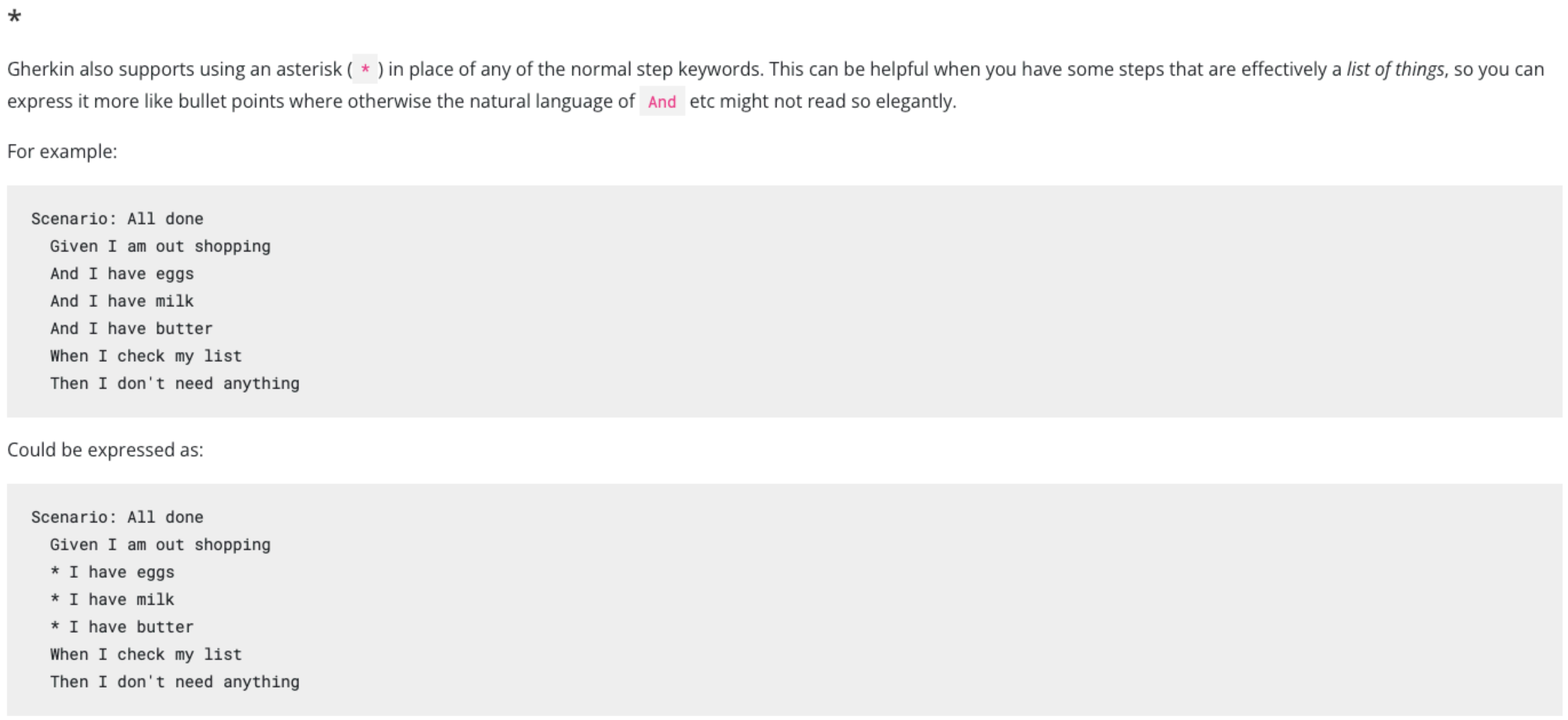
If you have successive `Given`’s or `Then`’s, you *could* write:

```
Example: Multiple Givens
  Given one thing
  Given another thing
  Given yet another thing
  When I open my eyes
  Then I should see something
  Then I shouldn't see something else
```

Or, you could make the example more fluidly structured by replacing the successive `Given`’s or `Then`’s with `And`’s and `But`’s:

```
Example: Multiple Givens
  Given one thing
  And another thing
  And yet another thing
  When I open my eyes
  Then I should see something
  But I shouldn't see something else
```

What is Gherkin?



*

Gherkin also supports using an asterisk (`*`) in place of any of the normal step keywords. This can be helpful when you have some steps that are effectively a *list of things*, so you can express it more like bullet points where otherwise the natural language of `And` etc might not read so elegantly.

For example:

```
Scenario: All done
  Given I am out shopping
  And I have eggs
  And I have milk
  And I have butter
  When I check my list
  Then I don't need anything
```

Could be expressed as:

```
Scenario: All done
  Given I am out shopping
  * I have eggs
  * I have milk
  * I have butter
  When I check my list
  Then I don't need anything
```

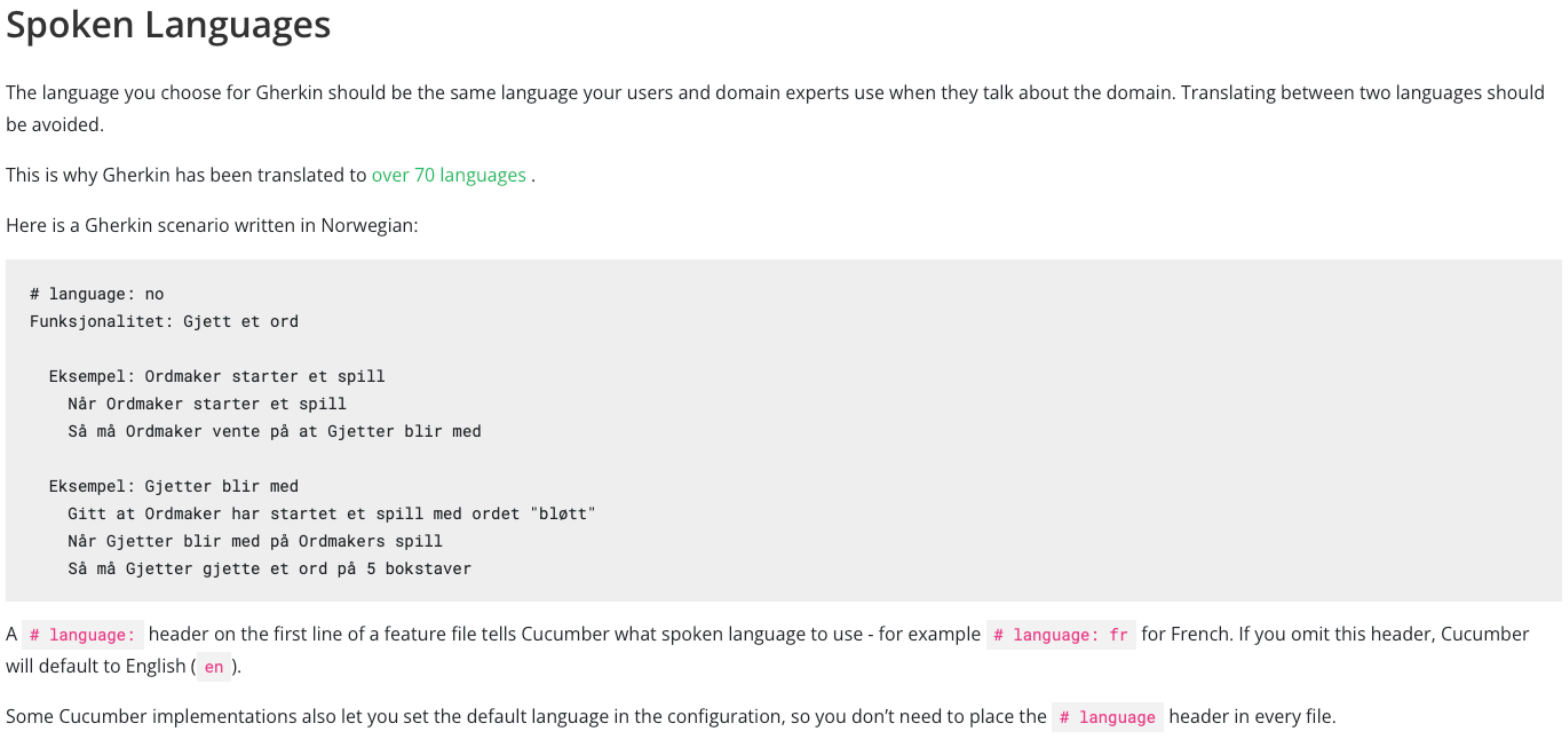
What is Gherkin?

Background

Occasionally you'll find yourself repeating the same Given steps in all of the scenarios in a Feature. These can be grouped in a background, eg:

```
Feature: Multiple site support
  Only blog owners can post to a blog, except administrators,
  who can post to all blogs.

  Background:
    Given a global administrator named "Greg"
    And a blog named "Greg's anti-tax rants"
    And a customer named "Dr. Bill"
    And a blog named "Expensive Therapy" owned by "Dr. Bill"

  Scenario: Dr. Bill posts to his own blog
    Given I am logged in as Dr. Bill
    When I try to post to "Expensive Therapy"
    Then I should see "Your article was published."

  Scenario: Dr. Bill tries to post to somebody else's blog, and fails
    Given I am logged in as Dr. Bill
    When I try to post to "Greg's anti-tax rants"
    Then I should see "Hey! That's not your blog!"
```

What is Gherkin?



Spoken Languages

The language you choose for Gherkin should be the same language your users and domain experts use when they talk about the domain. Translating between two languages should be avoided.

This is why Gherkin has been translated to over 70 languages .

Here is a Gherkin scenario written in Norwegian:

```
# language: no
Funksjonalitet: Gjett et ord

  Eksempel: Ordmaker starter et spill
    Når Ordmaker starter et spill
    Så må Ordmaker vente på at Gjetter blir med

  Eksempel: Gjetter blir med
    Gitt at Ordmaker har startet et spill med ordet "bløtt"
    Når Gjetter blir med på Ordmakers spill
    Så må Gjetter gjette et ord på 5 bokstaver
```

A `# language:` header on the first line of a feature file tells Cucumber what spoken language to use - for example `# language: fr` for French. If you omit this header, Cucumber will default to English (`en`).

Some Cucumber implementations also let you set the default language in the configuration, so you don't need to place the `# language` header in every file.

What is Gherkin?

Gherkin Reference

For more information on other parts of the language, see the **reference**:

https://cucumber.io/docs/gherkin/reference/

What is Gherkin?

For guidance on writing good gherkin models, see:

https://medium.com/@nic/writing-user-stories-with-gherkin-dda63461b1d2

https://docs.behat.org/en/v2.5/guides/1.gherkin.html

https://www.businessanalysisexperts.com/gherkin-user-stories-given-when-then-examples/

https://cucumber.io/docs/bdd/better-gherkin/

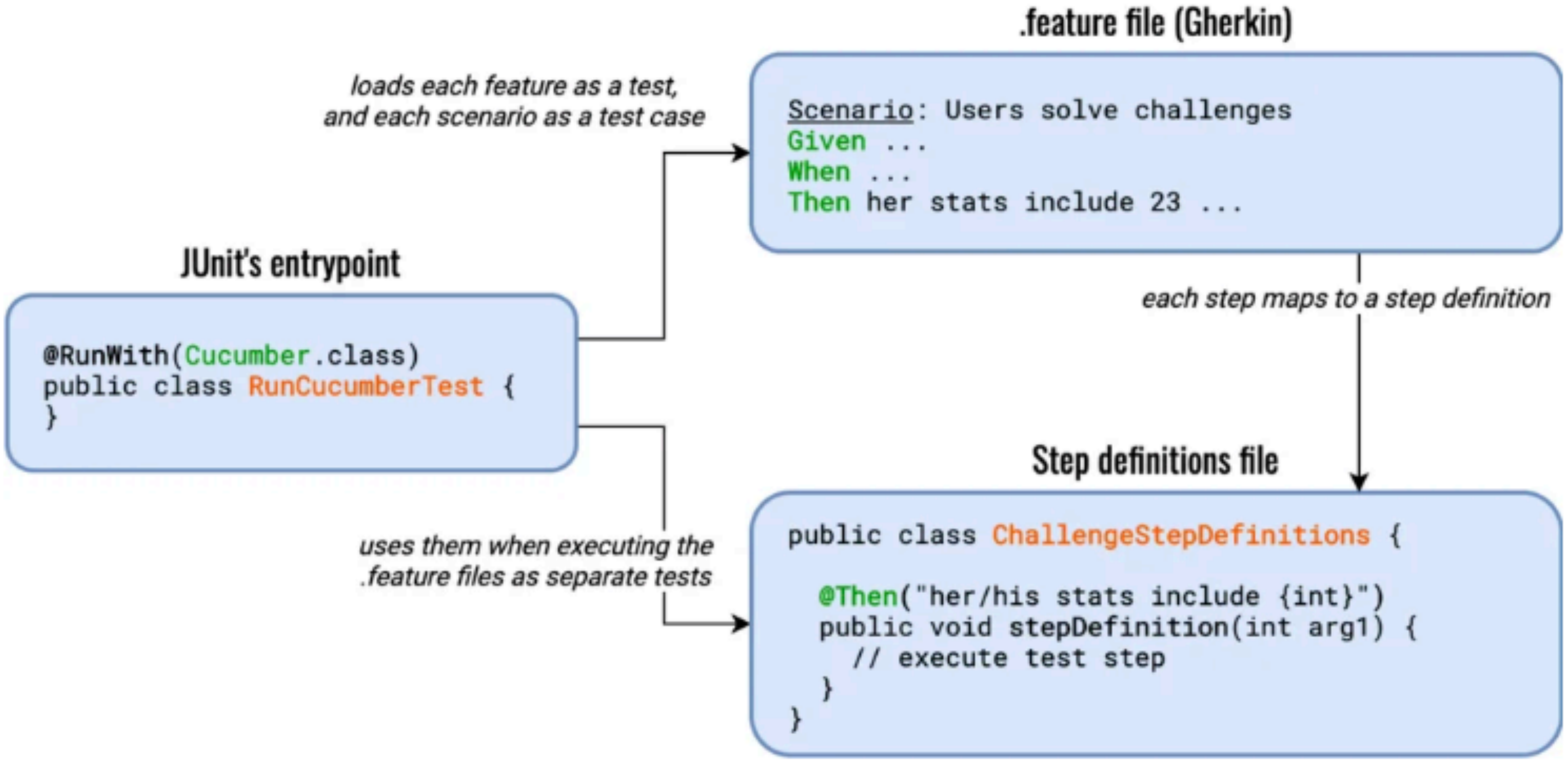
https://www.jbvigneron.fr/parlons-dev/bdd-rediger-scenarios-avec-gherkin/

https://allaboutqablog.com/ecrivez-de-beaux-cas-de-test-avec-gherkin/

What is Cucumber?

A BDD tool which automates the testing process (using Gherkin)

It uses JVM components


.feature file (Gherkin)
loads each feature as a test, and each scenario as a test case
JUnit's entrypoint
each step maps to a step definition
Step definitions file
uses them when executing the .feature files as separate tests

```
Scenario: Users solve challenges
Given ...
When ...
Then her stats include 23 ...
```

```
@RunWith(Cucumber.class)
public class RunCucumberTest {
}
```

```
public class ChallengeStepDefinitions {

  @Then("her/his stats include {int}")
  public void stepDefinition(int arg1) {
    // execute test step
  }
}
```

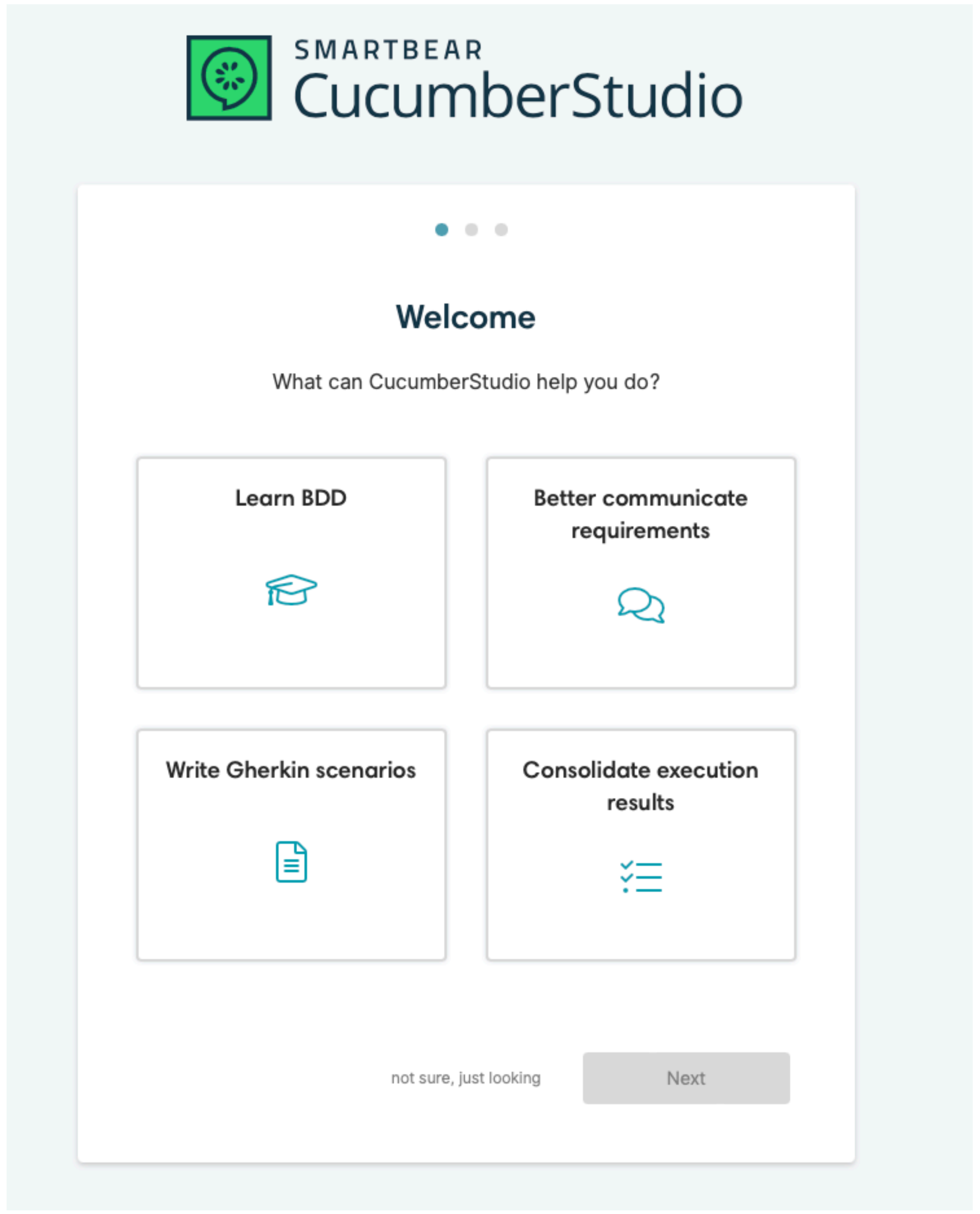
https://thepracticaldeveloper.com/cucumber-guide-1-intro-bdd-gherkin/

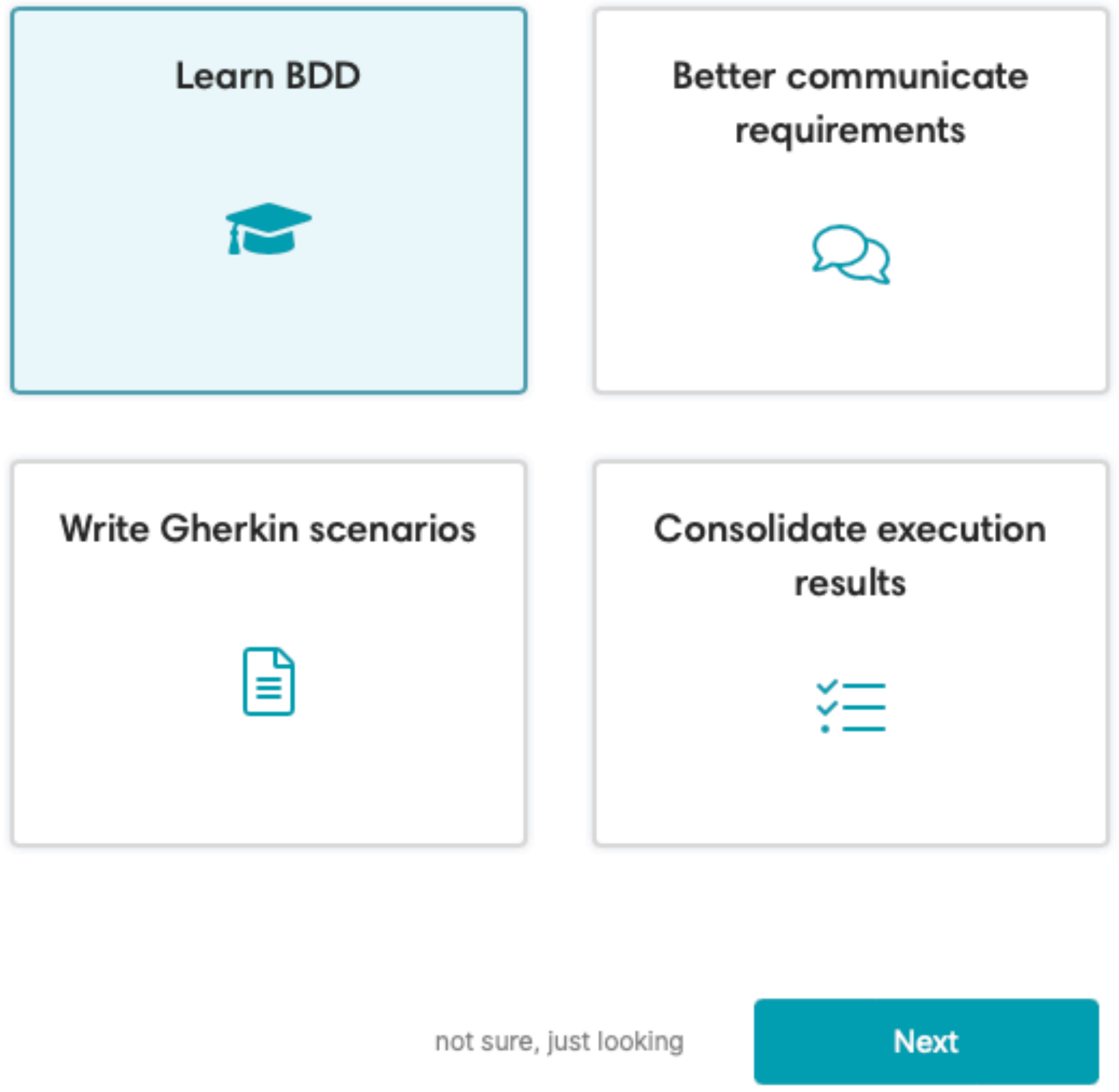
How Cucumber Works

- Cucumber processes text files that contain features looking for scenarios that can be executed against your system
- Each scenario is a list of steps that describe the pre-conditions, actions, and post-conditions of each scenario; if each step executes without error, the scenario is marked as having passed
- At the end of a run, Cucumber will report how many scenarios passed; if something fails, it provides information about what failed so the developer can make progress
- Features, scenarios, and steps are written in **Gherkin**

CucumberStudio

A good place to learn the basics - but only for a trial period

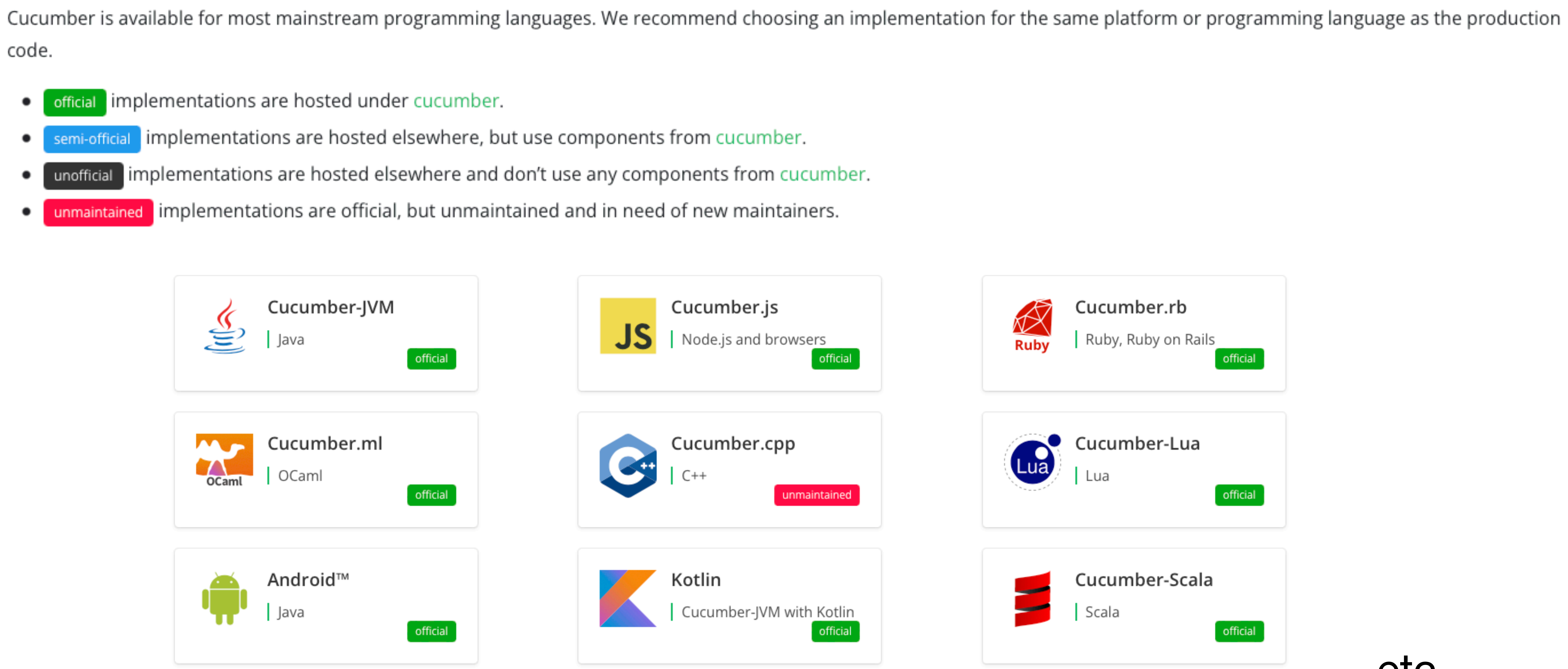

SMARTBEAR
CucumberStudio
Welcome
What can CucumberStudio help you do?
Learn BDD
Better communicate requirements
Write Gherkin scenarios
Consolidate execution results
not sure, just looking
Next


Learn BDD
Better communicate requirements
Write Gherkin scenarios
Consolidate execution results
not sure, just looking
Next

https://studio.cucumber.io/welcome

Installing and Using Cucumber in Eclipse

https://cucumber.io/docs/installation/



Cucumber is available for most mainstream programming languages. We recommend choosing an implementation for the same platform or programming language as the production code.

- official implementations are hosted under cucumber.
- semi-official implementations are hosted elsewhere, but use components from cucumber.
- unofficial implementations are hosted elsewhere and don't use any components from cucumber.
- unmaintained implementations are official, but unmaintained and in need of new maintainers.

Cucumber-JVM | Java — official

Cucumber.js | Node.js and browsers — official

Cucumber.rb | Ruby, Ruby on Rails — official

Cucumber.ml | OCaml — official

Cucumber.cpp | C++ — unmaintained

Cucumber-Lua | Lua — official

Android™ | Java — official

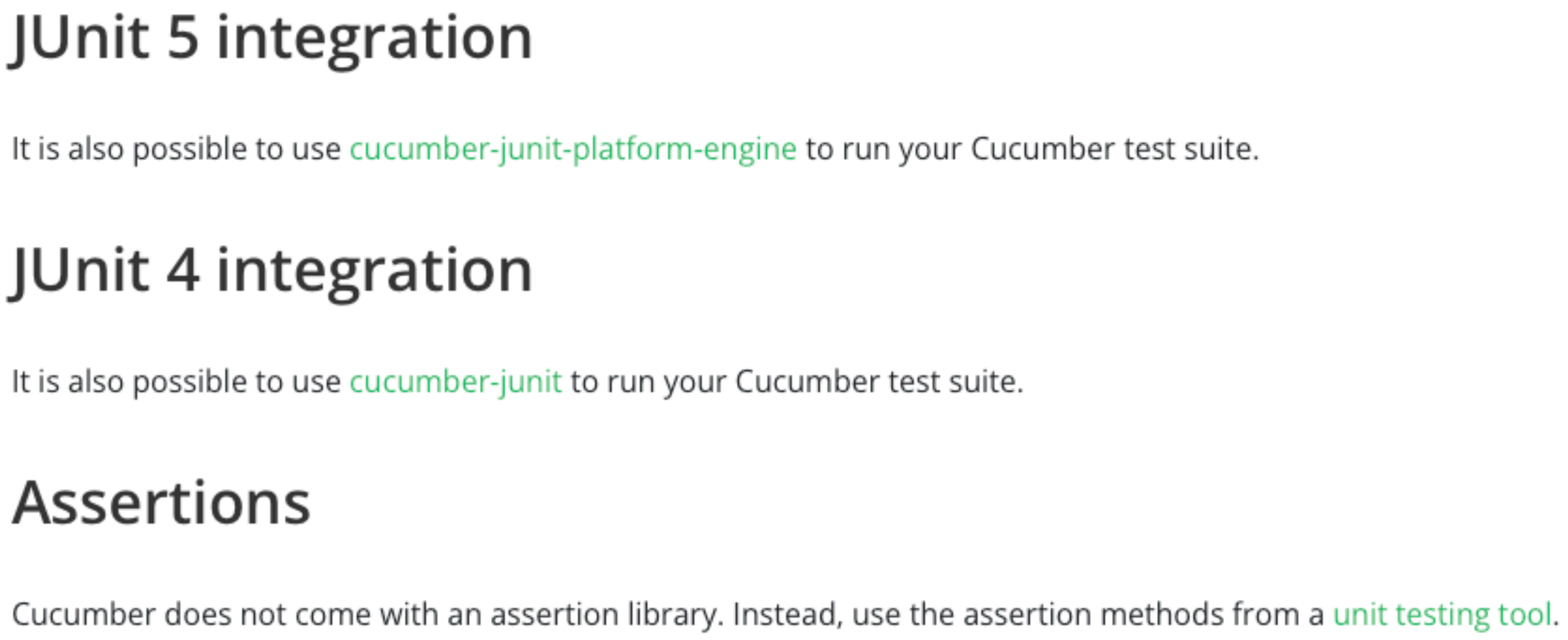
Kotlin | Cucumber-JVM with Kotlin — official

Cucumber-Scala | Scala — official

etc.

Installing and Using Cucumber in Eclipse

It is *best* combined with unit test tools:



JUnit 5 integration

It is also possible to use cucumber-junit-platform-engine to run your Cucumber test suite.

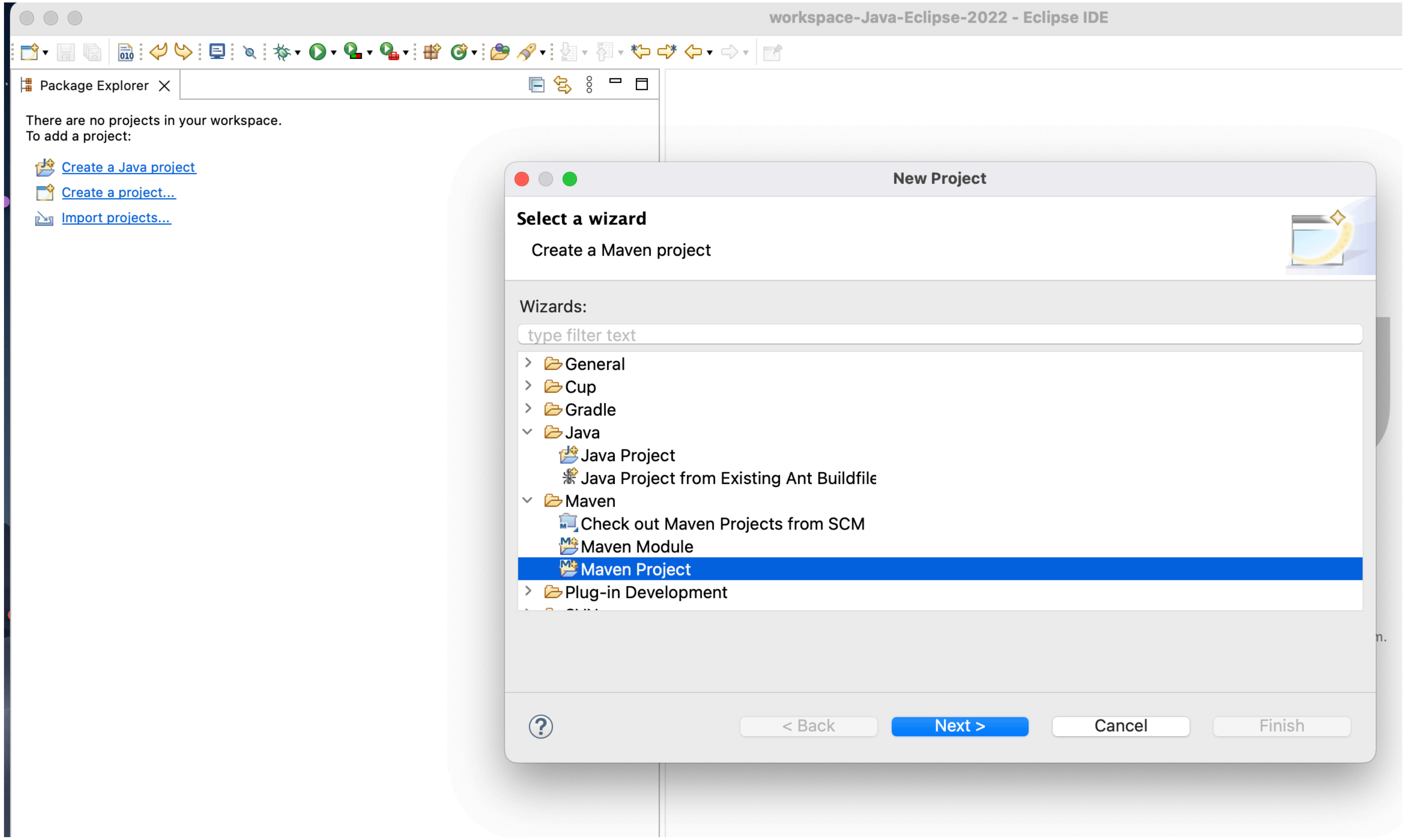
JUnit 4 integration

It is also possible to use cucumber-junit to run your Cucumber test suite.

Assertions

Cucumber does not come with an assertion library. Instead, use the assertion methods from a unit testing tool.

Installing and Using Cucumber in Eclipse



workspace-Java-Eclipse-2022 - Eclipse IDE

Package Explorer

There are no projects in your workspace.
To add a project:

Create a Java project
Create a project...
Import projects...

New Project

Select a wizard

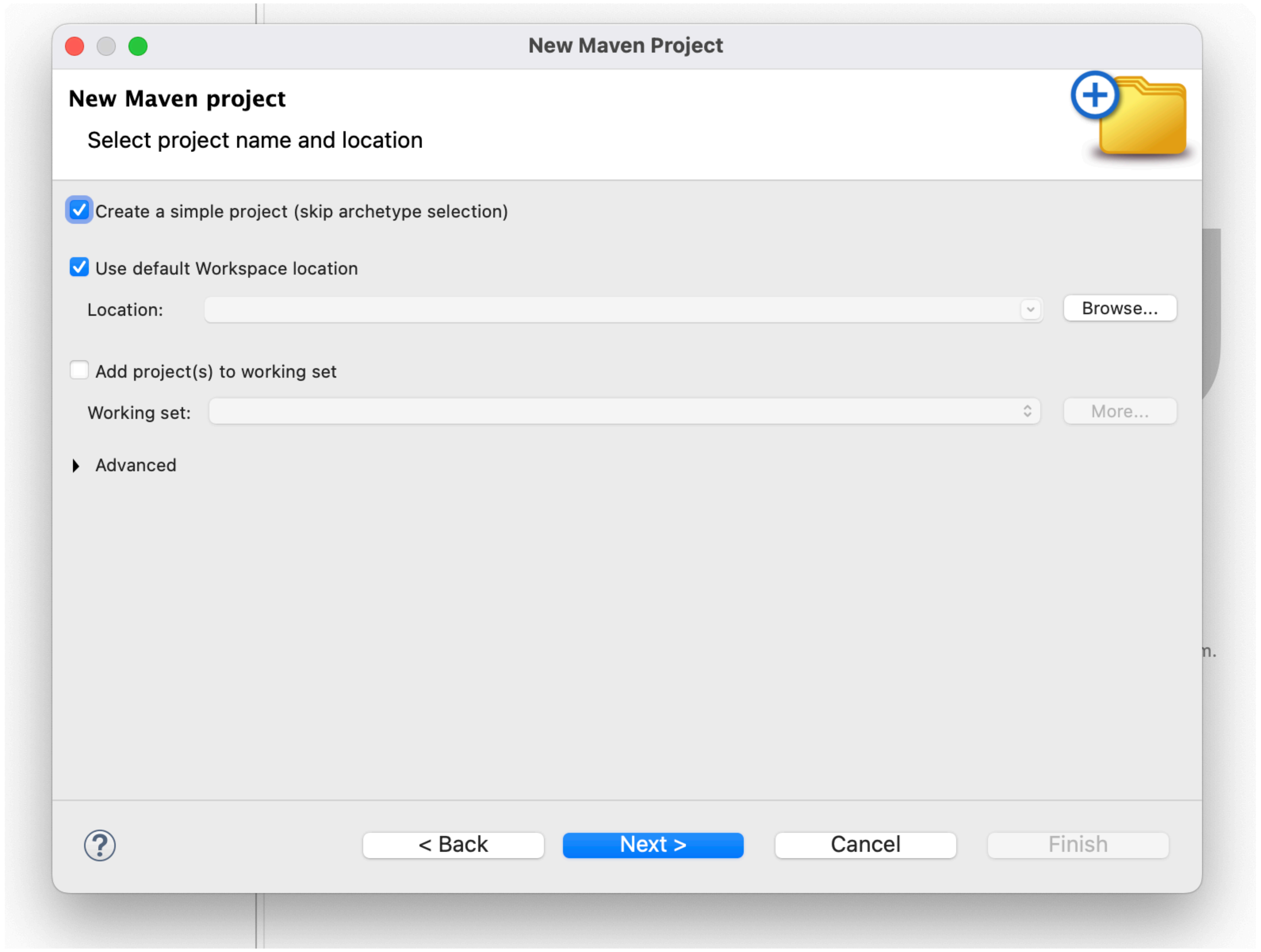
Create a Maven project

Wizards:

type filter text

- General
- Cup
- Gradle
- Java
  - Java Project
  - Java Project from Existing Ant Buildfile
- Maven
  - Check out Maven Projects from SCM
  - Maven Module
  - Maven Project
- Plug-in Development

Installing and Using Cucumber in Eclipse



New Maven Project

New Maven project

Select project name and location

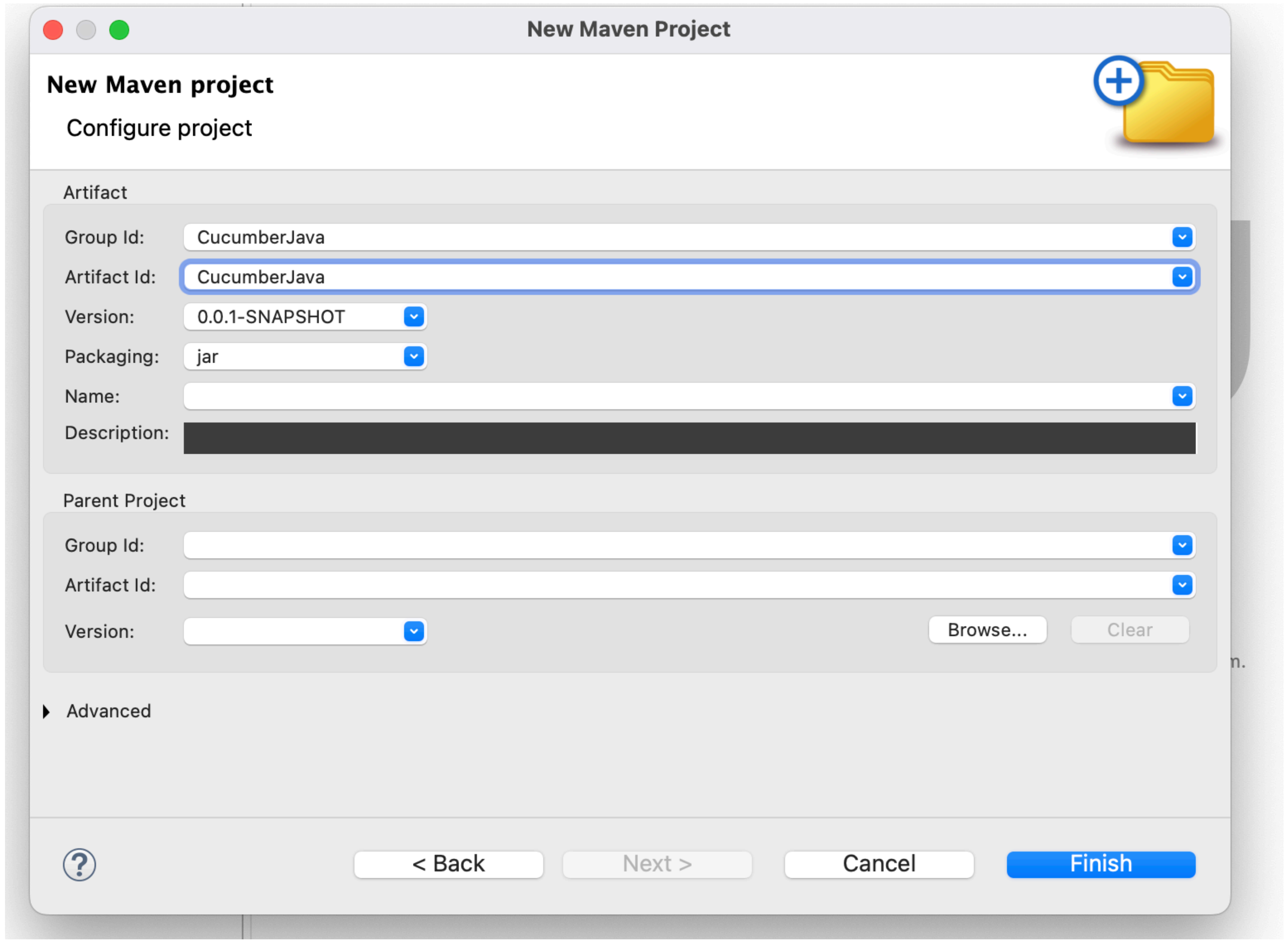
[✓] Create a simple project (skip archetype selection)

[✓] Use default Workspace location

Location:

[ ] Add project(s) to working set

Working set:



New Maven Project

New Maven project

Configure project

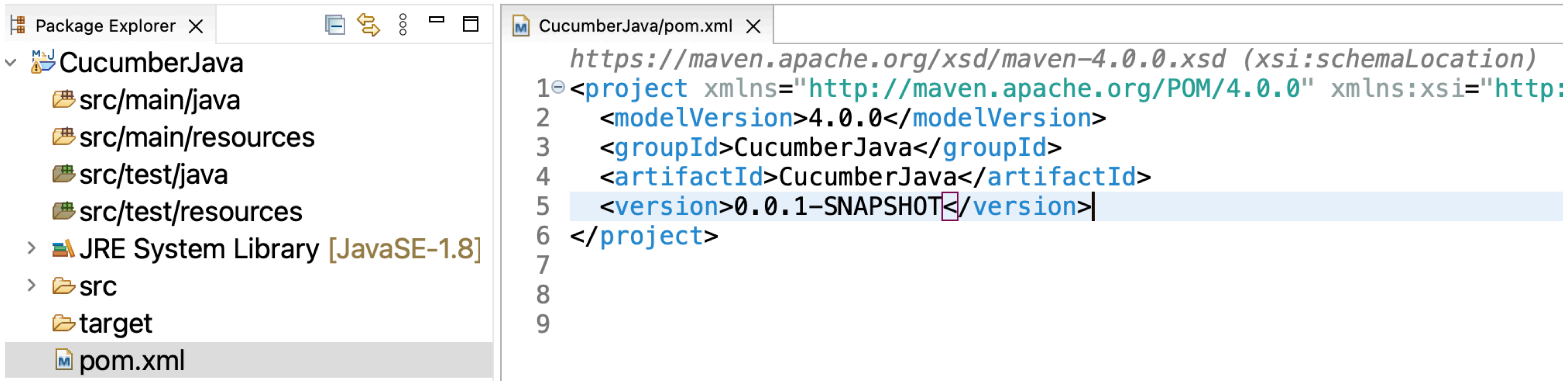
Artifact

Group Id:	CucumberJava
Artifact Id:	CucumberJava
Version:	0.0.1-SNAPSHOT
Packaging:	jar
Name:	
Description:	

Parent Project

Group Id:	
Artifact Id:	
Version:	

Installing and Using Cucumber in Eclipse



CucumberJava
- src/main/java
- src/main/resources
- src/test/java
- src/test/resources
- JRE System Library [JavaSE-1.8]
- src
- target
- pom.xml

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http:
  <modelVersion>4.0.0</modelVersion>
  <groupId>CucumberJava</groupId>
  <artifactId>CucumberJava</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

We need to add dependencies for

•cucumber-java

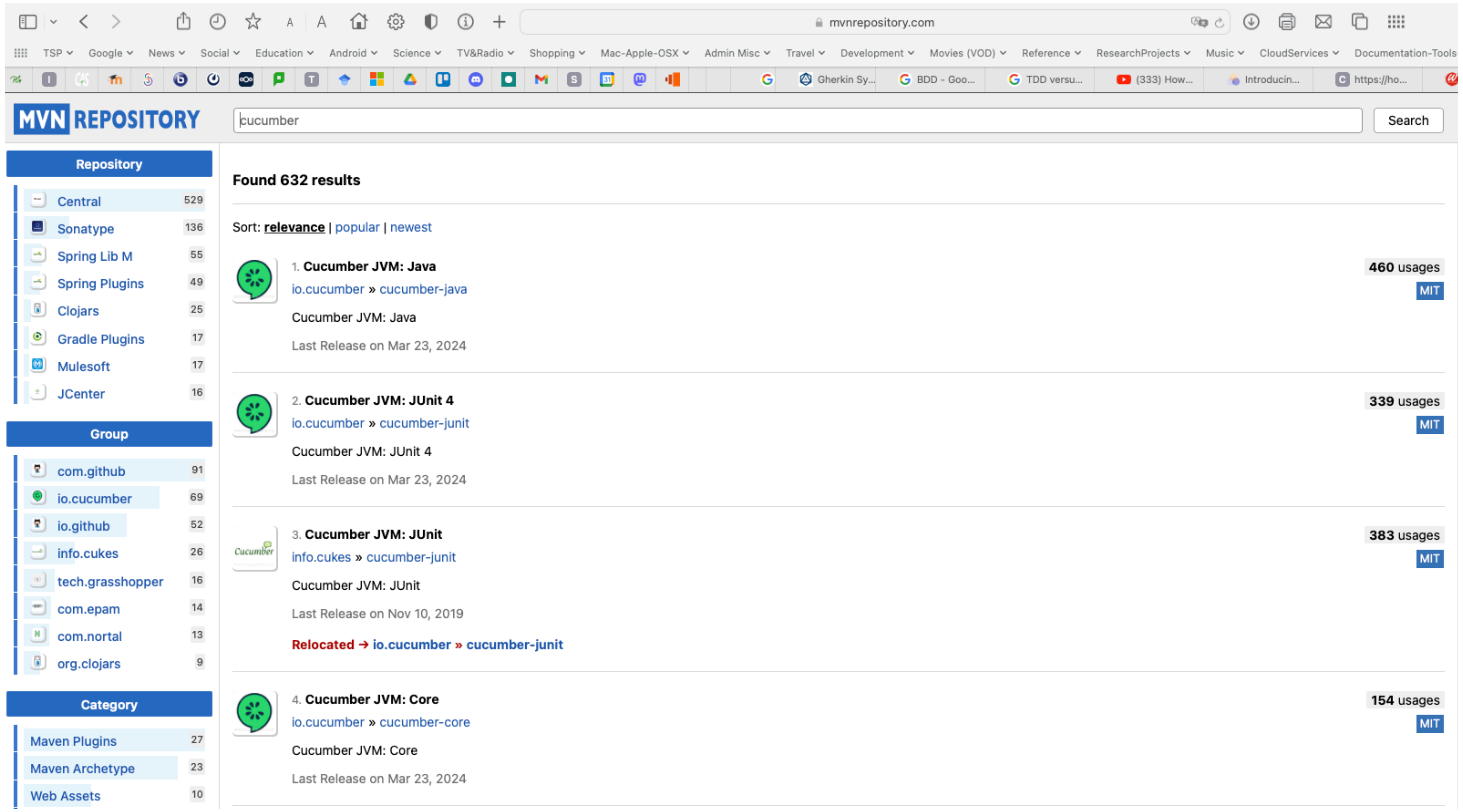
•cucumber-unit

Installing and Using Cucumber in Eclipse

CSC 4104

We need to search the man repository

- •cucumber-java
- •cucumber-unit



mvnrepository.com
TSP Google News Social Education Android Science TV&Radio Shopping Mac-Apple-OSX Admin Misc Travel Development Movies (VOD) Reference ResearchProjects Music CloudServices Documentation-Tools
Gherkin Sy... BDD - Goo... TDD versu... (333) How... Introducin... https://ho...
MVN REPOSITORY
cucumber
Search

Repository

Central	529
Sonatype	136
Spring Lib M	55
Spring Plugins	49
Clojars	25
Gradle Plugins	17
Mulesoft	17
JCenter	16

Group

com.github	91
io.cucumber	69
io.github	52
info.cukes	26
tech.grasshopper	16
com.epam	14
com.nortal	13
org.clojars	9

Category

Maven Plugins	27
Maven Archetype	23
Web Assets	10

Found 632 results

Sort: **relevance** | popular | newest

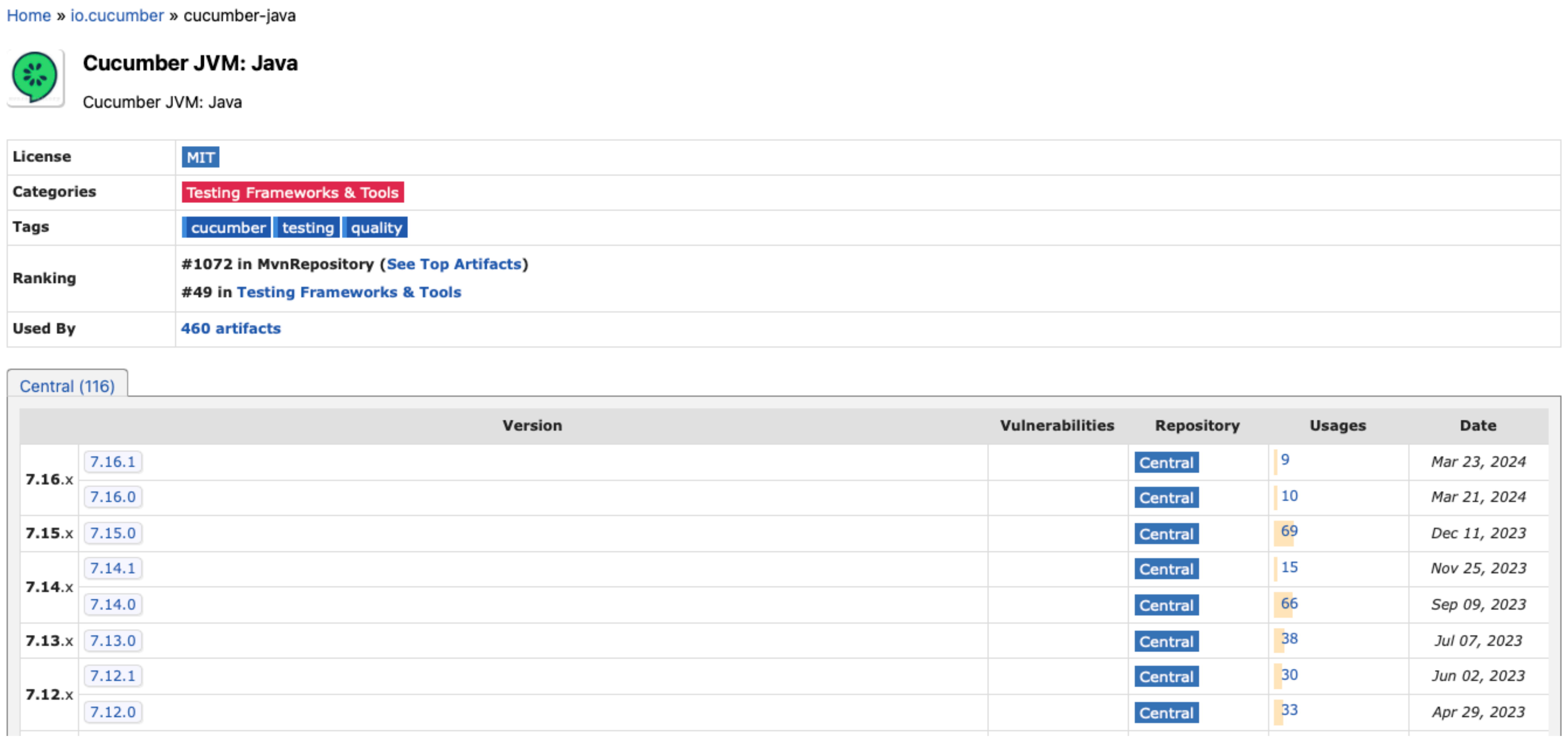
1. **Cucumber JVM: Java** — 460 usages — MIT
   io.cucumber » cucumber-java
   Cucumber JVM: Java
   Last Release on Mar 23, 2024

2. **Cucumber JVM: JUnit 4** — 339 usages — MIT
   io.cucumber » cucumber-junit
   Cucumber JVM: JUnit 4
   Last Release on Mar 23, 2024

3. **Cucumber JVM: JUnit** — 383 usages — MIT
   info.cukes » cucumber-junit
   Cucumber JVM: JUnit
   Last Release on Nov 10, 2019
   Relocated → io.cucumber » cucumber-junit

4. **Cucumber JVM: Core** — 154 usages — MIT
   io.cucumber » cucumber-core
   Cucumber JVM: Core
   Last Release on Mar 23, 2024

Installing and Using Cucumber in Eclipse



Home » io.cucumber » cucumber-java

Cucumber JVM: Java

Cucumber JVM: Java

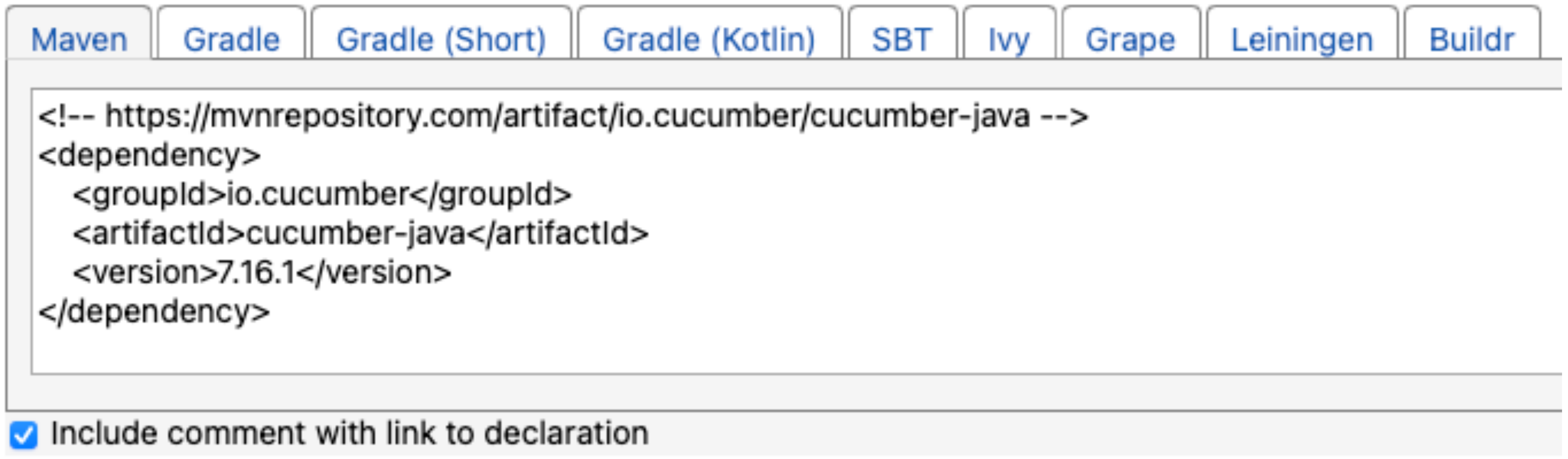
License	MIT
Categories	Testing Frameworks & Tools
Tags	cucumber testing quality
Ranking	#1072 in MvnRepository (See Top Artifacts) #49 in Testing Frameworks & Tools
Used By	460 artifacts

Central (116)

Version		Vulnerabilities	Repository	Usages	Date
7.16.x	7.16.1		Central	9	Mar 23, 2024
	7.16.0		Central	10	Mar 21, 2024
7.15.x	7.15.0		Central	69	Dec 11, 2023
7.14.x	7.14.1		Central	15	Nov 25, 2023
	7.14.0		Central	66	Sep 09, 2023
7.13.x	7.13.0		Central	38	Jul 07, 2023
7.12.x	7.12.1		Central	30	Jun 02, 2023
	7.12.0		Central	33	Apr 29, 2023

Download the latest version

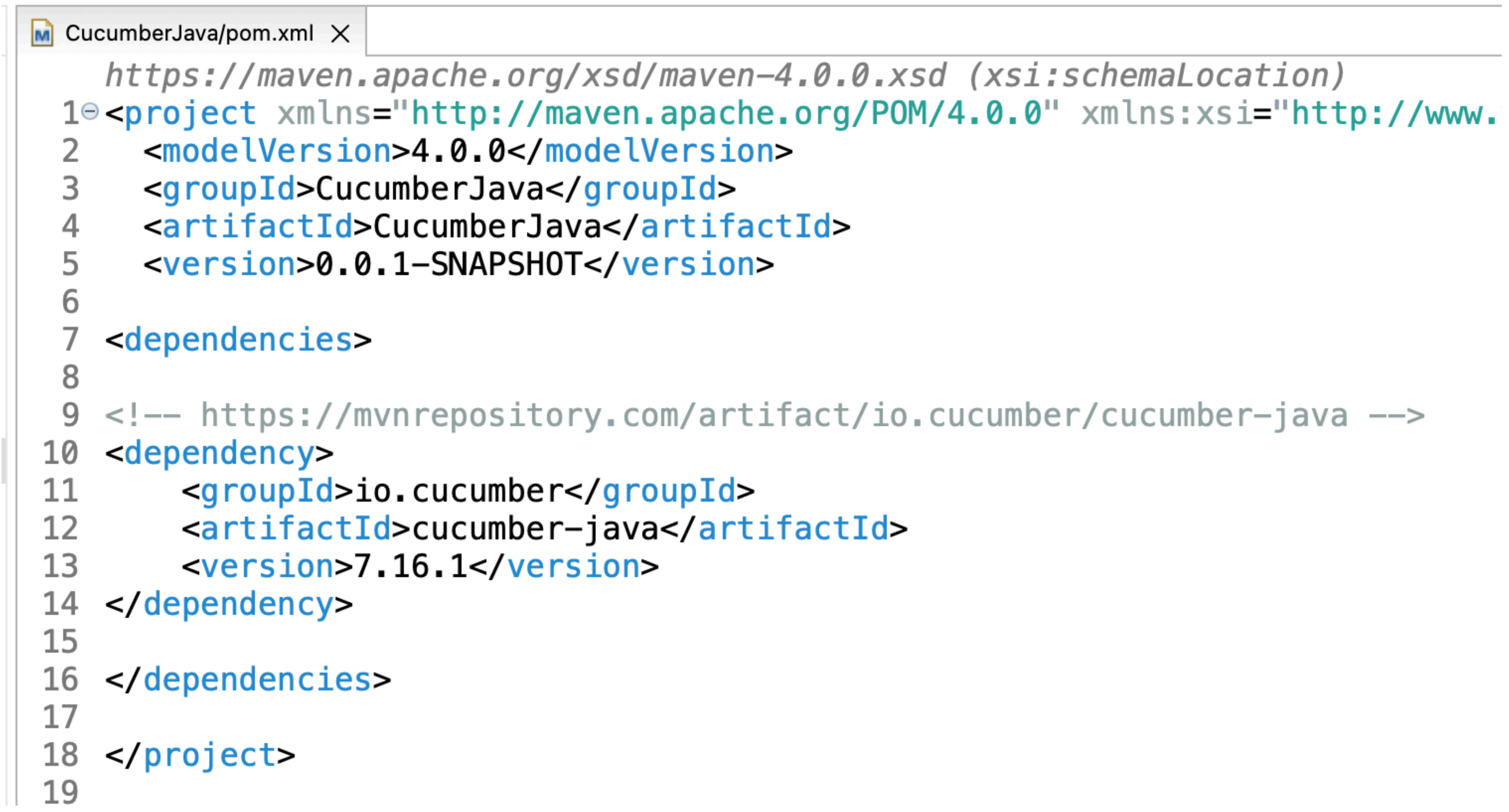
Installing and Using Cucumber in Eclipse



```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
<dependency>
   <groupId>io.cucumber</groupId>
   <artifactId>cucumber-java</artifactId>
   <version>7.16.1</version>
</dependency>
```

☑ Include comment with link to declaration

Cut-and-paste the "io.cucumber" depedency into the pom.xml



```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.
  <modelVersion>4.0.0</modelVersion>
  <groupId>CucumberJava</groupId>
  <artifactId>CucumberJava</artifactId>
  <version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.16.1</version>
</dependency>

</dependencies>

</project>

```

Installing and Using Cucumber in Eclipse

Follow the same steps for the cucumber JVM JUnit, and it it to the pom.xml

```xml
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>7.16.1</version>
    <scope>test</scope>
</dependency>
```

Note - the extra scope attribute - we may wish to change this later or remove it

What is pom.xml ?

A Project Object Model or POM is the fundamental unit of work in Maven.

It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

Dependency Scope

Dependency scope is used to limit the transitivity of a dependency and to determine when a dependency is included in a classpath.

There are 6 scopes:

- **compile**
  This is the default scope, used if none is specified. Compile dependencies are available in all classpaths of a project. Furthermore, those dependencies are propagated to dependent projects.
- **provided**
  This is much like `compile`, but indicates you expect the JDK or a container to provide the dependency at runtime. For example, when building a web application for the Java Enterprise Edition, you would set the dependency on the Servlet API and related Java EE APIs to scope `provided` because the web container provides those classes. A dependency with this scope is added to the classpath used for compilation and test, but not the runtime classpath. It is not transitive.
- **runtime**
  This scope indicates that the dependency is not required for compilation, but is for execution. Maven includes a dependency with this scope in the runtime and test classpaths, but not the compile classpath.
- **test**
  This scope indicates that the dependency is not required for normal use of the application, and is only available for the test compilation and execution phases. This scope is not transitive. Typically this scope is used for test libraries such as JUnit and Mockito. It is also used for non-test libraries such as Apache Commons IO if those libraries are used in unit tests (src/test/java) but not in the model code (src/main/java).
- **system**
  This scope is similar to `provided` except that you have to provide the JAR which contains it explicitly. The artifact is always available and is not looked up in a repository.
- **import**
  This scope is only supported on a dependency of type `pom` in the `<dependencyManagement>` section.

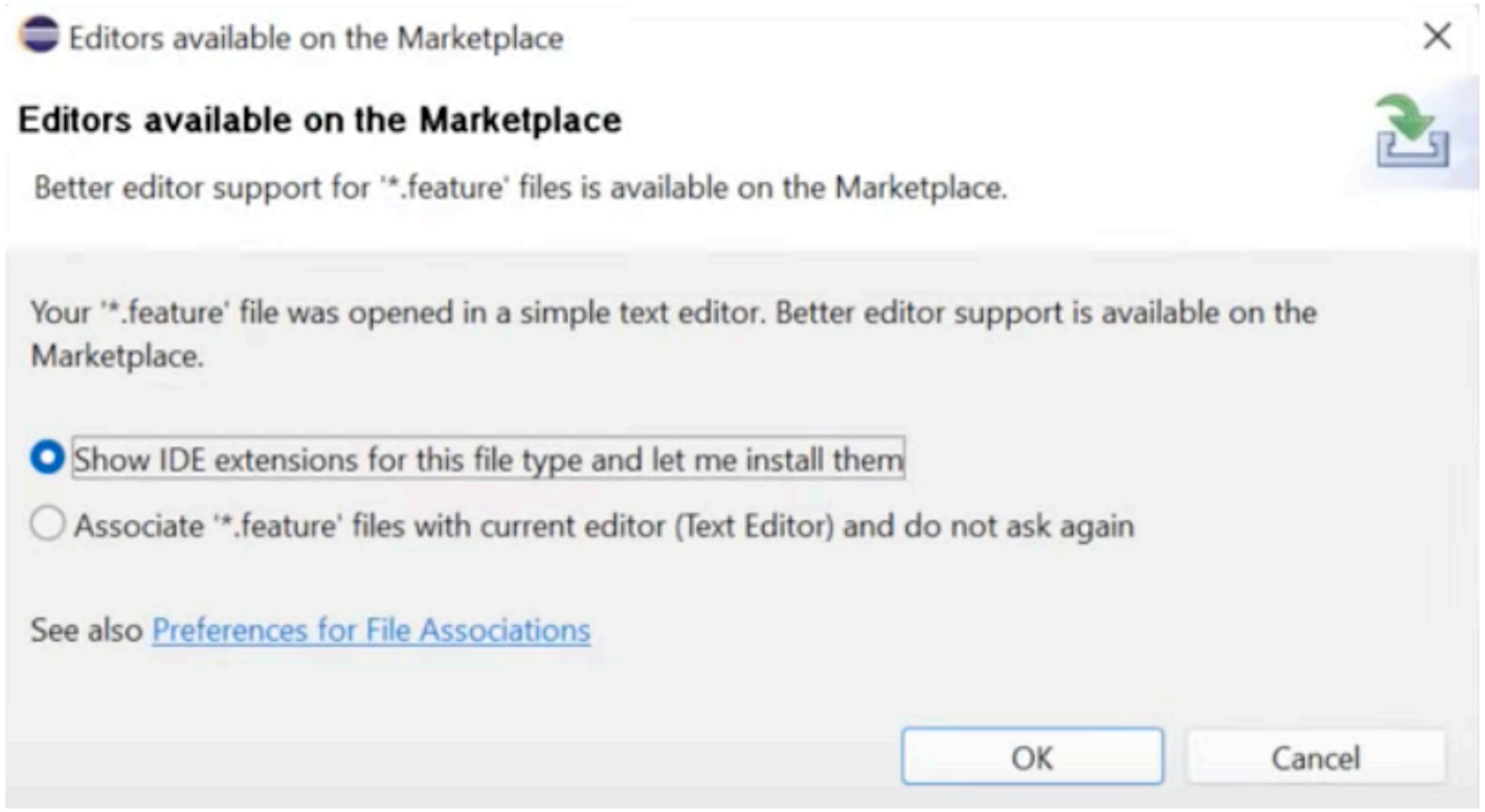
https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html

Installing and Using Cucumber in Eclipse

Create a folder for our feature/scenario files, and a test feature (for a login, eg):

login.feature

We need to associate the .feature file type with a plugin (editor)



Editors available on the Marketplace

Editors available on the Marketplace

Better editor support for '*.feature' files is available on the Marketplace.

Your '*.feature' file was opened in a simple text editor. Better editor support is available on the Marketplace.

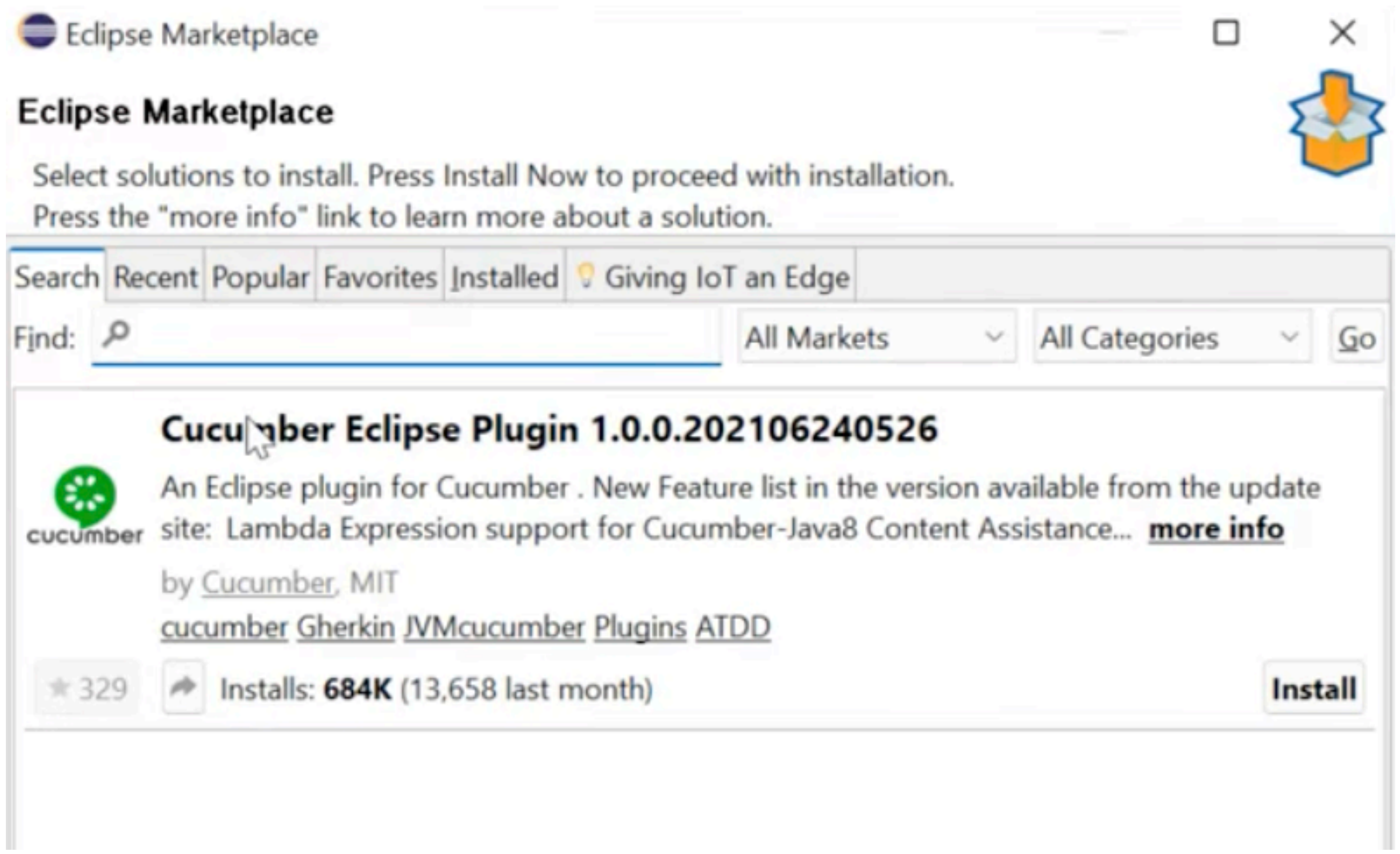
[✓ Show IDE extensions for this file type and let me install them]

Associate '*.feature' files with current editor (Text Editor) and do not ask again

See also Preferences for File Associations

Installing and Using Cucumber in Eclipse

If you have not already installed the appropriate (Cucumber plugin) it will take you to the Eclipse Marketplace


Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

Cucumber Eclipse Plugin 1.0.0.202106240526

An Eclipse plugin for Cucumber . New Feature list in the version available from the update site: Lambda Expression support for Cucumber-Java8 Content Assistance... **more info**

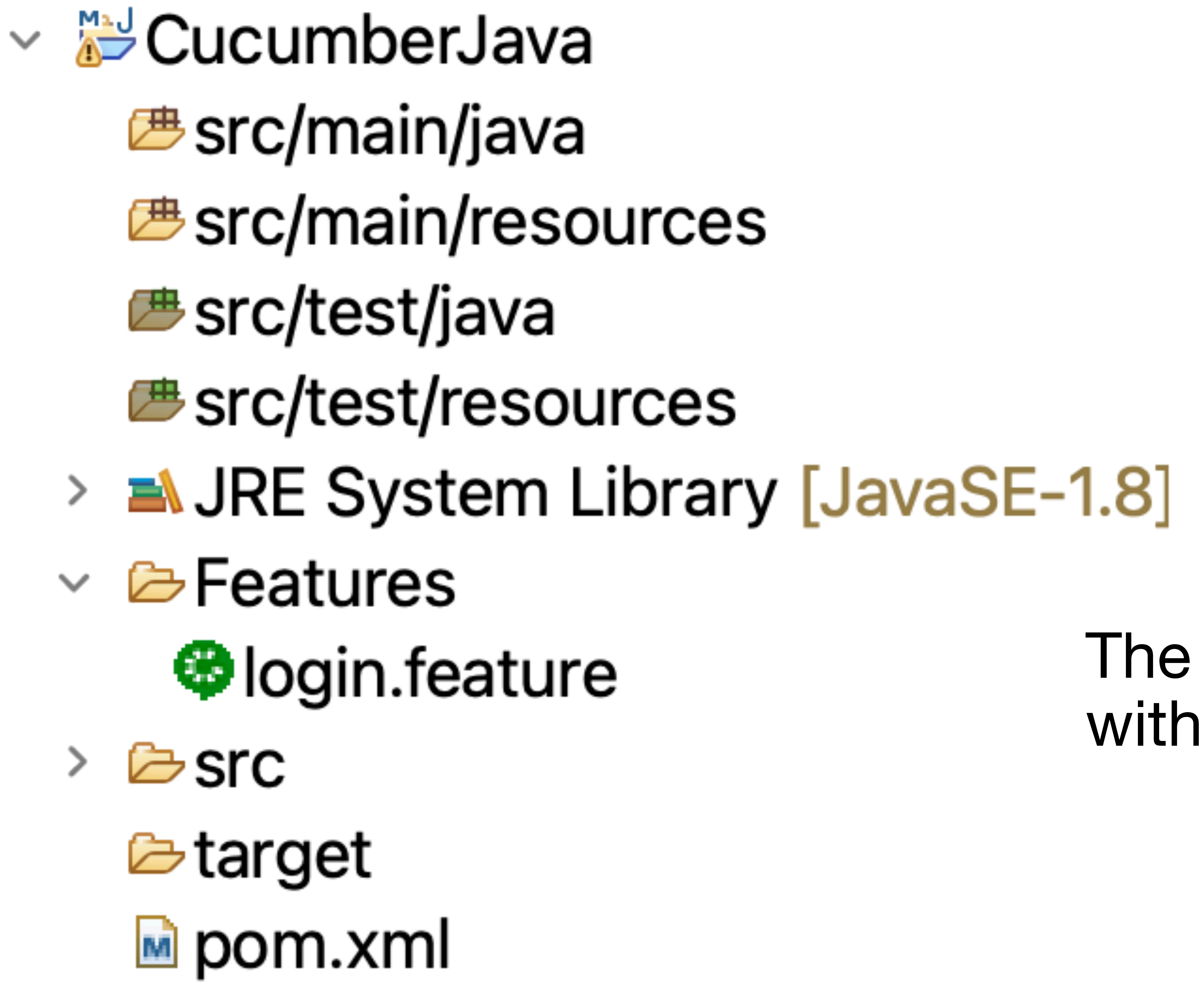
by Cucumber, MIT

cucumber Gherkin JVMcucumber Plugins ATDD

329 Installs: **684K** (13,658 last month) **Install**

So install it, and restart

Installing and Using Cucumber in Eclipse



- CucumberJava
  - src/main/java
  - src/main/resources
  - src/test/java
  - src/test/resources
  - JRE System Library [JavaSE-1.8]
  - Features
    - login.feature
  - src
  - target
  - pom.xml

The Cucumber icon should be associated with the “login” feature file

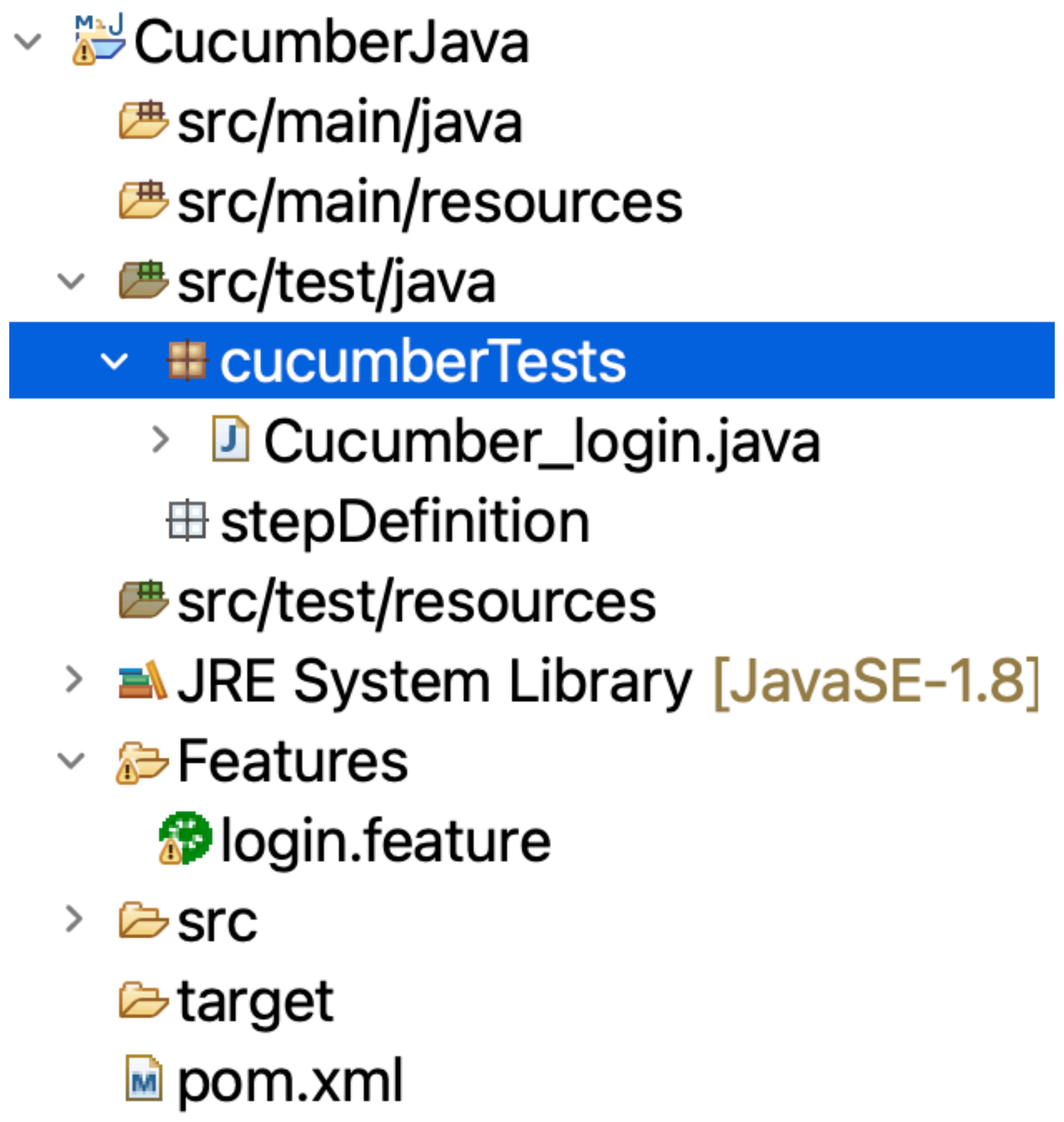
Installing and Using Cucumber in Eclipse

Now we will write a feature model/specification:

```
Feature: Login
  How to login to the system

  Scenario: Login Successful
    Given user navigates to the website javatpoint.com
    And there user logs in through Login Window by using Username as "USER" and Password as "PASSWORD"
    Then login must be successful.
```

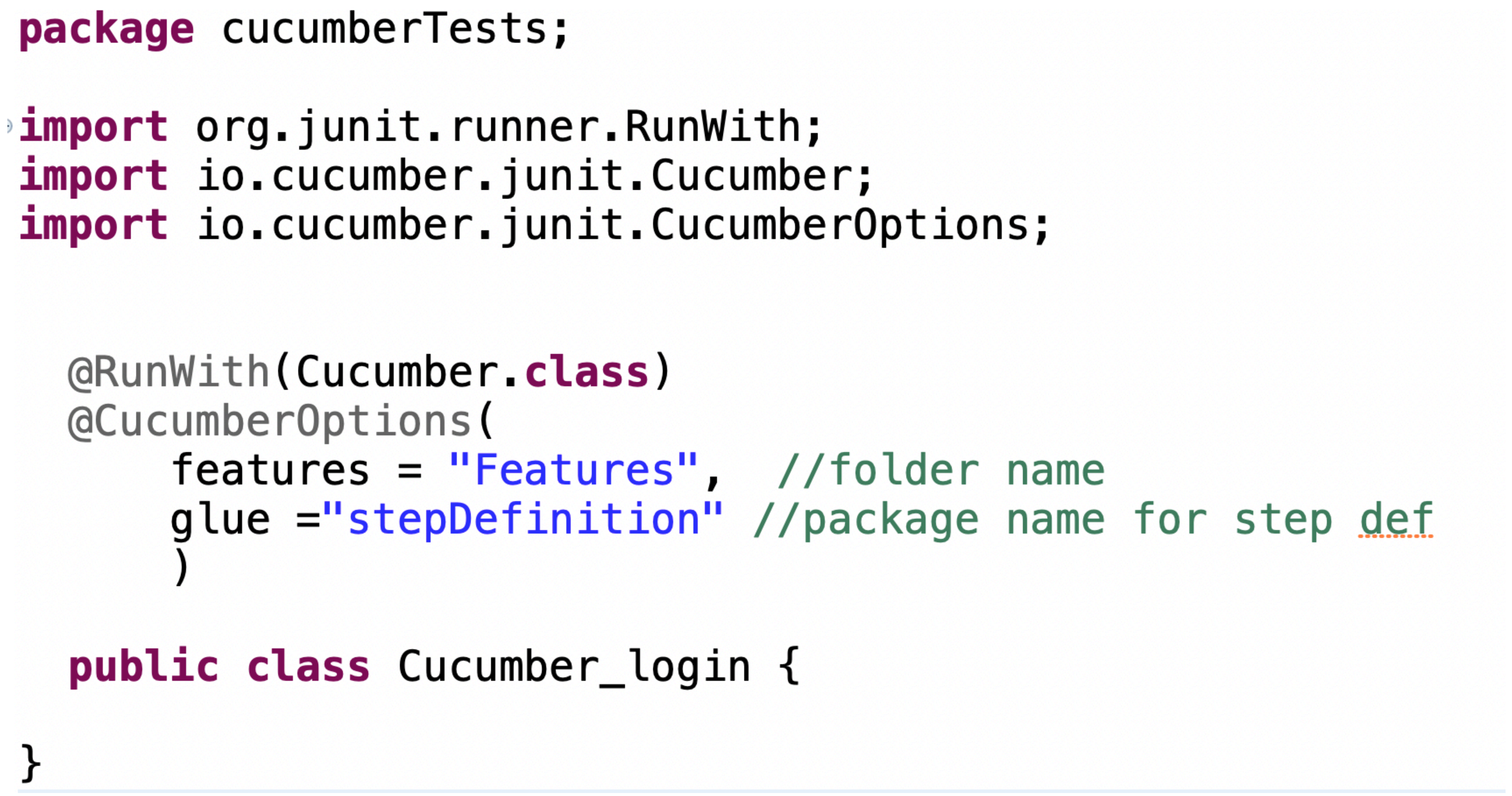
https://www.javatpoint.com/feature-file-in-cucumber-testing

Installing and Using Cucumber in Eclipse



Now add a new Java test file “Cucumber_login.java” for the login feature

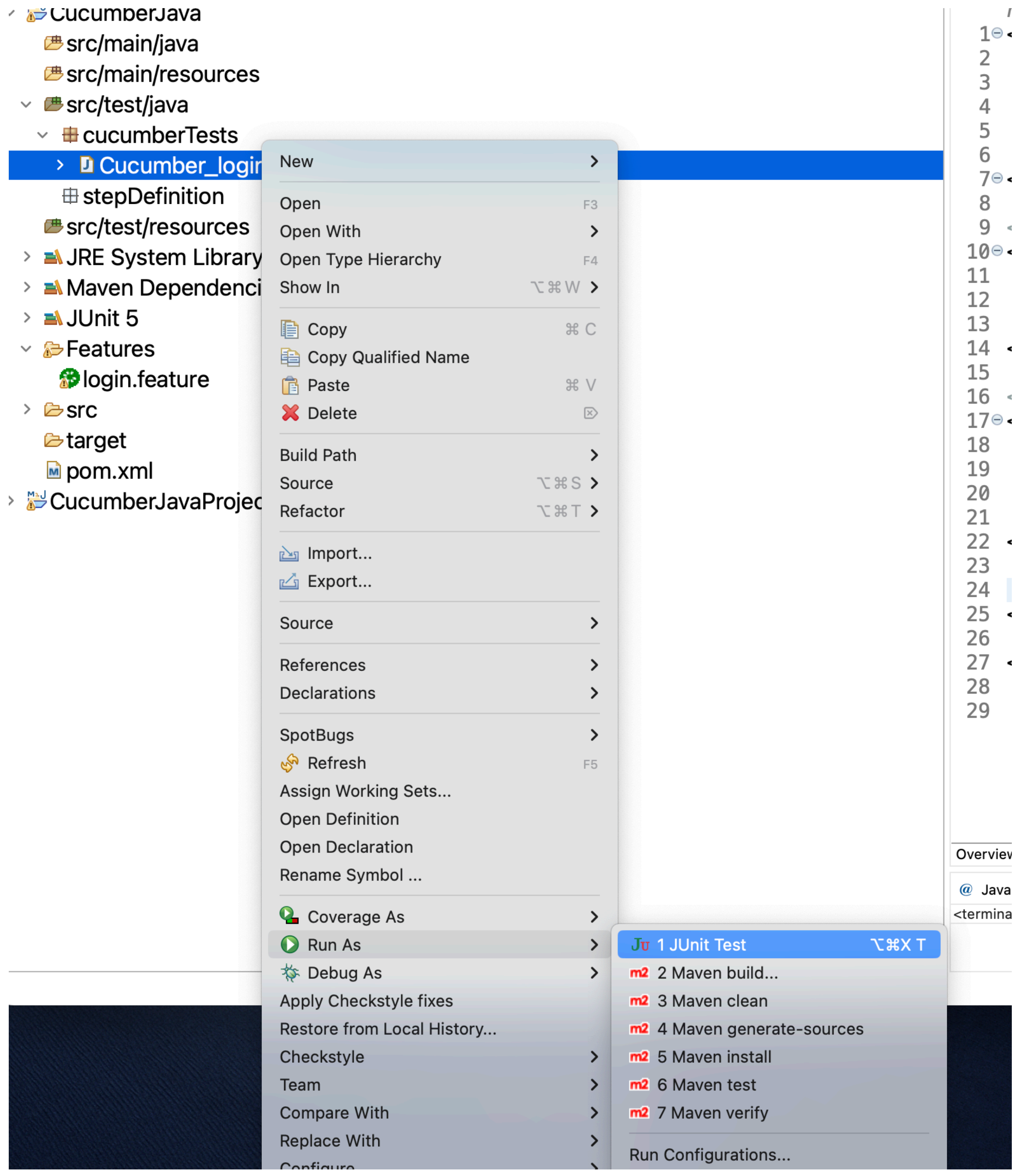
Installing and Using Cucumber in Eclipse



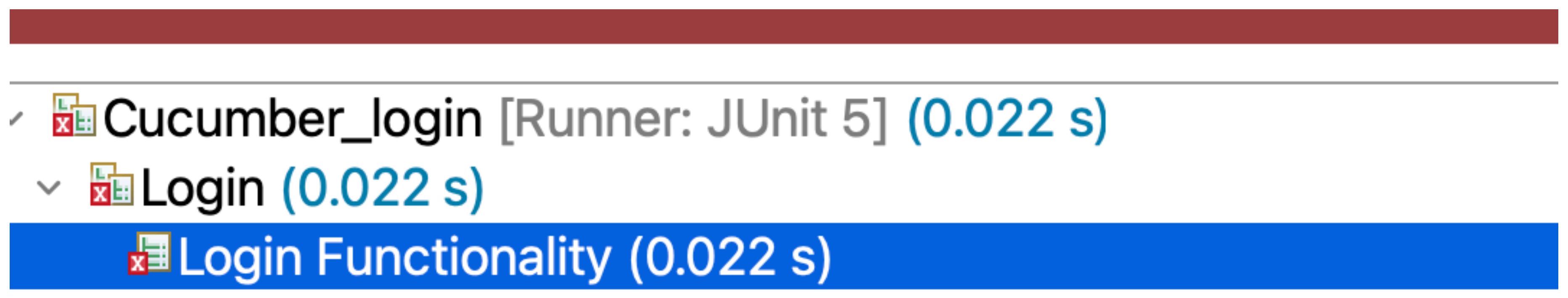
```java
package cucumberTests;

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


  @RunWith(Cucumber.class)
  @CucumberOptions(
      features = "Features",  //folder name
      glue ="stepDefinition" //package name for step def
      )

  public class Cucumber_login {

}
```

Installing and Using Cucumber in Eclipse

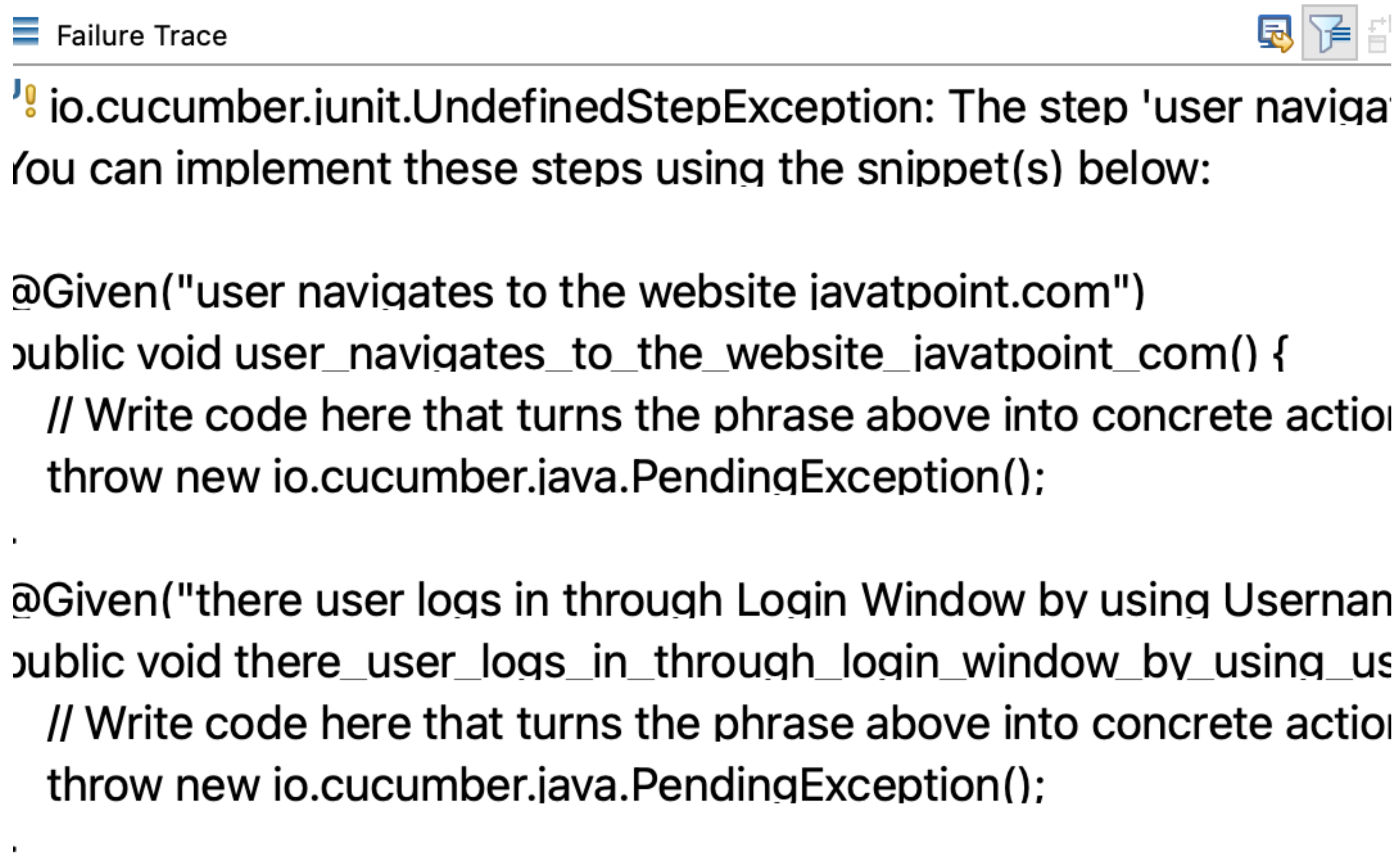
Now run as JUnit test



Installing and Using Cucumber in Eclipse


Cucumber_login [Runner: JUnit 5] (0.022 s)
Login (0.022 s)
Login Functionality (0.022 s)

Its not surprising that it doesn't work


Failure Trace

```
io.cucumber.junit.UndefinedStepException: The step 'user naviga
You can implement these steps using the snippet(s) below:

@Given("user navigates to the website javatpoint.com")
public void user_navigates_to_the_website_javatpoint_com() {
   // Write code here that turns the phrase above into concrete actio
   throw new io.cucumber.java.PendingException();
}

@Given("there user logs in through Login Window by using Usernan
public void there_user_logs_in_through_login_window_by_using_us
   // Write code here that turns the phrase above into concrete actio
   throw new io.cucumber.java.PendingException();
}
```

Installing and Using Cucumber in Eclipse

CSC 4104

The tool gives us a guide as to how to link the test code to the feature model

```
io.cucumber.junit.UndefinedStepException: The step 'user navigates to the website javatpoint.com' and 2
other step(s) are undefined.
You can implement these steps using the snippet(s) below:

@Given("user navigates to the website javatpoint.com")
public void user_navigates_to_the_website_javatpoint_com() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
@Given("there user logs in through Login Window by using Username as {string} and Password as {string}")
public void there_user_logs_in_through_login_window_by_using_username_as_and_password_as(String string,
String string2) {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
@Then("login must be successful.")
public void login_must_be_successful() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
```

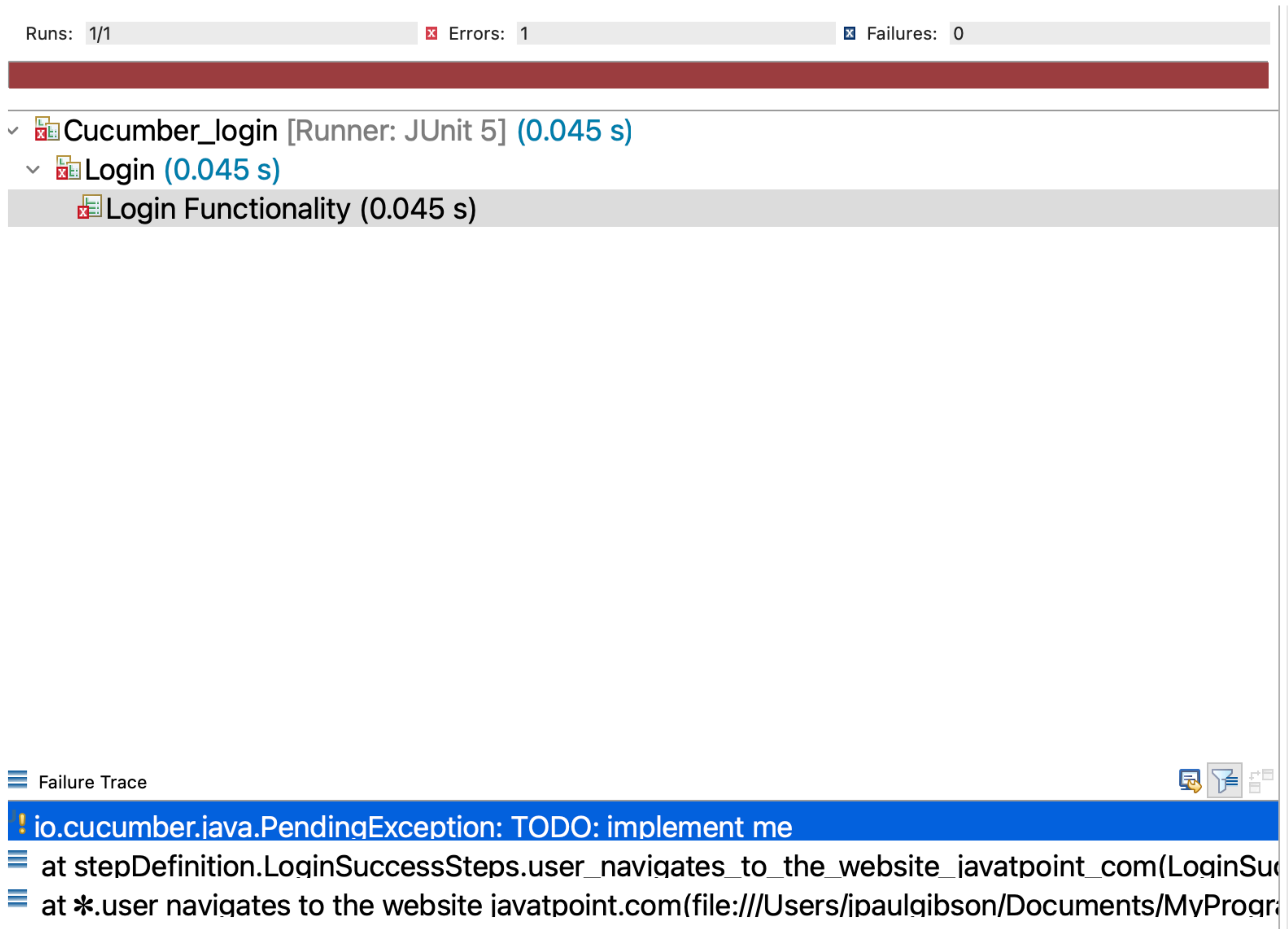
So we need to copy-paste these into a stepDefintion

Installing and Using Cucumber in Eclipse

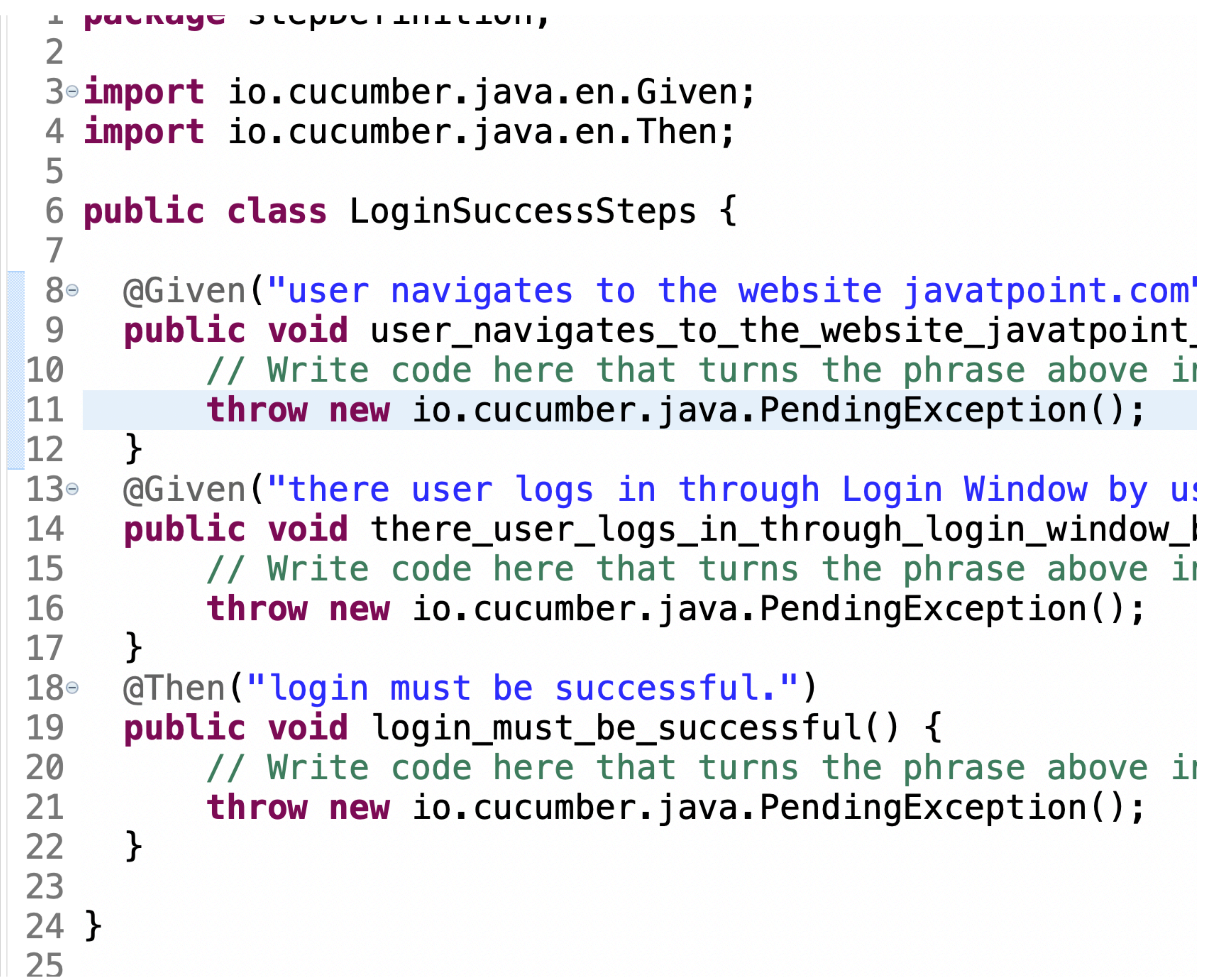
```
package stepDefinition;

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

public class LoginSuccessSteps {

  @Given("user navigates to the website javatpoint.com")
  public void user_navigates_to_the_website_javatpoint_com() {
      // Write code here that turns the phrase above into concrete actions
      throw new io.cucumber.java.PendingException();
  }
  @Given("there user logs in through Login Window by using Username as {string} and Password as {string}")
  public void there_user_logs_in_through_login_window_by_using_username_as_and_password_as(String string, String string2) {
      // Write code here that turns the phrase above into concrete actions
      throw new io.cucumber.java.PendingException();
  }
  @Then("login must be successful.")
  public void login_must_be_successful() {
      // Write code here that turns the phrase above into concrete actions
      throw new io.cucumber.java.PendingException();
  }

}
```

Installing and Using Cucumber in Eclipse

When we run the unit test code we still get an exception (but we expect that as we haven't written the appropriate implementation code):



```
Runs: 1/1        Errors: 1        Failures: 0

Cucumber_login [Runner: JUnit 5] (0.045 s)
  Login (0.045 s)
    Login Functionality (0.045 s)

Failure Trace
! io.cucumber.java.PendingException: TODO: implement me
at stepDefinition.LoginSuccessSteps.user_navigates_to_the_website_javatpoint_com(LoginSu
at ✱.user navigates to the website javatpoint.com(file:///Users/jpaulgibson/Documents/MyProgr
```



```java
package stepDefinition;

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

public class LoginSuccessSteps {

  @Given("user navigates to the website javatpoint.com'
  public void user_navigates_to_the_website_javatpoint_
      // Write code here that turns the phrase above i
      throw new io.cucumber.java.PendingException();
  }
  @Given("there user logs in through Login Window by u
  public void there_user_logs_in_through_login_window_
      // Write code here that turns the phrase above i
      throw new io.cucumber.java.PendingException();
  }
  @Then("login must be successful.")
  public void login_must_be_successful() {
      // Write code here that turns the phrase above i
      throw new io.cucumber.java.PendingException();
  }

}
```

FIRST STEPS

- Model the use cases using Gherkin syntax (in English or French)
- Install Cucumber in Eclipse
- Add Gherkin features to Eclipse project
- Write/Create test code *fragments* that are linked to the use case scenarios
- *Do not* try to write the test code implementations!