FROM TDD TO BDD AND BACK
HANDS ON LAB

GETTING STARTED GUIDE SCALATEST

Mieke Kemme and Elke Steegmans
Nov 4, 2015

TABLE OF CONTENTS

INTRODUCTION

SCALATEST

ScalaTest is a free, open-source testing toolkit for Scala and Java programmers.

Official Website: http://www.scalatest.org/

THIS LAB

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

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 *ScalaTest.*

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 `ScalaTest` as *Name* and http://download.scala-ide.org/sdk/lithium/e44/scala211/stable/site as *Location*.
- Choose *OK*
- Select *Scala IDE for Eclipse* and *Scala IDE plugins* 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 de 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 *ScalaTest*. See *Attachment 1: Additions to pom.xml'*.
5. Make it a Scala project:
    - Create a source folder for Scala: *File* | *New* | *Source Folder – Folder Name:* `src/test/scala`
    - Right click on project | *Configure* | *Add Scala nature*

DEVELOPMENT

INTRODUCTION: THE GENERAL IDEA

To write executable specifications with *ScalaTest*, you put everything in one Scala Class per story.

Example: `ShowPatientDetails`

1. **Headers** with the feature, story, steps, …
2. After each step, **Scala code** to make the step executable

STEP 1: WRITE SPECIFICATIONS

1. Create a package `org.ucll.demo.service` in the test folder *src/test/scala*.
2. Create a new Scala Class in this package:
   - Choose *New* | *Other…* | *Scala Wizards* | *Scala Class*
   - Click *Next*
   - Enter the name of the story, i.e. `org.ucll.demo.service.ShowPatientDetail`
   - Click *Finish*

   The story is created.
3. Open the file: right click on the file | *Open with* | *Scala Editor*.
4. Add the following after the name of the class:

   ```
   extends FeatureSpec with GivenWhenThen
   ```

   and add the import statements:

   ```
   import org.scalatest.FeatureSpec
   import org.scalatest.GivenWhenThen
   ```
5. Add the story and the scenario given in *Attachment 3: Specifications_1.txt*
   - In the body of the class, you can find the **feature**
   - In the body of the feature, you can find the **story**
   - After the story, you can find the **scenario**
   - In the body of the scenario, you can find the **steps**
   - Add the end, we added **`pending`**, as there is no code to execute yet

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

import org.scalatest.FeatureSpec
import org.scalatest.GivenWhenThen

class ShowPatientDetails extends FeatureSpec with GivenWhenThen {
  feature("Show patient details") {
    info("In order to check the physical condition of a patient")
    info("As a caretaker")
    info("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")
      pending
    })
  }
}
```

Figure 1. Example pending story

EXECUTE SPECIFICATIONS

Select the Scala class, right mouse click and choose *Run as ScalaTest - Test.*

In the *ScalaTest* view you can see the test result:

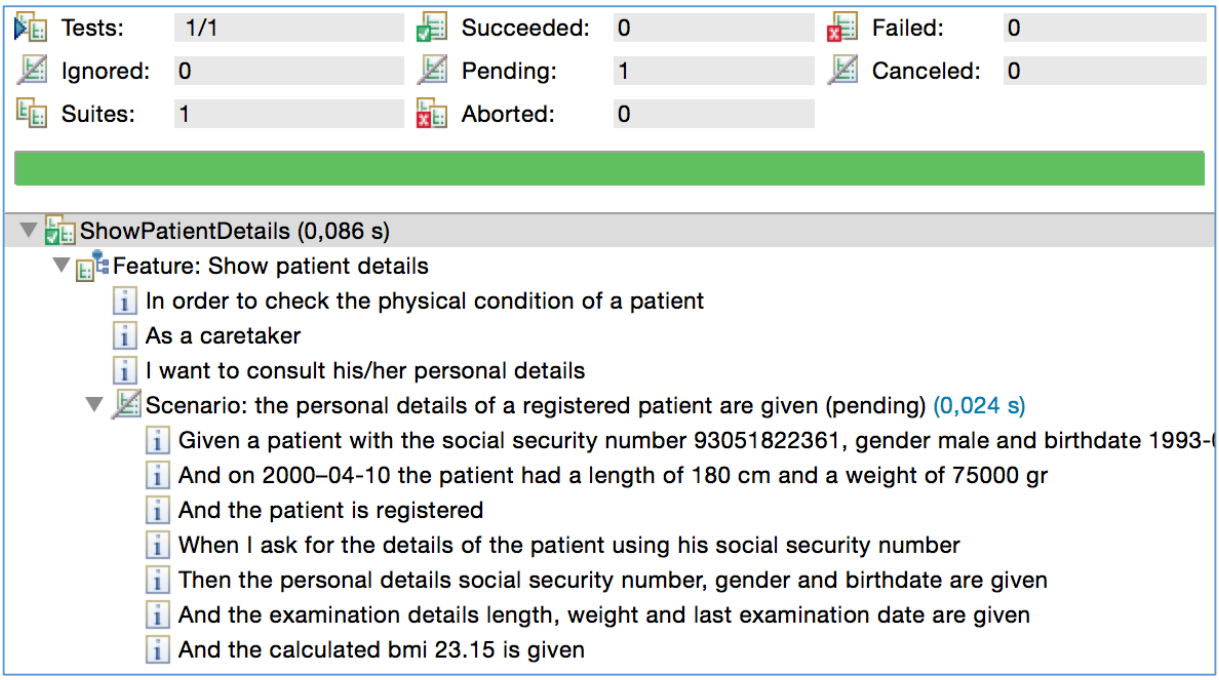


Figure 2. Example test output

In the **output console** you can see the textual representation:

```
<terminated> Feature- Show patient details Scenario- the personal details of a registered patient are given [ScalaTest] /Library/Java/JavaVirtu
Run starting. Expected test count is: 1
ShowPatientDetails:
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 (pending)
    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
Run completed in 247 milliseconds.
Total number of tests run: 0
Suites: completed 1, aborted 0
Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 1
No tests were executed.
```

Figure 3. Example console output

STEP 2: MAKE THE STEPS EXECUTABLE

After each step, write the Scala code to connect to your actual domain classes…
Remove the `pending` statement.

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

import org.scalatest.FeatureSpec
import org.scalatest.GivenWhenThen
import java.text.SimpleDateFormat
import org.ucll.demo.domain.Gender
import org.ucll.demo.service.api.java.to.PersonDetail

class ShowPatientDetails extends FeatureSpec with GivenWhenThen {
  val service = new PersonServiceJavaApi
  val dateFormatter= new SimpleDateFormat("yyyy-MM-dd")

  feature("Show patient details") {
    info("In order to check the physical condition of a patient")
    info("As a caretaker")
    info("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")
      val socialSecurityNumber = "93051822361"
      val gender = Gender.MALE
      val birthDate = dateFormatter.parse("1993-04-18")
      val patient = new PersonDetail(socialSecurityNumber, gender, birthDate)

      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 4. Example implementation of the first step

See *Attachment 2: Example scenario without using parameters* 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/htmldir*, you can find website-pages created by *ScalaTest*.
- Open the file `index.html` in your browser. Try to navigate to the html report of your story.

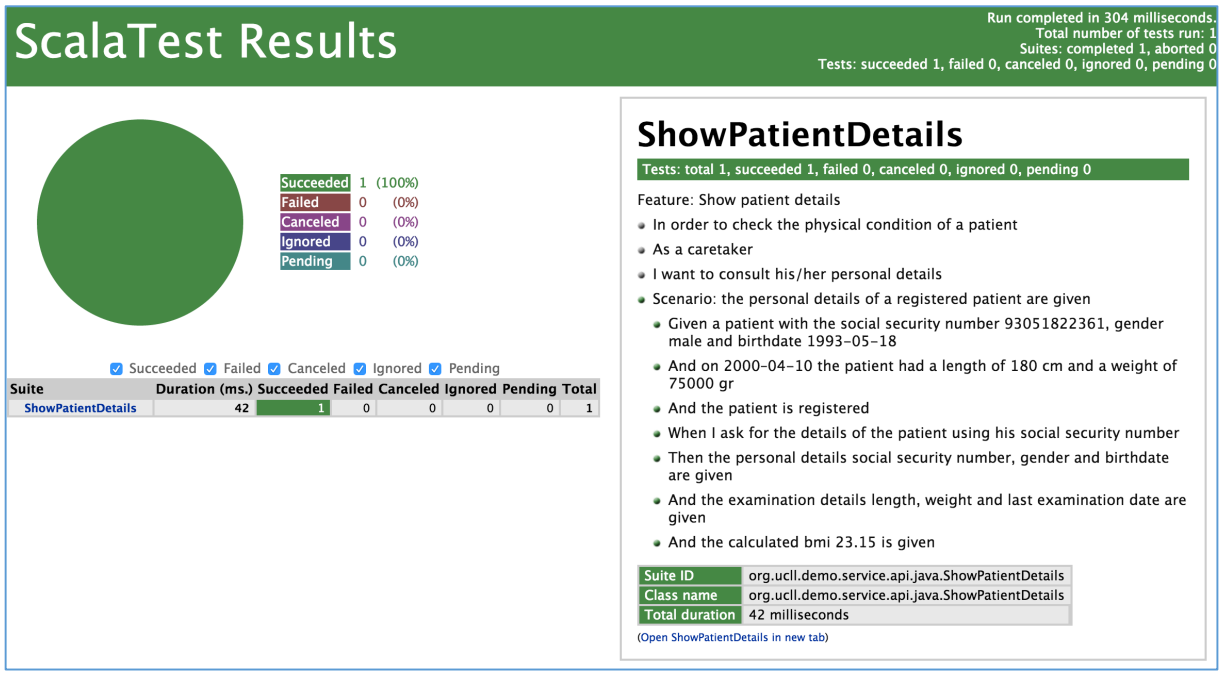


Figure 5. Example HTML report

EXTRA

REUSE FIXTURE

1. Add the scenario's given in *Attachment 5: Specifications_2.txt*.
2. You can see that you get code duplication: the *Given* steps of both scenario's, for instance, are identical. If you need the same mutable fixture objects in multiple tests, *ScalaTest* allows you to use *get-fixture* methods. A *get-fixture* method returns a new instance of a needed fixture object each time it is called.
    - Write a *get-fixture* method
    - Move the code to create a patient object to this method
    - In the *Given* step, call the *get-fixture* method and store the result in a local variable
    - Refactor the other steps: get the data you need from the fixture object.

```
class ShowPatientDetails extends FeatureSpec with GivenWhenThen {
  val service = new PersonServiceJavaApi
  val dateFormatter= new SimpleDateFormat("yyyy-MM-dd")

  def fixture() = new {
    val socialSecurityNumber = "93051822361"
    val gender = Gender.MALE
    val birthDate = dateFormatter.parse("1993-05-18")

    val patient = new PersonDetail(socialSecurityNumber, gender, birthDate)
  }

  feature("Show patient details") {
    info("In order to check the physical condition of a patient")
    info("As a caretaker")
    info("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")
      val testData = fixture();

      And("on 2000-04-10 the patient had a length of 180 cm and a weight of 75000 gr")
      val examinationDate = dateFormatter.parse("2000-04-10")
      val examination = new ExaminationDetail(180, 75000, examinationDate)
      testData.patient.setExaminationDetail(examination)

      And("the patient is registered")
      service.addPerson(testData.patient)
```

Figure 6. get-fixture method

3. Run the test to check everything still works.

The second step of both scenarios is almost the same: only the actual data used to create the examination object differ. As you can use parameters in a *get-fixture* method, this should not be a problem.

```
class ShowPatientDetails extends FeatureSpec with GivenWhenThen {
  val service = new PersonServiceJavaApi
  val dateFormatter= new SimpleDateFormat("yyyy-MM-dd")

  def fixture(length: Int, weight: Int, date: String) = new {
    val socialSecurityNumber = "93051822361"
    val gender = Gender.MALE
    val birthDate = dateFormatter.parse("1993-05-18")
    val patient = new PersonDetail(socialSecurityNumber, gender, birthDate)

    val examinationDate = dateFormatter.parse(date)
    val examination = new ExaminationDetail(length, weight, examinationDate)
    patient.setExaminationDetail(examination)

    service.addPerson(patient)
  }

  feature("Show patient details") {
    info("In order to check the physical condition of a patient")
    info("As a caretaker")
    info("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")
      val testData = fixture(180, 75000, "2000-04-10")

      When("I ask for the details of the patient using his social security number")
      val detailsRetrieved = service.getPerson(testData.socialSecurityNumber)

      Then("the personal details social security number, gender and birthdate are given")
```

Figure 7.get-fixture method with parameters

4. Implement the other scenario, using the fixture.

   Of course, you can avoid duplicate methods by writing ‘plain’ methods...

```
def differNoMoreThanFewSeconds(date: Date, otherDate: Date): Boolean = {
  return date.compareTo(date) <= 0 && date.compareTo(otherDate) >= -2;
}
```

Figure 8. Example method written in Scala

SCENARIO'S WITH EXPECTED EXCEPTIONS

1. Add the scenario given in *Attachment 6: Specifications_3.txt*.
2. To check if an exception is thrown, do not ask for the patient's details in the *When* step. Instead, leave the When step empty, and try to ask the details in the *Then* step. To check if an exception is thrown, surround the code with: `intercept[IllegalArgumentException] { }`

```
scenario("an error message is given if the patient cannot be found"){
  Given(" a patient that is not registered")
  val testData = fixture(180, 75000, "2000-04-10")

  When("I ask for the details of the patient using his social security number")

  Then("an error message is given")
  var detailsRetrieved: PersonDetail = null
  intercept[IllegalArgumentException] {
    detailsRetrieved = service.getPerson(testData.socialSecurityNumber)
  }
  And("no details are given")
  assert(detailsRetrieved == null);
})
```

Figure 9. Intercept expected exception

DATA TABLES

1. Add the scenario given in *Attachment 7: Specifications_4.txt*. This scenario is different from the previous ones:
   - It has no concrete values in the *Given When Then* steps
   - After the scenario, you can see a table of examples. We want to run the scenario for all the examples.
2. Add the import statement
   `import org.scalatest.prop.TableDrivenPropertyChecks._`

```
scenario("the bmi is rounded to 2 digits"){
val examples = Table(
  ("length", "weight", "bmi"),
  (160, 65000, 25.39),
  (160, 65001, 25.39),
  (160, 65009, 25.39),
  (180, 75000, 23.15),
  (180, 75009, 23.15))

    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 is given rounded to 2 digits")
})
```

Figure 10. Table with examples

3. Iterate over the table to test all the examples:

```
scenario("the bmi is rounded to 2 digits"){
val examples = Table(
  ("length", "weight", "bmi"),
  (160, 65000, 25.39),
  (160, 65001, 25.39),
  (160, 65009, 25.39),
  (180, 75000, 23.15),
  (180, 75009, 23.15))

  forAll(examples) { (length: Int, weight: Int, bmi: Double) =>
    whenever(length != 0) {
      Given("a patient that is registered with a length " + length + " cm and weight " + weight + " gr")
      val testData = fixture(length, weight, "2000-04-10")
      service.addPerson(testData.patient)

      When("I ask for the details of the patient")
      val detailsRetrieved = service.getPerson(testData.socialSecurityNumber)

      Then("the bmi is given rounded to 2 digits")
      assert(bmi == detailsRetrieved.getBmi)
      }
    }
})
```

Figure 11. Test data table

See *Attachment 4: Example step class after implementing the 4 scenarios* for an example implementation of all scenarios

AND THEN …

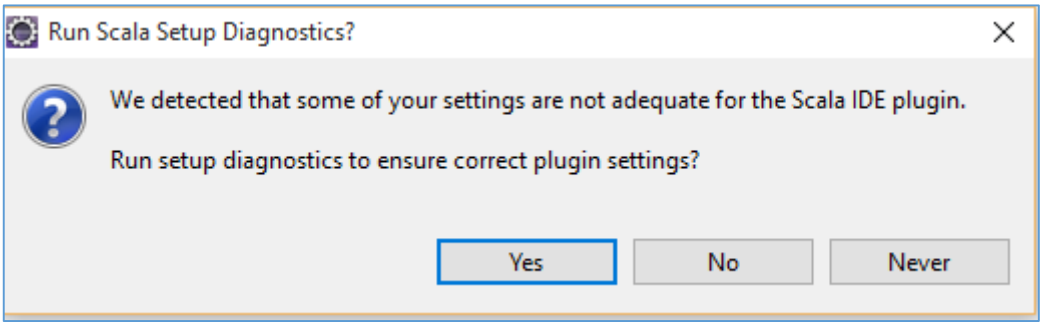
Write specifications for the user story *Add physical examination data*.

1. How easy is it to focus on the content, without thinking about technical aspects?
2. Investigate other possibilities of *ScalaTest*. 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 *ScalaTest* used in combination with *Mockito* and *Selenium*.

TROUBLESHOOTING

- If you get the message about your settings, click *Yes* and choose the recommended settings.



- If the scalatest-maven-plugin complains it cannot find …`/htmldir`, check if you gave any spaces in the path of your project. If so, move your project to a location without spaces in the path.

- If Eclipse is very slow: replace following settings in the *eclipse.ini* file:

```
-Xms40m
-Xmx512m
```

by

```
-Xms512m
-Xmx1024m
```

On a **Mac OS X** system, you can find `eclipse.ini` by right-clicking (or *Ctrl+click*) on the Eclipse executable in Finder, choose *Show Package Contents*, and then locate *eclipse.ini* in the *Eclipse* folder under *Contents*.

- If you cannot choose *Run As* | *Run as ScalaTest - Test*, check if your package declaration is OK

ATTACHMENTS

ATTACHMENT 1: ADDITIONS TO POM.XML

```
<project ...
    <properties>
        ...
        <scalatest.version>2.2.5</scalatest.version>
        <pegdown.version>1.2.1</pegdown.version>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>net.alchim31.maven</groupId>
                <artifactId>scala-maven-plugin</artifactId>
                <executions>
                    <execution>
                        <id>scala-compile-first</id>
                        <phase>process-resources</phase>
                        <goals>
                            <goal>add-source</goal>
                            <goal>compile</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>scala-test-compile</id>
                        <phase>process-test-resources</phase>
                        <goals>
                            <goal>testCompile</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>

            <plugin>
                <groupId>org.scalatest</groupId>
                <artifactId>scalatest-maven-plugin</artifactId>
                <version>1.0</version>
                <configuration>
```

```
                    <reportsDirectory>
                        ${project.build.directory}/surefire-reports
                    </reportsDirectory>
                    <junitxml>.</junitxml>
                    <htmlreporters>
                        ${project.build.directory}/htmldir
                    </htmlreporters>
                    <filereports>WDF TestSuite.txt</filereports>
                </configuration>
                <executions>
                    <execution>
                        <id>test</id>
                        <goals>
                            <goal>test</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            ...
        </plugins>
    </build>

    <dependencies>
        <dependency>
            <groupId>org.scalatest</groupId>
            <artifactId>scalatest_2.11</artifactId>
            <version>${scalatest.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.pegdown</groupId>
            <artifactId>pegdown</artifactId>
            <version>${pegdown.version}</version>
            <scope>test</scope>
        </dependency>
        ...
    </dependencies>
</project>
```

ATTACHMENT 2: EXAMPLE SCENARIO WITHOUT USING PARAMETERS

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

import org.scalatest.FeatureSpec
import org.scalatest.GivenWhenThen
import java.text.SimpleDateFormat
import org.ucll.demo.domain.Gender
import org.ucll.demo.service.api.java.to.PersonDetail
import org.ucll.demo.service.api.java.to.ExaminationDetail

class ShowPatientDetails extends FeatureSpec with GivenWhenThen {
  val service = new PersonServiceJavaApi
  val dateFormatter= new SimpleDateFormat("yyyy-MM-dd")

  feature("Show patient details") {
    info("In order to check the physical condition of a patient")
    info("As a caretaker")
    info("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")
      val socialSecurityNumber = "93051822361"
      val gender = Gender.MALE
      val birthDate = dateFormatter.parse("1993-05-18")
      val patient = new PersonDetail(socialSecurityNumber, gender, birthDate)

      And("on 2000-04-10 the patient had a length of 180 cm and a weight of 75000 gr")
      val examinationDate = dateFormatter.parse("2000-04-10")
      val examination = new ExaminationDetail(180, 75000, examinationDate)
      patient.setExaminationDetail(examination)

      And("the patient is registered")
      service.addPerson(patient)

      When("I ask for the details of the patient using his social security number")
      val detailsRetrieved = service.getPerson(socialSecurityNumber)

      Then("the personal details social security number, gender and birthdate are given")
      assert(socialSecurityNumber == detailsRetrieved.getSocialSecurityNumber)
      assert(gender == detailsRetrieved.getGender);
```

```
      assert(birthDate.compareTo(detailsRetrieved.getBirthdate) == 0)

      And("the examination details length, weight and last examination date are given")
      val examinationData = detailsRetrieved.getExaminationDetail
      assert(180 == examinationData.getLength)
      assert(75000 == examinationData.getWeight);
      assert(examinationDate.compareTo(examinationData.getExaminationDate) == 0)

      And("the calculated bmi 23.15 is given")
      assert(23.15 == detailsRetrieved.getBmi);
    })
  }
}
```

ATTACHMENT 3: SPECIFICATIONS_1.TXT

```
feature("Show patient details") {
  info("In order to check the physical condition of a patient")
  info("As a caretaker")
  info("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")
    pending
  })
}
```

ATTACHMENT 4: EXAMPLE STEP CLASS AFTER IMPLEMENTING THE 4 SCENARIOS

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

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

import org.scalatest.FeatureSpec
import org.scalatest.GivenWhenThen
import org.scalatest.TestData
import org.scalatest.prop.TableDrivenPropertyChecks._
import org.ucll.demo.domain.Gender
import org.ucll.demo.service.api.java.to.ExaminationDetail
import org.ucll.demo.service.api.java.to.PersonDetail

class ShowPatientDetails extends FeatureSpec with GivenWhenThen {
  val service = new PersonServiceJavaApi
  val dateFormatter= new SimpleDateFormat("yyyy-MM-dd")

  def fixture(length: Int, weight: Int, date: String) =
    new {
      val socialSecurityNumber = "93051822361"
      val gender = Gender.MALE
      val birthDate = dateFormatter.parse("1993-04-18")
      val examinationDate = dateFormatter.parse(date)
      val examination = new ExaminationDetail(length, weight, examinationDate)

      val patient = new PersonDetail(socialSecurityNumber, gender, birthDate)
      patient.setExaminationDetail(examination)

      service.deletePerson(socialSecurityNumber)
    }

  feature("Show patient details") {
    info("In order to check the physical condition of a patient")
    info("As a caretaker")
    info("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")
  val testData = fixture(180, 75000, "2000-04-10")

  And("the patient is registered")
  service.addPerson(testData.patient)

  When("I ask for the details of the patient using his social security number")
  val detailsRetrieved = service.getPerson(testData.socialSecurityNumber)

  Then("the personal details social security number, gender and birthdate are given")
  assert(testData.socialSecurityNumber == detailsRetrieved.getSocialSecurityNumber)
  assert(testData.gender == detailsRetrieved.getGender)
  assert(differNoMoreThanFewSeconds(testData.birthDate, detailsRetrieved.getBirthdate))

  And("the examination details length, weight and last examination date are given")
  assert(testData.examination.getLength == detailsRetrieved.getExaminationDetail.getLength)
  assert(testData.examination.getWeight == detailsRetrieved.getExaminationDetail.getWeight)
  assert(differNoMoreThanFewSeconds(testData.examinationDate, detailsRetrieved.getExaminationDetail.getExaminationDate))

  And("the calculated bmi 23.15 is given")
  assert(23.15 == detailsRetrieved.getBmi);
})

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")
  val testData = fixture(180, 80000, "2000-04-17")

  And("the patient is registered")
  service.addPerson(testData.patient)

  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")
  val olderExamination = new ExaminationDetail(180, 75000, dateFormatter.parse("2000-04-10"))
  service.addExamination(olderExamination, testData.socialSecurityNumber)

  When("I ask for the details of the patient using his social security number")
  val detailsRetrieved = service.getPerson(testData.socialSecurityNumber)

  Then("the length 180, weight 80000, and date of the examination date 2000-04-17 are given")
  assert(testData.examination.getLength == detailsRetrieved.getExaminationDetail.getLength)
  assert(testData.examination.getWeight == detailsRetrieved.getExaminationDetail.getWeight)
```

```
  assert(differNoMoreThanFewSeconds(testData.examinationDate, detailsRetrieved.getExaminationDetail.getExaminationDate))

  And("And the calculated bmi 24.69 is based on these data")
  assert(24.69 == detailsRetrieved.getBmi);
})

scenario("an error message is given if the patient cannot be found")({
  Given(" a patient that is not registered")
  val testData = fixture(180, 75000, "2000-04-10")

  When("I ask for the details of the patient using his social security number")

  Then("an error message is given")
  var detailsRetrieved: PersonDetail = null
  intercept[IllegalArgumentException] {
    detailsRetrieved = service.getPerson(testData.socialSecurityNumber)
  }

  And("no details are given")
  assert(detailsRetrieved == null);
})

scenario("the bmi is rounded to 2 digits")({
  val examples = Table(
    ("length", "weight", "bmi"),
    (160, 65000, 25.39),
    (160, 65001, 25.39),
    (160, 65009, 25.39),
    (180, 75000, 23.15),
    (180, 75009, 23.15))

  forAll(examples) { (length: Int, weight: Int, bmi: Double) =>
    whenever(length != 0) {

      Given("a patient that is registered with a length " + length + " cm and weight " + weight + " gr")
      val testData = fixture(length, weight, "2000-04-10")
      service.addPerson(testData.patient)

      When("I ask for the details of the patient")
      val detailsRetrieved = service.getPerson(testData.socialSecurityNumber)

      Then("the bmi " + bmi + " is given rounded to 2 digits")
```

```
        assert(bmi == detailsRetrieved.getBmi);

      }
    }
  })

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

ATTACHMENT 5: 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 10-04-2000 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 date 2000-04-17 are given")
  And("the calculated bmi 24.69 is based on these data")
})
```

ATTACHMENT 6: 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 7: SPECIFICATIONS_4.TXT

```
  scenario("the bmi is rounded to 2 digits")({
    val examples = Table(
      ("length", "weight", "bmi"),
      (160, 65000, 25.39),
      (160, 65001, 25.39),
      (160, 65009, 25.39),
      (180, 75000, 23.15),
      (180, 75009, 23.15))

        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 is given rounded to 2 digits")
})
```

ATTACHMENT 8: EXAMPLE USING MOCKITO

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

import org.scalatest.FlatSpec
import org.mockito.MockitoAnnotations.Mock
import org.ucll.demo.service.PersonService
import org.ucll.demo.domain.Person
import org.ucll.demo.service.api.java.assembler.PersonToAssembler
import org.ucll.demo.service.api.java.to.PersonDetail
import org.ucll.demo.repository.PersonRepository
import org.scalatest.mock.MockitoSugar
import org.ucll.demo.service.api.java.assembler.ExaminationToAssembler
import org.mockito.BDDMockito.given
import org.mockito.BDDMockito.then
import org.mockito.Matchers.any
import org.mockito.Mockito.never

class PersonServiceJavaApiSpec extends FlatSpec with MockitoSugar {
  val SocialSecurityNumber = "93051822361"

  def fixture =
    new {
      val mockPersonService = mock[PersonService]
      val mockExaminationToAssembler = mock[ExaminationToAssembler]
      val mockPersonToAssembler = mock[PersonToAssembler]
      val mockPerson = mock[Person]
      val mockPersonDetail = mock[PersonDetail]

      var service = new PersonServiceJavaApi(mockPersonService, mockExaminationToAssembler, mockPersonToAssembler)
    }

  "getPerson" should "retrieve a person from the repository if the personId is provided" in {
    val testData = fixture

    given(testData.mockPersonService.getPerson(SocialSecurityNumber)).willReturn(testData.mockPerson)
    given(testData.mockPersonToAssembler.createPersonDetailTo(testData.mockPerson)).willReturn(testData.mockPersonDetail)

    val personRetrieved = testData.service.getPerson(SocialSecurityNumber)
```

```
    then(testData.mockPersonService).should.getPerson(SocialSecurityNumber)
    then(testData.mockPersonToAssembler).should.createPersonDetailTo(testData.mockPerson)
    assert(personRetrieved != null)
  }

  it should "throw an IllegalArgumentException if an unknown personId is provided" in {
    val testData = fixture

    given(testData.mockPersonService.getPerson(SocialSecurityNumber)).willThrow(new IllegalArgumentException)
    intercept[IllegalArgumentException] {
      testData.service.getPerson(SocialSecurityNumber)
    }
    then(testData.mockPersonService).should.getPerson(SocialSecurityNumber)
    then(testData.mockPersonToAssembler).should(never).createPersonDetailTo(testData.mockPerson)
  }

  it should "throw an IllegalArgumentException if no personId is provided" in {
    val testData = fixture

    given(testData.mockPersonService.getPerson(null)).willThrow(new IllegalArgumentException)
    intercept[IllegalArgumentException] {
      testData.service.getPerson(null)
    }
    then(testData.mockPersonService).should.getPerson(null)
    then(testData.mockPersonToAssembler).should(never).createPersonDetailTo(testData.mockPerson)
  }

}
```

ATTACHMENT 9: EXAMPLE USING SELENIUM

```
package org.ucll.demo.ui

import java.util.Calendar
import java.util.Date
import org.openqa.selenium.WebDriver
import org.openqa.selenium.firefox.FirefoxDriver
import org.scalatest.FeatureSpec
import org.scalatest.GivenWhenThen
import org.scalatest.prop.TableDrivenPropertyChecks._
import org.scalatest.selenium.WebBrowser
import org.ucll.demo.domain.Gender
import org.ucll.demo.ui.pages.ExaminationDetailPageScala
import org.ucll.demo.ui.pages.PersonDetailPageScala
import org.ucll.demo.ui.pages.PersonOverviewPageScala
import java.text.SimpleDateFormat

class ShowPatientDetailsIT extends FeatureSpec with GivenWhenThen with WebBrowser {
  var personDetailPage: PersonDetailPageScala = _
  var examinationDetailPage: ExaminationDetailPageScala = _
  var personOverviewPage: PersonOverviewPageScala = _
  var driver: WebDriver = _
  val HostAndPortUrl = "http://localhost:8080"
  val ContextUrl = HostAndPortUrl + "/bmi"
  val dateFormatter= new SimpleDateFormat("yyyy-MM-dd")

  def fixture(length: Int, weight: Int, date: String) =
    new {
      val registeredSocialSecurityNumber = "93051822361"
      val registeredGender = Gender.MALE
      val registeredBirthdate = dateFormatter.parse("1993-04-18")
      val registeredLength = length
      val registeredWeight = weight
      val registeredExaminationDate = dateFormatter.parse(date)

      registerPerson(registeredSocialSecurityNumber, registeredGender, registeredBirthdate,
          registeredLength, registeredWeight, registeredExaminationDate)
    }
```

```
feature("Show patient details") {
  info("In order to check the physical condition of a patient")
  info("As a caretaker")
  info("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")
  val testData = fixture(180, 75000, "2000-04-10")

  When("I ask for the details of the patient using his social security number")
  personDetailPage = personOverviewPage.showPerson(testData.registeredSocialSecurityNumber)

  Then("the personal details social security number, gender and birthdate are given")
  assert(testData.registeredSocialSecurityNumber == personDetailPage.getSocialSecurityNumber)
  assert(testData.registeredGender == personDetailPage.getGender)
  assert(differNoMoreThanFewSeconds(testData.registeredBirthdate, personDetailPage.getBirthdate))

  And("the examination details length, weight and last examination date are given")
  assert(testData.registeredLength == personDetailPage.getLength)
  assert(testData.registeredWeight == personDetailPage.getWeight)
  assert(differNoMoreThanFewSeconds(testData.registeredExaminationDate, personDetailPage.getExaminationDate))

  And("the calculated bmi 23.15 is given")
  assert(23.15 == personDetailPage.getBmi)

  clearTestData
})

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")
  val testData = fixture(180, 80000, "2000-04-17")

  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")
  addOlderExamination(testData.registeredSocialSecurityNumber, 180, 75000, "2000-04-10")

  When("I ask for the details of the patient using his social security number")
  personDetailPage = personOverviewPage.showPerson(testData.registeredSocialSecurityNumber)
```

```
  Then("the length 180, weight 80000, and date of the examination date 2000-04-17 are given")
  assert(testData.registeredLength == personDetailPage.getLength)
  assert(testData.registeredWeight == personDetailPage.getWeight)
  assert(differNoMoreThanFewSeconds(testData.registeredExaminationDate, personDetailPage.getExaminationDate))

  And("And the calculated bmi 24.69 is based on these data")
  assert(24.69 == personDetailPage.getBmi)

  clearTestData
})

scenario("the bmi is rounded to 2 digits")({
  val examples = Table(
    ("length", "weight", "bmi"),
    (160, 65000, 25.39),
    (160, 65001, 25.39),
    (160, 65009, 25.39),
    (180, 75000, 23.15),
    (180, 75009, 23.15))

  forAll(examples) { (length: Int, weight: Int, bmi: Double) =>
    whenever(length != 0) {

      Given("a patient that is registered with a length " + length + " cm and weight " + weight + " gr")
      val testData = fixture(length, weight, "2000-04-17")

      When("I ask for the details of the patient")
      personDetailPage = personOverviewPage.showPerson(testData.registeredSocialSecurityNumber)

      Then("the bmi " + bmi + " is given rounded to 2 digits")
      assert(bmi == personDetailPage.getBmi)

      clearTestData
    }
  }
})

def differNoMoreThanFewSeconds(date: Date, otherDate: Date): Boolean = {
  return date.compareTo(date) <= 0 && date.compareTo(otherDate) >= -5
}
```

```
  def registerPerson(number: String, gender: Gender, birthdate: Date, length: Int, weight: Int, date: Date) = {
    //TODO replace with DBUnit, ...
    driver = new FirefoxDriver
    driver.get(ContextUrl)
    personOverviewPage = new PersonOverviewPageScala(driver)
    personDetailPage = personOverviewPage.addPerson
    personOverviewPage = personDetailPage.addPerson(number, birthdate, gender, length, weight, date)
  }

  def addOlderExamination(number: String, length: Int, weight: Int, date: String) = {
    //TODO replace with DBUnit, ...
    personDetailPage = personOverviewPage.showPerson(number)
    examinationDetailPage = personDetailPage.addExamination
    personDetailPage = examinationDetailPage.addExamination(length, weight, dateFormatter.parse(date))
    personOverviewPage = personDetailPage.cancel
  }

  def clearTestData = {
    //TODO replace with DBUnit, ...
    driver.get(ContextUrl)
    personOverviewPage = new PersonOverviewPageScala(driver)
    personOverviewPage.deletePerson
    driver.quit
  }
}
```

ATTACHMENT 10: SCALATEST PAGE OBJECTS

PERSONOVERVIEWPAGE

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

import org.scalatest.selenium.Page
import org.openqa.selenium.WebDriver
import org.scalatest.selenium.WebBrowser

class PersonOverviewPageScala(webDriver: WebDriver) extends Page with WebBrowser {
  implicit val driver = webDriver
  val url = driver.getCurrentUrl

  def showPerson(id: String): PersonDetailPageScala = {
    click on partialLinkText(id)

    return new PersonDetailPageScala(driver)
  }

  def addPerson(): PersonDetailPageScala = {
    click on id("personOverviewForm:addPersonButton")
    return new PersonDetailPageScala(driver)
  }

  def deletePerson(): PersonDetailPageScala = {
    click on id("personOverviewForm:personTable:0:delete")
    return new PersonDetailPageScala(driver)
  }
}
```

PERSONDETAILPAGE

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

import org.openqa.selenium.WebDriver
import org.scalatest.selenium.Page
import org.scalatest.selenium.WebBrowser
import java.util.Date
import org.ucll.demo.domain.Gender
import java.text.SimpleDateFormat

class PersonDetailPageScala(webDriver: WebDriver) extends Page with WebBrowser {
  implicit val driver = webDriver
  val url = driver.getCurrentUrl
  val examinationfieldsPage = new ExaminationFieldsPageScala(driver, "personForm")

  def addPerson(socialSecurityNumber: String, birthdate: Date, gender: Gender, length: Int, weight: Int, examinationDate: Date):
PersonOverviewPageScala = {
    textField("personForm:number").value = socialSecurityNumber
    textField("personForm:birthDate").value = new SimpleDateFormat("yyyy-MM-dd").format(birthdate)
    textField("personForm:birthDate").value = new SimpleDateFormat("yyyy-MM-dd").format(birthdate)
    singleSel("personForm:gender").value = "MALE"
    examinationfieldsPage.addExaminationData(length, weight, examinationDate);

    click on id("personForm:actionSave")

    return new PersonOverviewPageScala(driver);
  }

  def getSocialSecurityNumber: String = {
    return textField("personForm:number").value;
  }

  def getGender: Gender = {
    val genderAsString = singleSel("personForm:gender").selection.iterator.next()
    return Gender.valueOf(genderAsString.toUpperCase());
  }

  def getBirthdate(): Date = {
    val birhtdate = textField("personForm:birthDate").value
    val format = new SimpleDateFormat("yyyy-MM-dd")
```

```
    return format.parse(birhtdate)
  }

  def getLength(): Int = {
    return examinationfieldsPage.getLength();
  }

  def getWeight(): Int = {
    return examinationfieldsPage.getWeight();
  }

  def getExaminationDate(): Date = {
    return examinationfieldsPage.getExaminationDate();
  }

  def getBmi(): Double = {
    return textField("personForm:bmi").value.toDouble
  }

  def addExamination: ExaminationDetailPageScala = {
    click on id("personToExaminationForm:addExamination")
    return new ExaminationDetailPageScala(driver);
  }

  def cancel: PersonOverviewPageScala = {
    click on id("personToExaminationForm:cancel")
    return new PersonOverviewPageScala(driver);
  }
}
```

EXAMINATIONDETAILPAGE

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

import org.openqa.selenium.WebDriver
import org.scalatest.selenium.Page
import org.scalatest.selenium.WebBrowser
import java.util.Date

class ExaminationDetailPageScala(webDriver: WebDriver) extends Page with WebBrowser {
  implicit val driver = webDriver
  val url = driver.getCurrentUrl
  val examinationfieldsPage = new ExaminationFieldsPageScala(driver, "examinationForm")

  def addExamination(length: Int, weight: Int, examinationDate: Date): PersonDetailPageScala = {
    examinationfieldsPage.addExaminationData(length, weight, examinationDate);
    click on id("examinationForm:saveExamination")
    return new PersonDetailPageScala(driver);
  }

}
```

EXAMINATIONFIELDSPAGE

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

import org.openqa.selenium.WebDriver
import org.scalatest.selenium.Page
import org.scalatest.selenium.WebBrowser
import java.text.SimpleDateFormat
import java.util.Date

class ExaminationFieldsPageScala(webDriver: WebDriver, prefix: String) extends Page with WebBrowser {
  implicit val driver = webDriver
  val url = driver.getCurrentUrl
  val formPrefix = prefix;

  def addExaminationData(length: Int, weight: Int, examinationDate: Date) {
    textField(formPrefix + ":length").value = Integer.toString(length)
    textField(formPrefix + ":weight").value = Integer.toString(weight)
    textField(formPrefix + ":examinationDate").value = new SimpleDateFormat("yyyy-MM-dd").format(examinationDate)
  }

  def getLength(): Int = {
    val length = textField(formPrefix + ":length").value
    return Integer.parseInt(length);
  }

    def getWeight() : Int = {
    val weight = textField(formPrefix + ":weight").value
    return Integer.parseInt(weight);
    }

    def getExaminationDate() : Date = {
    val examinationDate = textField(formPrefix + ":examinationDate").value
    val format = new SimpleDateFormat("yyyy-MM-dd")
         return format.parse(examinationDate)
    }


}
```