FROM TDD TO BDD AND BACK

HANDS ON LAB

GETTING STARTED GUIDE CUCUMBER

Mieke Kemme and Elke Steegmans
Nov 4, 2015

TABLE OF CONTENTS

INTRODUCTION

CUCUMBER

Cucumber lets software development teams describe how software should behave in plain text. The text is written in a business-readable domain-specific language and serves as documentation, automated tests and development-aid - all rolled into one format.

Official Website: https://cucumber.io/

THIS LAB

This lab should give you a practical introduction to writing executable specifications with *Cucumber*.

You receive:

- a small demo-application written in Java
- a few user stories
- scenarios for one of the stories

The goal of this lab is to write a few specifications for this application and make them executable, using the tool *Cucumber.*

This lab is organized as follows:

1. *installation* of the demo-application and the software needed to write the tests,
2. follow the *step-by-step* guide to make the scenarios provided executable. This should give you the general idea of the tool,
3. play around and write the *scenarios* for another user story yourself. Look at the *whole picture*: can you make better user stories this way?

At the end we reserve half an hour to discuss with other participants and *compare* your tool with the tools they used.

SETUP

PREREQUISITES

- Eclipse
- Java
- Maven
- Maven-eclipse-plugin: http://download.eclipse.org/technology/m2e/releases

INSTALLATION ECLIPSE PLUGIN

Eclipse installer:

- In the menu choose *Help* | *Install New Software …*
- Click *Add…* to add a new software site
- Enter `Cucumber` as *Name* and http://cucumber.github.com/cucumber-eclipse/update-site as *Location*.
- Choose *OK*
- Select *Cucumber Eclipse Plugin* and choose *Next*
- *Next*
- Accept the license agreement and choose *Finish*
- Ignore the warning and choose *OK*
- Choose *Yes* to restart Eclipse

CREATE ECLIPSE PROJECT

1. Create a new workspace: *File* | *Switch Workspace* | *Other …*
2. Via your file system, unzip the bmi-application you received on the USB-stick to this workspace
3. Import de bmi-application in *Eclipse*:
    - *File* | *Import…* | *Maven* | *Existing Maven Projects*
    - Click *Next*
    - For the field *Root Directory:* browse to the folder bmi-app and choose *Open*
    - The pom.xml should appear in the *Projects* list
    - Click *Finish*

    The project is created. In the root of your project, you will find the *pom.xml*. Click on the file and choose *Run As | Maven install* to check if you can build the application.

4. Edit the *pom.xml*. Add the plugins and dependencies needed for *Cucumber*. See Attachment 1: Additions to pom.xml.

DEVELOPMENT

INTRODUCTION: THE GENERAL IDEA

To write executable specifications with *Cucumber*, you always have to write 3 types of files:

1. The **feature**: a plain text file containing the specifications in Given-When-Then style. The filename is in CamelCase and the extension is .feature

   Example: `ShowPatientDetails.feature`

2. A **configuration file**: a special kind of Java test class that is used to run the steps as a JUnit test class. The filename is the same name as the name of the story + TestRunner and with extension .java

   Example: `ShowPatientDetailsTestRunner.java`

3. The **story steps**: a Java test class containing the java code to make each Given-, When- or Then- step executable. The filename is the same name as the name of the story + Steps and with extension .java

   Example: `ShowPatientDetailsSteps.java`

STEP 1: WRITE SPECIFICATIONS

1. Create a package `org.ucll.demo.service` in the test folder *src/test/resources*.
2. Create a new feature:
   - Choose *New* | *Other…* | *General* | *File*
   - Click *Next*
   - Enter the name of the feature, i.e. `ShowPatientDetails.feature`
     - Make sure that the extension is `.feature`
   - Click *Finish*
   - The feature is created
3. Add the story and the scenario given in Attachment 2: Specifications_1.txt.
4. Choose *Save*.

```
Feature: Show patient details

In order to check the physical condition of a patient
As a caretaker
I want to consult his/her personal details

    Scenario: the personal details of a registered patient are given
        Given a patient with the social security number 93051822361, gender male and birthdate 1993-05-18
        And on 2000-04-10 the patient had a length of 180 cm and a weight of 75000 gr
        And the patient is registered
        When I ask for the details of the patient using his social security number
        Then the personal details social security number, gender and birthdate are given
        And the examination details length, weight and last examination date are given
        And the calculated bmi 23.15 is given
```

Figure 1. Example feature show patient details

STEP 2: CONFIGURATION FILE

1. Create in the package `org.ucll.demo.service` in the test folder *src/test/java* a Java class with the configuration, i.e. `ShowPatientDetailsTestRunner.java`.

2. Add the following before the name of the class:

```
@RunWith(Cucumber.class)
```

   and add the import statements:

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

3. Add the following between the name of the class and the @RunWith annotation

```
@CucumberOptions(
        features={"src/test/resources/org/ucll/demo/service/ShowPatient
        Details.feature"},
        plugin={"html:target/cucumber"}
)
```

   and add the import statement:

```
import cucumber.api.CucumberOptions;
```

EXECUTE SPECIFICATIONS

Select the configuration file, right mouse click and choose *Run as JUnit Test.*

In the **output console** you can see the result:

- The number of scenarios and steps ran
- For each scenario, the steps with the current status
- For each step an empty Java method with annotation is generated. The annotations are used to:
    - map the method to a step in the story
    - say the step is pending (=no useful test is executed)

```
1 Scenarios ([33m1 undefined[0m)
7 Steps ([33m7 undefined[0m)
0m0.000s


You can implement missing steps with the snippets below:

@Given("^a patient with the social security number (\\d+), gender male and birthdate (\\d+)-(\\d+)-(\\d+)$")
public void a_patient_with_the_social_security_number_gender_male_and_birthdate(int arg1, int arg2, int arg3, int arg4) throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Given("^on (\\d+)-(\\d+)-(\\d+) the patient had a length of (\\d+) cm and a weight of (\\d+) gr$")
public void on_the_patient_had_a_length_of_cm_and_a_weight_of_gr(int arg1, int arg2, int arg3, int arg4, int arg5) throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Given("^the patient is registered$")
public void the_patient_is_registered() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@When("^I ask for the details of the patient using his social security number$")
public void i_ask_for_the_details_of_the_patient_using_his_social_security_number() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Then("^the personal details social security number, gender and birthdate are given$")
public void the_personal_details_social_security_number_gender_and_birthdate_are_given() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
```

Figure 2. Example console output

In the **JUnit console** you can see the result:

- The number of scenarios and steps ran
- For each scenario, the steps with the current status

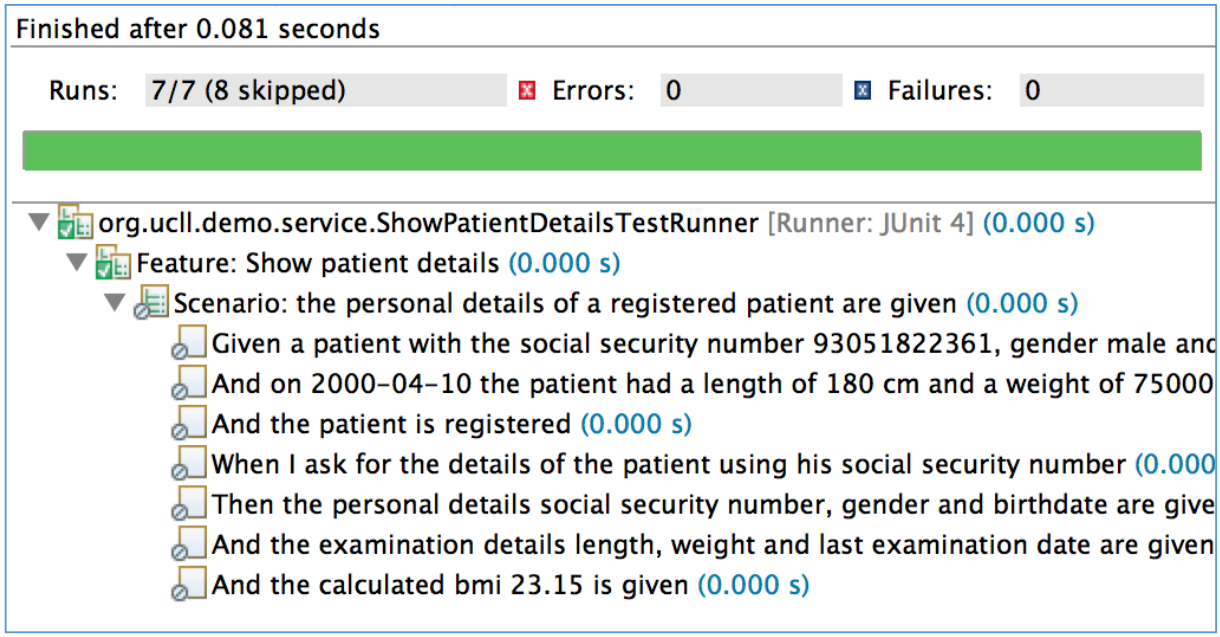

Figure 3. Example JUnit output

STEP 3: CREATE THE STEPS FILE

1. Create a package `org.ucll.demo.service` in the test folder *src/test/java*.
    - The package name must be the same as of the package in the folder *src/test/resources*.
2. Create a Java class for the story steps, i.e. `ShowPatientDetailsSteps.java`.

MAKE THE STEPS EXECUTABLE

1. Copy the method headings outputted in the console to the Steps-file (`ShowPatientDetailsSteps.java`).
2. Add the import statements:

```
import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;
```

3. Implement each step-method:
    - remove the *@Pending*
    - call the necessary code to connect to your actual domain classes…

```
public class ShowPatientDetailsSteps {

    private String socialSecurityNumber;
    private Gender gender;
    private Date birthDate;
    private PersonDetail patient;

    private static final SimpleDateFormat DATE_FORMATTER = new SimpleDateFormat("yyyy-MM-dd");

    @Given("^a patient with the social security number (\\d+), gender male and birthdate (\\d+)-(\\d+)-(\\d+)$")
    public void a_patient_with_the_social_security_number_gender_male_and_birthdate(int arg1, int arg2, int arg3, int arg4) throws Throwable {
        this.socialSecurityNumber = ""+arg1+"";
        this.gender = Gender.MALE;
        this.birthDate = DATE_FORMATTER.parse(""+arg2+"-"+arg3+"-"+arg4+"");
        patient = new PersonDetail(this.socialSecurityNumber, this.gender, this.birthDate);
    }

    @Given("^on (\\d+)-(\\d+)-(\\d+) the patient had a length of (\\d+) cm and a weight of (\\d+) gr$")
    public void on_the_patient_had_a_length_of_cm_and_a_weight_of_gr(int arg1, int arg2, int arg3, int arg4, int arg5) throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @Given("^the patient is registered$")
    public void the_patient_is_registered() throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @When("^I ask for the details of the patient using his social security number$")
    public void i_ask_for_the_details_of_the_patient_using_his_social_security_number() throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @Then("^the personal details social security number, gender and birthdate are given$")
    public void the_personal_details_social_security_number_gender_and_birthdate_are_given() throws Throwable {
        // Write code here that turns the phrase above into concrete actions
```

Figure 4. Example implementation of the first step

4. Run the Java story configuration again as a JUnit Test. The console and JUnit output has changed:
    - The Java methods have disappeared
    - The step you implemented is not PENDING anymore

```
1 Scenarios ([33m1 pending[0m)
7 Steps ([36m5 skipped[0m, [33m1 pending[0m, [32m1 passed[0m)
0m0.101s

cucumber.api.PendingException: TODO: implement me
        at org.ucll.demo.service.ShowPatientDetailsSteps.on_the_patient_had_a_length_of_cm_and_a_weight_of_gr(ShowPatientDetailsSteps.jav
        at *.And on 2000-04-10 the patient had a length of 180 cm and a weight of 75000 gr(src/test/resources/org/ucll/demo/service/Showl
```

Figure 5.Example console output after implementing a step

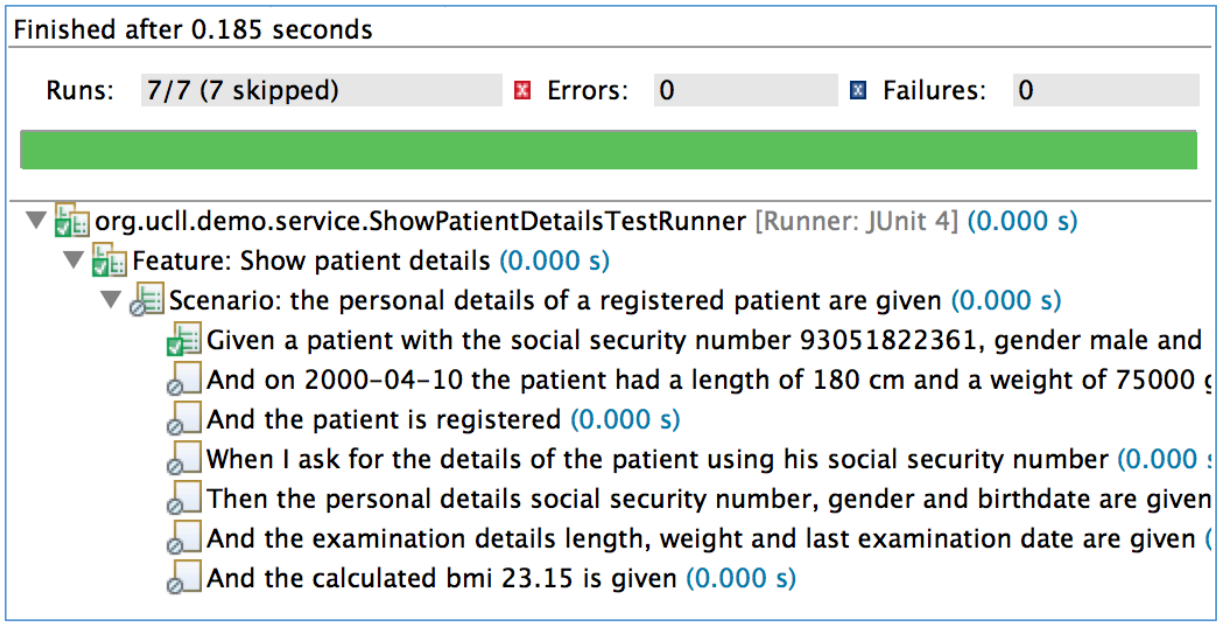

Figure 6. Example JUnit output after implementing a step

5. Implement all the steps until you have a successful test

See *Attachment 3: Example scenario* for a first implementation of all the steps

CONSULT THE HTML REPORT

Run the Maven build:

- Select your project
- Right mouse click
- Choose *Run As* | *Maven install*
  The build will be executed: the java code will be compiled, tested, … outside Eclipse

Consult the report:

- Refresh your project. You will find a folder *target*.
- In the subfolder *target/cucumber*, you can find website-pages created by Cucumber.
- Open the file `index.html` in your browser. Try to navigate to the html report of your story.

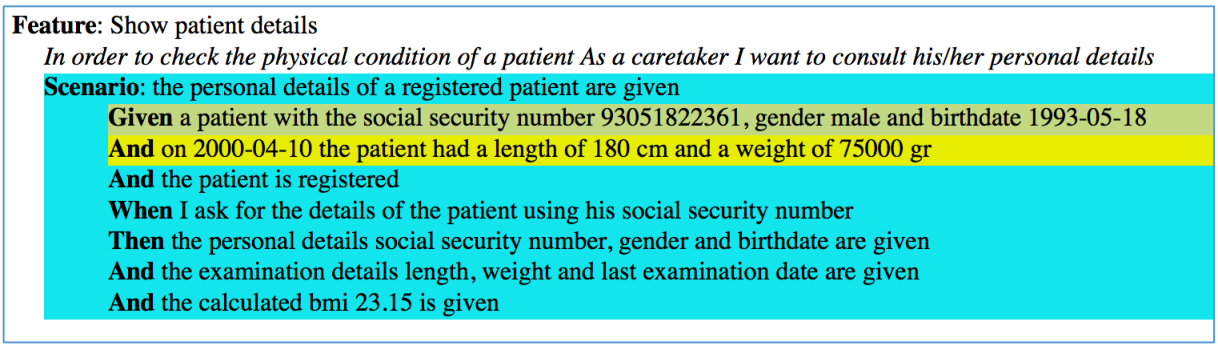

Figure 7. Example HTML report

EXTRA

FINETUNE PARAMETERS

The method headings in our Steps-file are not very nice, with only int or double types of parameters. We can refactor this to make them more reusable, by redefining the steps in the feature file and by using parameters:

1. In the Given step set the examples that you want to use as String inputs between “”

```
Feature: Show patient details

In order to check the physical condition of a patient
As a caretaker
I want to consult his/her personal details

    Scenario: the personal details of a registered patient are given
        Given a patient with the social security number "93051822361", gender "male" and birthdate "1993-05-18"
        And on "2000-04-10" the patient had a length of 180 cm and a weight of 75000 gr
        And the patient is registered
        When I ask for the details of the patient using his social security number
        Then the personal details social security number, gender and birthdate are given
        And the examination details length, weight and last examination date are given
        And the calculated bmi 23.15 is given
```

Figure 8. The usage of more parameters in the feature

2. Run the test.

```
You can implement missing steps with the snippets below:

@Given("^a patient with the social security number \"([^\"]*)\", gender \"([^\"]*)\" and birthdate \"([^\"]*)\"$")
public void a_patient_with_the_social_security_number_gender_and_birthdate(String arg1, String arg2, String arg3) throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Given("^on \"([^\"]*)\" the patient had a length of (\\d+) cm and a weight of (\\d+) gr$")
public void on_the_patient_had_a_length_of_cm_and_a_weight_of_gr(String arg1, int arg2, int arg3) throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}
```

Figure 9. Extra parameters are added to the steps now

3. Copy the new step to implement in the steps class.
4. Rename the parameters of the method into readable parameter names.

   Example: change `arg1` into `socialSecurityNumber`

5. Change the types of the parameters if you want in order to make your life easier. For example so that you can easier parse the date by using the annotation @Format

   Example: change String `arg3` into `@Format("yyyy-MM-dd") Date date`

6. Use the method parameters in your method body.

```
@Given("^a patient with the social security number \"([^\"]*)\", gender \"([^\"]*)\" and birthdate \"([^\"]*)\"$")
public void a_patient_with_the_social_security_number_gender_and_birthdate(String socialSecurityNumber, String gender,
        @Format("yyyy-MM-dd") Date date) throws Throwable {
    this.socialSecurityNumber = socialSecurityNumber;
    this.gender = Gender.valueOf(gender.toUpperCase());
    this.birthDate = date;
    patient = new PersonDetail(this.socialSecurityNumber, this.gender, this.birthDate);
}
```

Figure 10. The usage of parameters

1. Have a look at the implementation of the "The calculated bmi 23.15 is given" step. The strange thing here is that the double 23.15 from the specification is translated to two integer parameters. As such we need to parse it in our code to a double, this is not the best way to implement.

```
@Then("^the calculated bmi (\\d+)\\.(\\d+) is given$")
public void the_calculated_bmi_is_given(int arg1, int arg2) throws Throwable {
    assertEquals(Double.parseDouble(""+arg1+"."+arg2+""), detailsRetrieved.getBmi(), 0.0);
}
```

Figure 11. A double from the specification is translated to two integer values.

2. We can refactor this by using another and better regex for the parameter in order to make it a double right away.

```
@Then("^the calculated bmi ((?:\\d|\\.)+) is given$")
public void the_calculated_bmi_is_given(double bmi) throws Throwable {
    assertEquals(bmi, detailsRetrieved.getBmi(), 0.0);
}
```

Figure 12. Using another regex makes it a double right away.

REUSE STEPS

Identical steps:

1. Add the scenario's given in Attachment 4: Specifications_2.txt
2. Run the test. In the console you can see the *@Given* and *@Then* steps are created, but there is no suggestion for the *@When* steps. That's because Cucumber saw we can reuse the *@When* step from the previous scenario…

```
You can implement missing steps with the snippets below:

@Given("^a patient with the social security number \"([^\"]*)\"$")
public void a_patient_with_the_social_security_number(String arg1) throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Given("^on \"([^\"]*)\" the patient had a length of (\\d+) cm and a weight of (\\d+) gr but these data were added later$")
public void on_the_patient_had_a_length_of_cm_and_a_weight_of_gr_but_these_data_were_added_later(String arg1, int arg2, int arg3) t
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Then("^the length (\\d+), weight (\\d+), and date of the examination \"([^\"]*)\" are given$")
public void the_length_weight_and_date_of_the_examination_are_given(int arg1, int arg2, String arg3) throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@Then("^the calculated bmi (\\d+)\\.(\\d+) is based on these data$")
public void the_calculated_bmi_is_based_on_these_data(int arg1, int arg2) throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}
```

Figure 13. Example console output adding a second story – existing steps are left out

SCENARIO'S WITH EXPECTED EXCEPTIONS

1. Add the scenarios given in Attachment 5: Specifications_3.txt.
2. Declare an instance variable: `private boolean errorThrown;`
3. In the methods for the *@When* steps, where an exception might be thrown*:*
    - put a try-catch block around the code.
    - In the catch block you can set the instance variable `errorThrown` to `true`;
4. In the *@Then* steps where you want to check if an exception is thrown, you can check the value of the instance variable `errorThrown` …

DATA TABLES

1. Add the scenario given in *Attachment 6: Specifications_4.txt*. This scenario is different from the previous ones:
    - It must be a Scenario Outline instead of a Scenario
    - Instead of concrete values, you can see parameter names enclosed with < and >
    - After the scenario, you can see a table of examples. Columns are separated with the '|' character.

```
Scenario Outline: the bmi is rounded to 2 digits
    Given a patient that is registered with a length <length> cm and weight <weight> gr
    When I ask for the details of the patient
    Then the bmi <bmi> is given rounded to two digits

    Examples:
        |length |weight |bmi    |
        |160    |65000  |25.39  |
        |160    |65001  |25.39  |
        |160    |65009  |25.39  |
        |180    |75000  |23.15  |
        |180    |75009  |23.15  |
```

Figure 14. Table with examples

2. Run as JUnit test and copy the new missing steps from the console output to your Steps-file.
3. In these steps, you also use parameters as described in one of the above paragraphs
4. Run again as JUnit test.
5. Now you can see that this Scenario is ran for all the examples defined in the data table.

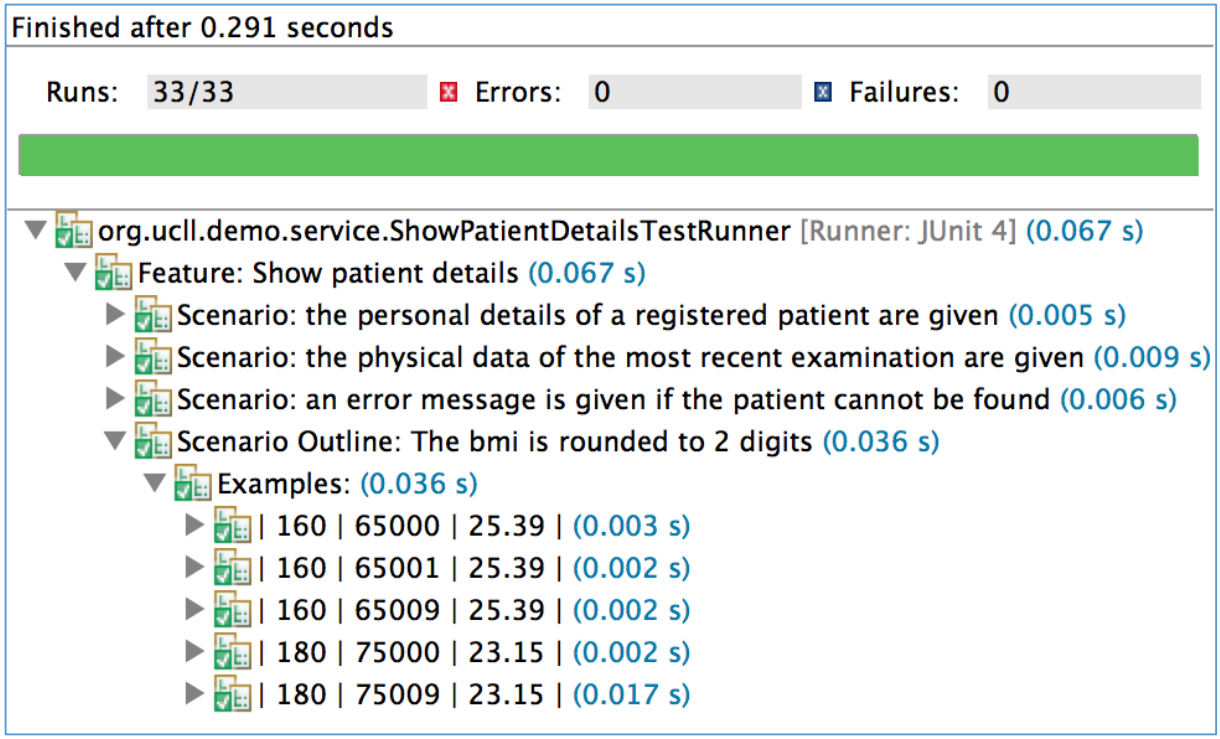

Figure 15. JUnit output for an example table

See in *Attachment 7: Example steps after implementing the 4 scenarios* for an example implementation of all scenarios

AND THEN …

Write specifications for the user story *Add physical examination data*.

1. How easy is it to focus on the content, without thinking about technical aspects?
2. Investigate other possibilities of *Cucumber*. Look at features not described in this manual.
    - Can you avoid duplicate methods using a setup or teardown method?
    - How does *Property-based testing* work?
    - …

In the attachments you can find an example of *Cucumber* used in combination with *Mockito* and *Selenium*.

ATTACHMENTS

ATTACHMENT 1: ADDITIONS TO POM.XML

```
<project ...
    <properties>
        <cukes.version>1.2.4</cukes.version>       <cucumber.reporting.version>0.2.1</cucumber.reporting.version>
        ...
    </properties>

    <dependencies>
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-picocontainer</artifactId>
            <version>${cukes.version}</version>
        </dependency>
        <dependency>
            <groupId>net.masterthought</groupId>
            <artifactId>cucumber-reporting</artifactId>
            <version>${cucumber.reporting.version}</version>
        </dependency>
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-junit</artifactId>
            <version>${cukes.version}</version>
        </dependency>
        ...
    </dependencies>
</project>
```

ATTACHMENT 2: SPECIFICATIONS_1.TXT

```
Feature: Show patient details

In order to check the physical condition of a patient
As a caretaker
I want to consult his/her personal details

Scenario: the personal details of a registered patient are given
Given a patient with the social security number 93051822361, gender male and birthdate 1993-05-18
And on 2000-04-10 the patient had a length of 180 cm and a weight of 75000 gr
And the patient is registered
When I ask for the details of the patient using his social security number
Then the personal details social security number, gender and birthdate are given
And the examination details length, weight and last examination date are given
And the calculated bmi 23.15 is given
```

ATTACHMENT 3: EXAMPLE SCENARIO

```
package org.ucll.demo.service;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.ucll.demo.domain.Gender;
import org.ucll.demo.service.api.java.PersonServiceJavaApi;
import org.ucll.demo.service.api.java.to.ExaminationDetail;
import org.ucll.demo.service.api.java.to.PersonDetail;

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

public class ShowPatientDetailsSteps {

	private PersonServiceJavaApi service = new PersonServiceJavaApi();

	private String socialSecurityNumber;
	private Gender gender = Gender.MALE;
	private Date birthDate;
	private int length;
	private int weight;
	private Date examinationDate;

	private PersonDetail detailsRetrieved;

	private PersonDetail patient;

	private static final SimpleDateFormat DATE_FORMATTER = new SimpleDateFormat("yyyy-MM-dd");

	@Given("^a patient with the social security number (\\d+), gender male and birthdate (\\d+)-(\\d+)-(\\d+)$")
	public void a_patient_with_the_social_security_number_gender_male_and_birthdate(int arg1, int arg2, int arg3, int arg4) throws Throwable {
		this.socialSecurityNumber = ""+arg1+"";
		this.gender = Gender.MALE;
```

```
        this.birthDate = DATE_FORMATTER.parse(""+arg2+"-"+arg3+"-"+arg4+"");
        patient = new PersonDetail(this.socialSecurityNumber, this.gender, this.birthDate);
}

@Given("^on (\\d+)-(\\d+)-(\\d+) the patient had a length of (\\d+) cm and a weight of (\\d+) gr$")
public void on_the_patient_had_a_length_of_cm_and_a_weight_of_gr(int arg1, int arg2, int arg3, int arg4, int arg5) throws Throwable {
        this.examinationDate = DATE_FORMATTER.parse(""+arg1+"-"+arg2+"-"+arg3+"");
        this.length = arg4;
        this.weight = arg5;
        patient.setExaminationDetail(new ExaminationDetail(this.length, this.weight, this.examinationDate));
}

@Given("^the patient is registered$")
public void the_patient_is_registered() throws Throwable {
        service.addPerson(patient);
}

@When("^I ask for the details of the patient using his social security number$")
public void i_ask_for_the_details_of_the_patient_using_his_social_security_number() throws Throwable {
        detailsRetrieved = service.getPerson(this.socialSecurityNumber);
}

@Then("^the personal details social security number, gender and birthdate are given$")
public void the_personal_details_social_security_number_gender_and_birthdate_are_given() throws Throwable {
        assertEquals(this.socialSecurityNumber, detailsRetrieved.getSocialSecurityNumber());
        assertEquals(this.gender, detailsRetrieved.getGender());
        assertTrue(differNoMoreThanFewSeconds(this.birthDate, detailsRetrieved.getBirthdate()));
}

@Then("^the examination details length, weight and last examination date are given$")
public void the_examination_details_length_weight_and_last_examination_date_are_given() throws Throwable {
        ExaminationDetail examinationDetails = service.getPerson(this.socialSecurityNumber).getExaminationDetail();
        assertEquals(this.length, examinationDetails.getLength());
        assertEquals(this.weight, examinationDetails.getWeight());
        assertTrue(differNoMoreThanFewSeconds(this.examinationDate, examinationDetails.getExaminationDate()));
}

@Then("^the calculated bmi (\\d+)\\.(\\d+) is given$")
public void the_calculated_bmi_is_given(int arg1, int arg2) throws Throwable {
        assertEquals(Double.parseDouble(""+arg1+"."+arg2+""), detailsRetrieved.getBmi(), 0.0);
}
```

```
        private boolean differNoMoreThanFewSeconds(Date date, Date otherDate) {
                return date.compareTo(otherDate) <= 0 && date.compareTo(otherDate) >= -5;
        }

}
```

ATTACHMENT 4: SPECIFICATIONS_2.TXT

```
Scenario: the physical data of the most recent examination are given
        Given a patient with the social security number "93051822361"
        And on "2000-04-17" the patient had a length of 180 cm and a weight of 80000 gr
        And the patient is registered
        And on "2000-04-10" the patient had a length of 180 cm and a weight of 75000 gr but these data were added later
        When I ask for the details of the patient using his social security number
        Then the length 180, weight 80000, and date of the examination "17-04-2000" are given
        And the calculated bmi 24.69 is based on these data
```

ATTACHMENT 5: SPECIFICATIONS_3.TXT

```
Scenario: an error message is given if the patient cannot be found
        Given a patient that is not registered
        When I ask for the details of the patient using his social security number
        Then an error message is given
        And no details are given
```

ATTACHMENT 6: SPECIFICATIONS_4.TXT

```
Scenario Outline: the bmi is rounded to 2 digits
        Given a patient that is registered with a length <length> cm and weight <weight> gr
        When I ask for the details of the patient
        Then the bmi <bmi> is given rounded to two digits

        Examples:
                |length |weight |bmi    |
                |160    |65000  |25.39  |
                |160    |65001  |25.39  |
                |160    |65009  |25.39  |
                |180    |75000  |23.15  |
                |180    |75009  |23.15  |
```

ATTACHMENT 7: EXAMPLE STEPS AFTER IMPLEMENTING THE 4 SCENARIOS

```
package org.ucll.demo.service;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.ucll.demo.domain.Gender;
import org.ucll.demo.service.api.java.PersonServiceJavaApi;
import org.ucll.demo.service.api.java.to.ExaminationDetail;
import org.ucll.demo.service.api.java.to.PersonDetail;

import cucumber.api.Format;
import cucumber.api.java.After;
import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;

import static org.junit.Assert.*;

public class ShowPatientDetailsSteps {

        private PersonServiceJavaApi service = new PersonServiceJavaApi();

        private String socialSecurityNumber;
        private Gender gender = Gender.MALE;
        private Date birthDate;
        private int length;
        private int weight;
        private Date examinationDate;

        private PersonDetail detailsRetrieved;

        private boolean errorThrown = false;
        private PersonDetail patient;

        private static final String REGISTERED_SOCIAL_SECURITY_NUMBER = "93051822361";
        private static final String UNKNOWN_SOCIAL_SECURITY_NUMBER = "93051822363";
        private static final SimpleDateFormat DATE_FORMATTER = new SimpleDateFormat("yyyy-MM-dd");

        @After
```

```
	public void teardown () {
		clearTestData();
	}

	@Given("^a patient with the social security number \"([^\"]*)\", gender \"([^\"]*)\" and birthdate \"([^\"]*)\"$")
	public void a_patient_with_the_social_security_number_gender_and_birthdate(String socialSecurityNumber, String gender, @Format("yyyy-MM-
dd") Date date) throws Throwable {
		this.socialSecurityNumber = socialSecurityNumber;
		this.gender = Gender.valueOf(gender.toUpperCase());
		this.birthDate = date;
		patient = new PersonDetail(this.socialSecurityNumber, this.gender, this.birthDate);
	}

	@Given("^the patient is registered$")
	public void the_patient_is_registered() throws Throwable {
		service.addPerson(patient);
	}

	@When("^I ask for the details of the patient using his social security number$")
	public void i_ask_for_the_details_of_the_patient_using_his_social_security_number() throws Throwable {
		try {
			detailsRetrieved = service.getPerson(this.socialSecurityNumber);
		}
		catch (Exception exc) {
			errorThrown = true;
		}
	}

	@Then("^the personal details social security number, gender and birthdate are given$")
	public void the_personal_details_social_security_number_gender_and_birthdate_are_given() throws Throwable {
		assertEquals(this.socialSecurityNumber, detailsRetrieved.getSocialSecurityNumber());
		assertEquals(this.gender, detailsRetrieved.getGender());
		assertTrue(differNoMoreThanFewSeconds(this.birthDate, detailsRetrieved.getBirthdate()));
	}

	@Then("^the examination details length, weight and last examination date are given$")
	public void the_examination_details_length_weight_and_last_examination_date_are_given() throws Throwable {
		ExaminationDetail examinationDetails = service.getPerson(this.socialSecurityNumber).getExaminationDetail();
		assertEquals(this.length, examinationDetails.getLength());
		assertEquals(this.weight, examinationDetails.getWeight());
		assertTrue(differNoMoreThanFewSeconds(this.examinationDate, examinationDetails.getExaminationDate()));
	}
```

```
	@Then("^the calculated bmi ((?:\\d|\\.)+) is given$")
	public void the_calculated_bmi_is_given(double bmi) throws Throwable {
		assertEquals(bmi, detailsRetrieved.getBmi(), 0.0);
	}

	@Given("^a patient with the social security number \"([^\"]*)\"$")
	public void a_patient_with_the_social_security_number(String socialSecurityNumber) throws Throwable {
		this.socialSecurityNumber = socialSecurityNumber;
		this.gender = Gender.MALE;
		this.birthDate = DATE_FORMATTER.parse("1993-05-18");
		patient = new PersonDetail(this.socialSecurityNumber, this.gender, this.birthDate);
	}

	@Given("^on \"([^\"]*)\" the patient had a length of (\\d+) cm and a weight of (\\d+) gr$")
	public void the_patient_had_an_examination_with_a_length_of_cm_and_a_weight_of_gr_on_the_date(@Format("yyyy-MM-dd") Date examinationDate,
int length, int weight) throws Throwable {
		this.examinationDate = examinationDate;
		this.length = length;
		this.weight = weight;
		patient.setExaminationDetail(new ExaminationDetail(length, weight, this.examinationDate));
	}

	@Given("^on \"([^\"]*)\" the patient had a length of (\\d+) cm and a weight of (\\d+) gr but these data were added later$")
	public void a_new_examination_on_the_patient_had_a_length_of_cm_and_a_weight_of_gr(@Format("yyyy-MM-dd") Date examinationDate, int length,
int weight) throws Throwable {
		service.addExamination(new ExaminationDetail(length, weight, examinationDate), this.socialSecurityNumber);
	}


	@Then("^the length (\\d+), weight (\\d+), and date of the examination date \"([^\"]*)\" are given$")
	public void the_length_weight_and_date_of_the_examination_date_are_given(int length, int weight, @Format("yyyy-MM-dd") Date
examinationDate) throws Throwable {
		assertEquals(length, detailsRetrieved.getExaminationDetail().getLength());
		assertEquals(weight, detailsRetrieved.getExaminationDetail().getWeight());
		assertTrue(differNoMoreThanFewSeconds(examinationDate, detailsRetrieved.getExaminationDetail().getExaminationDate()));
	}

	@Then("^the calculated bmi ((?:\\d|\\.)+) is based on these data$")
	public void the_calculated_bmi_is_based_on_these_data(double bmi) throws Throwable {
		assertEquals(bmi, detailsRetrieved.getBmi(), 0.0);
	}
```

```
	@Given("^a patient that is not registered$")
	public void a_patient_that_is_not_registered() throws Throwable {
		this.socialSecurityNumber = UNKNOWN_SOCIAL_SECURITY_NUMBER;
	}

	@Then("^an error message is given$")
	public void an_error_message_is_given() throws Throwable {
		assertTrue(errorThrown);
	}

	@Then("^no details are given$")
	public void no_details_are_given() throws Throwable {
		assertNull(detailsRetrieved);
	}

	@Given("^a patient that is registered with a given (\\d+) and (\\d+)$")
	public void a_patient_that_is_registered_with_a_given_and(int length, int weight) throws Throwable {
		this.socialSecurityNumber = REGISTERED_SOCIAL_SECURITY_NUMBER;
		this.gender = Gender.MALE;
		this.birthDate = DATE_FORMATTER.parse("1993-05-18");
		examinationDate = DATE_FORMATTER.parse("2000-04-10");
		this.length = length;
		this.weight = weight;

		PersonDetail patient = new PersonDetail(this.socialSecurityNumber, this.gender, this.birthDate, new ExaminationDetail(length,
weight, this.examinationDate));
		service.addPerson(patient);
	}

	@When("^I ask for details of the patient$")
	public void i_ask_for_details_of_the_patient() throws Throwable {
		detailsRetrieved = service.getPerson(this.socialSecurityNumber);
	}

	@Then("^the ((?:\\d|\\.)+) given is rounded to two digits$")
	public void the_given_is_rounded_to_digits(double bmi) throws Throwable {
		assertEquals(bmi, detailsRetrieved.getBmi(), 0.0);
	}

	private void clearTestData() {
		//TODO replace with DBUnit, ...
```

```
                service.deletePerson(this.socialSecurityNumber);
        }

        private boolean differNoMoreThanFewSeconds(Date date, Date otherDate) {
                return date.compareTo(otherDate) <= 0 && date.compareTo(otherDate) >= -5;
        }

}
```

ATTACHMENT 8: EXAMPLE USING MOCKITO

```
package org.ucll.demo.service.api.java;

import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertTrue;
import static org.mockito.BDDMockito.given;
import static org.mockito.BDDMockito.then;
import static org.mockito.Matchers.any;
import static org.mockito.Mockito.never;
import static org.mockito.MockitoAnnotations.initMocks;

import org.mockito.Mock;
import org.ucll.demo.domain.Person;
import org.ucll.demo.service.PersonService;
import org.ucll.demo.service.api.java.assembler.ExaminationToAssembler;
import org.ucll.demo.service.api.java.assembler.PersonToAssembler;
import org.ucll.demo.service.api.java.to.PersonDetail;

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

public class PersonServiceJavaApiSteps {
	@Mock
	private PersonService mockPersonService;
	@Mock
	private ExaminationToAssembler mockExaminationToAssembler;
	@Mock
	private PersonToAssembler mockPersonToAssembler;
	@Mock
	private Person mockPerson;
	@Mock
	private PersonDetail mockPersonDetail;

	private PersonServiceJavaApi service;

	private String socialSecurityNumber;
	private PersonDetail patientRetrieved;
	private boolean exceptionThrown = false;
```

```
public PersonServiceJavaApiSteps () {
        initMocks(this);
        socialSecurityNumber = null;
        service = new PersonServiceJavaApi(mockPersonService, mockExaminationToAssembler, mockPersonToAssembler);
}

@Given("^the social security number \"([^\"]*)\" of a patient who is registered$")
public void the_social_security_number_of_a_patient_who_is_registered(String socialSecurityNumber) throws Throwable {
        this.socialSecurityNumber = socialSecurityNumber;
        given(mockPersonService.getPerson(socialSecurityNumber)).willReturn(mockPerson);
        given(mockPersonToAssembler.createPersonDetailTo(mockPerson)).willReturn(mockPersonDetail);
}

@When("^I get the details of the patient using his social security number$")
public void i_get_the_details_of_the_patient_using_his_social_security_number() throws Throwable {
        try{
                patientRetrieved = service.getPerson(socialSecurityNumber);
        } catch(Exception e){
                exceptionThrown = true;
        }
}

@Then("^a persondetail object is returned$")
public void a_persondetail_object_is_returned() throws Throwable {
        then(mockPersonService).should().getPerson(socialSecurityNumber);
        then(mockPersonToAssembler).should().createPersonDetailTo(any(Person.class));
        assertNotNull(patientRetrieved);
}

@Given("^the social security number \"([^\"]*)\" of a patient who is not registered$")
public void the_social_security_number_of_a_patient_who_is_not_registered(String socialSecurityNumber) throws Throwable {
        this.socialSecurityNumber = socialSecurityNumber;
        given(mockPersonService.getPerson(socialSecurityNumber)).willReturn(null);
}

@Given("^no social security number$")
public void no_social_security_number() throws Throwable {
        given(mockPersonService.getPerson(null)).willThrow(new RuntimeException());
}
```

```
	@When("^I get the details of the patient using no social security number$")
	public void i_get_the_details_of_the_patient_using_no_social_security_number() throws Throwable {
		try{
			patientRetrieved = service.getPerson(null);
		} catch(Exception e){
			exceptionThrown = true;
		}
	}

	@Then("^an exception is thrown$")
	public void an_exception_is_thrown() throws Throwable {
		then(mockPersonService).should().getPerson(socialSecurityNumber);
		then(mockPersonToAssembler).should(never()).createPersonDetailTo(any(Person.class));
		assertTrue(exceptionThrown);
	}

}
```

ATTACHMENT 9: EXAMPLE USING SELENMIUM

```
package org.ucll.demo.ui;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.NoSuchElementException;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.ucll.demo.domain.Gender;
import org.ucll.demo.ui.pages.ExaminationDetailPage;
import org.ucll.demo.ui.pages.PersonDetailPage;
import org.ucll.demo.ui.pages.PersonOverviewPage;

import cucumber.api.Format;
import cucumber.api.java.After;
import cucumber.api.java.Before;
import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;

import static org.junit.Assert.*;

public class ShowPatientDetailsITSteps {

        private String registeredSocialSecurityNumber;
        private Date registeredBirthdate;
        private Gender registeredGender;
        private int registeredLength, registeredWeight;
        private Date registeredExaminationDate;

        private PersonDetailPage personDetailPage;
        private ExaminationDetailPage examinationDetailPage;
        private PersonOverviewPage personOverviewPage;

        private WebDriver driver;
        private final static String HOST_AND_PORT_URL = "http://localhost:8080";
        private final static String CONTEXT_URL = HOST_AND_PORT_URL + "/bmi";
        private static final SimpleDateFormat DATE_FORMATTER = new SimpleDateFormat("yyyy-MM-dd");
```

```
	private static final String SOCIAL_SECURITY_NUMBER = "93051822361";

	@Before
	public void setUp(){
		driver = new FirefoxDriver();
		driver.get(CONTEXT_URL);
	}

	@After
	public void clean(){
		driver.get(CONTEXT_URL);
		clearTestData();

		driver.quit();
	}

	@Given("^a patient with the social security number \"([^\"]*)\", gender \"([^\"]*)\" and birthdate \"([^\"]*)\"$")
	public void a_patient_with_the_social_security_number_gender_and_birthdate(String socialSecurityNumber, String gender, @Format("yyyy-MM-
dd") Date birthDate) throws Throwable {
		//TODO replace with DBUnit, ...
		registeredSocialSecurityNumber = socialSecurityNumber;
		registeredGender = Gender.valueOf(gender.toUpperCase());
		registeredBirthdate = birthDate;
	}

	@Given("^on \"([^\"]*)\" the patient had a length of (\\d+) cm and a weight of (\\d+) gr$")
	public void the_patient_had_an_examination_with_a_length_of_cm_and_a_weight_of_gr_on_the_date(@Format("yyyy-MM-dd") Date examinationDate,
int length, int weight) throws Throwable {
		//TODO replace with DBUnit, ...
		registeredLength = length;
		registeredWeight = weight;
		registeredExaminationDate = examinationDate;
	}

	@Given("^the patient is registered$")
	public void the_patient_is_registered() throws Throwable {
		personOverviewPage = new PersonOverviewPage(driver);
		personDetailPage = personOverviewPage.addPerson();
		personOverviewPage = personDetailPage.addPerson(registeredSocialSecurityNumber, registeredBirthdate, registeredGender,
registeredLength, registeredWeight, registeredExaminationDate);
	}
```

```
	@When("^I ask for the details of the patient using his social security number$")
	public void i_ask_for_the_details_of_the_patient_using_his_social_security_number() throws Throwable {
		personDetailPage = personOverviewPage.showPerson(registeredSocialSecurityNumber);
	}

	@Then("^the personal details social security number, gender and birthdate are given$")
	public void the_personal_details_social_security_number_gender_and_birthdate_are_given() throws Throwable {
		assertEquals(registeredSocialSecurityNumber, personDetailPage.getSocialSecurityNumber());
		assertEquals(registeredGender, personDetailPage.getGender());
		assertTrue(differNoMoreThanFewSeconds(registeredBirthdate, personDetailPage.getBirthdate()));
	}

	@Then("^the examination details length, weight and last examination date are given$")
	public void the_examination_details_length_weight_and_last_examination_date_are_given() throws Throwable {
		assertEquals(registeredLength, personDetailPage.getLength());
		assertEquals(registeredWeight, personDetailPage.getWeight());
		assertTrue(differNoMoreThanFewSeconds(registeredExaminationDate, personDetailPage.getExaminationDate()));
	}

	@Then("^the calculated bmi ((?:\\d|\\.)+) is given$")
	public void the_calculated_bmi_is_given(double bmi) throws Throwable {
		assertEquals(bmi, personDetailPage.getBmi(), 0.0);
	}

	@Given("^a patient with the social security number \"([^\"]*)\"$")
	public void a_patient_with_the_social_security_number(String socialSecurityNumber) throws Throwable {
		//TODO replace with DBUnit, ...
		registeredSocialSecurityNumber = socialSecurityNumber;
		registeredGender = Gender.MALE;
		registeredBirthdate = DATE_FORMATTER.parse("1993-05-18");
	}

	@Given("^on \"([^\"]*)\" the patient had a length of (\\d+) cm and a weight of (\\d+) gr but these data were added later$")
	public void a_new_examination_on_the_patient_had_a_length_of_cm_and_a_weight_of_gr(@Format("yyyy-MM-dd") Date examinationDate, int length,
int weight) throws Throwable {
		personDetailPage = personOverviewPage.showPerson(registeredSocialSecurityNumber);
		examinationDetailPage = personDetailPage.addExamination();
		personDetailPage = examinationDetailPage.addExamination(length, weight, examinationDate);
		personOverviewPage = personDetailPage.cancel();
	}
```

```
	@Then("^the length (\\d+), weight (\\d+), and date of the examination date \"([^\"]*)\" are given$")
	public void the_length_weight_and_date_of_the_examination_date_are_given(int length, int weight, @Format("yyyy-MM-dd") Date
examinationDate) throws Throwable {
		assertEquals(length, personDetailPage.getLength());
		assertEquals(weight, personDetailPage.getWeight());
		assertTrue(differNoMoreThanFewSeconds(examinationDate, personDetailPage.getExaminationDate()));
	}

	@Then("^the calculated bmi ((?:\\d|\\.)+) is based on these data$")
	public void the_calculated_bmi_is_based_on_these_data(double bmi) throws Throwable {
		assertEquals(bmi, personDetailPage.getBmi(), 0.0);
	}

	@Given("^a patient that is registered with a given (\\d+) and (\\d+)$")
	public void a_patient_that_is_registered_with_a_given_and(int length, int weight) throws Throwable {
		//TODO replace with DBUnit, ...
		registeredSocialSecurityNumber = SOCIAL_SECURITY_NUMBER;
		registeredGender = Gender.MALE;
		registeredBirthdate = DATE_FORMATTER.parse("1993-05-18");
		registeredLength = length;
		registeredWeight = weight;
		registeredExaminationDate = new Date();
		personOverviewPage = new PersonOverviewPage(driver);
		personDetailPage = personOverviewPage.addPerson();
		personOverviewPage = personDetailPage.addPerson(registeredSocialSecurityNumber, registeredBirthdate, registeredGender,
registeredLength, registeredWeight, registeredExaminationDate);
	}

	@When("^I ask for the details of the patient$")
	public void i_ask_for_details_of_the_patient() throws Throwable {
		personDetailPage = personOverviewPage.showPerson(registeredSocialSecurityNumber);
	}

	@Then("^the ((?:\\d|\\.)+) given is rounded to two digits$")
	public void the_given_is_rounded_to_two_digits(double bmi) throws Throwable {
		assertEquals(bmi, personDetailPage.getBmi(), 0.0);
	}

	private void clearTestData() {
		//TODO replace with DBUnit, ...
		personOverviewPage = new PersonOverviewPage(driver);
		try {
```

```
                        personOverviewPage.deletePerson();
                } catch (NoSuchElementException mightBeExpectedIfPersonWasNotRegistered) {
                }
        }

        private boolean differNoMoreThanFewSeconds(Date date, Date otherDate) {
                return date.compareTo(otherDate) <= 0 && date.compareTo(otherDate) >= -5;
        }

}
```

ATTACHMENT 10: SELENIUMPAGE CLASS

```
package org.ucll.demo.ui.pages;

import java.util.Calendar;
import java.util.Date;

import org.openqa.selenium.WebDriver;

public abstract class SeleniumPage {
	private final WebDriver driver;

	public SeleniumPage(WebDriver driver) {
		this.driver = driver;
	}

	public WebDriver getDriver() {
		return driver;
	}

	protected Date convertToDate(String dateAsString) {
		String[] dateParts = dateAsString.split("-");
		int year = Integer.parseInt(dateParts[0]);
		int month = Integer.parseInt(dateParts[1]);
		int day = Integer.parseInt(dateParts[2]);
		Calendar calendar = Calendar.getInstance();
		calendar.set(year, month - 1, day);
		return calendar.getTime();
	}

}
```

ATTACHMENT 11: PERSONOVERVIEWPAGE CLASS

```
package org.ucll.demo.ui.pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

public class PersonOverviewPage extends SeleniumPage {

	public PersonOverviewPage(WebDriver driver) {
		super(driver);
	}

	public PersonDetailPage showPerson(String id) {
		WebElement userLink = getDriver().findElement(By.partialLinkText(id));
		userLink.click();
		return new PersonDetailPage(getDriver());
	}

	public boolean isCurrentPage() {
		WebElement heading = getDriver().findElement(By.cssSelector("h2"));
		return heading.getText().equals("Persons");
	}

	public PersonDetailPage addPerson() {
		WebElement addButton = getDriver().findElement(By.id("personOverviewForm:addPersonButton"));
		addButton.click();
		return new PersonDetailPage(getDriver());
	}

	public PersonDetailPage deletePerson() { // TODO do better
		WebElement deleteLink = getDriver().findElement(By.id("personOverviewForm:personTable:0:delete"));
		deleteLink.click();
		return new PersonDetailPage(getDriver());
	}

}
```

ATTACHMENT 12: PERSONDETAILPAGE CLASS

```
package org.ucll.demo.ui.pages;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.Select;
import org.ucll.demo.domain.Gender;

public class PersonDetailPage extends SeleniumPage {
	private ExaminationFieldsPage examinationfieldsPage;

	public PersonDetailPage(WebDriver driver) {
		super(driver);
		examinationfieldsPage = new ExaminationFieldsPage(getDriver(), "personForm");
	}

	public PersonOverviewPage addPerson(String socialSecurityNumber, Date birthdate,
			Gender gender, int length, int weight, Date examinationDate) {
		WebElement socialSecurityNumberField = getDriver().findElement(By.id("personForm:number"));
		socialSecurityNumberField.clear();
		socialSecurityNumberField.sendKeys(socialSecurityNumber);

	WebElement birthdateField = getDriver().findElement(By.id("personForm:birthDate"));
	birthdateField.clear();
	birthdateField.sendKeys(new SimpleDateFormat("yyyy-MM-dd").format(birthdate));

		Select genderField = new Select(getDriver().findElement(By.id("personForm:gender")));
		genderField.selectByVisibleText("MALE");

		examinationfieldsPage.addExaminationData(length, weight, examinationDate);

	WebElement saveButton = getDriver().findElement(By.id("personForm:actionSave"));
	saveButton.click();

	return new PersonOverviewPage(getDriver());
```

```
}

public String getSocialSecurityNumber() {
        WebElement socialSecurityNumberField = getDriver().findElement(By.id("personForm:number"));
        return socialSecurityNumberField.getAttribute("value");
}

public Gender getGender() {
        Select genderField = new Select(getDriver().findElement(By.id("personForm:gender")));
        String genderAsString = genderField.getFirstSelectedOption().getText();

        return Gender.valueOf(genderAsString.toUpperCase());
}

public Date getBirthdate() {
        WebElement birhtdateField = getDriver().findElement(By.id("personForm:birthDate"));
        String dateAsString = birhtdateField.getAttribute("value");
        return convertToDate(dateAsString);
}

public int getLength() {
        return examinationfieldsPage.getLength();
}

public int getWeight() {
        return examinationfieldsPage.getWeight();
}

public Date getExaminationDate() {
        return examinationfieldsPage.getExaminationDate();
}

public double getBmi() {
        WebElement bmiField = getDriver().findElement(By.id("personForm:bmi"));
        String bmiAsString = bmiField.getAttribute("value");
        return Double.parseDouble(bmiAsString);
}

public ExaminationDetailPage addExamination() {
        WebElement buttonExamination = getDriver().findElement(By.id("personToExaminationForm:addExamination"));
        buttonExamination.click();
```

```
                return new ExaminationDetailPage(getDriver());
        }

        public PersonOverviewPage cancel() {
                WebElement buttonCancel = getDriver().findElement(By.id("personToExaminationForm:cancel"));
                buttonCancel.click();
                return new PersonOverviewPage(getDriver());
        }

}
```

ATTACHMENT 13: EXAMINATIONDETAILPAGE CLASS

```
package org.ucll.demo.ui.pages;

import java.util.Date;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

public class ExaminationDetailPage extends SeleniumPage {
        private ExaminationFieldsPage examinationfieldsPage;

        public ExaminationDetailPage(WebDriver driver) {
                super(driver);
                examinationfieldsPage = new ExaminationFieldsPage(getDriver(), "examinationForm");
        }

        public PersonDetailPage addExamination(int length, int weight, Date examinationDate) {
                examinationfieldsPage.addExaminationData(length, weight, examinationDate);

         WebElement saveButton = getDriver().findElement(By.id("examinationForm:saveExamination"));
         saveButton.click();

         return new PersonDetailPage(getDriver());
        }

}
```

ATTACHMENT 14: EXAMINATIONFIELDSPAGE CLASS

```
package org.ucll.demo.ui.pages;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

public class ExaminationFieldsPage extends SeleniumPage {
	private String formPrefix;

	public ExaminationFieldsPage(WebDriver driver, String formPrefix) {
		super(driver);
		this.formPrefix = formPrefix;
	}

	public void addExaminationData(int length, int weight, Date examinationDate) {
		WebElement lenghtField = getDriver().findElement(By.id(formPrefix + ":length"));
		lenghtField.clear();
		lenghtField.sendKeys(Integer.toString(length));

		WebElement weightField = getDriver().findElement(By.id(formPrefix + ":weight"));
		weightField.clear();
		weightField.sendKeys(Integer.toString(weight));

	 WebElement dateField = getDriver().findElement(By.id(formPrefix + ":examinationDate"));
	 dateField.clear();
	 dateField.sendKeys(new SimpleDateFormat("yyyy-MM-dd").format(examinationDate));
	}

	public int getLength() {
		WebElement lengthField = getDriver().findElement(By.id(formPrefix + ":length"));
		String lengthAsString = lengthField.getAttribute("value");
		return Integer.parseInt(lengthAsString);
	}

	public int getWeight() {
```

```
                WebElement weightField = getDriver().findElement(By.id(formPrefix + ":weight"));
                String weightAsString = weightField.getAttribute("value");
                return Integer.parseInt(weightAsString);
        }

        public Date getExaminationDate() {
                WebElement examinationDateField = getDriver().findElement(By.id(formPrefix + ":examinationDate"));
                String dateAsString = examinationDateField.getAttribute("value");
                return convertToDate(dateAsString);
        }

}
```