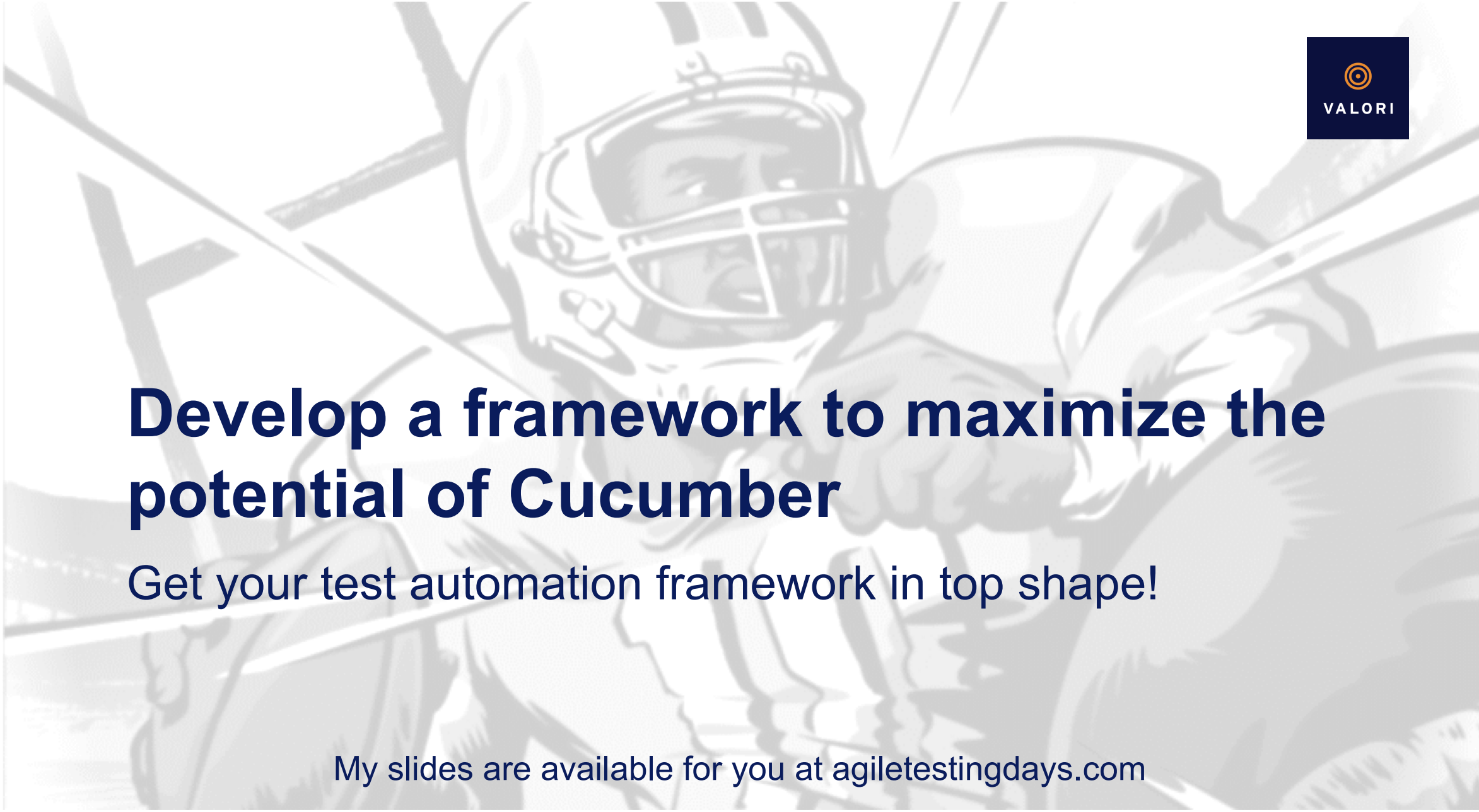


VALORI

Develop a framework to maximize the potential of Cucumber

Get your test automation framework in top shape!

My slides are available for you at agiletestingdays.com

@MichelAmin47

Introduction

- Who am I?

- Michel Lalmohamed
  - Senior Test Automation Specialist
  - Valori Full Stack Testing
  - 12 years automation experience

Introduction

- Workshop prerequisites:
  - Laptop
  - Java8 installed
  - Java IDE (preferably IntelliJ) + Cucumber plugin/support
  - Google Chrome
  - Java programming knowledge
  - OOP knowledge
  - Cucumber knowledge

Introduction

- Too ensure wide range of support for Cucumber IDE plugins

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

@Given("User is on the homepage")
@And("The user is on the homepage")
public void userIsOnTheHomepage() {
    // Open the website
    driver.get("https://webshop.mobiletestautomation.nl/");
    driver.manage().window().maximize();
    Assert.assertTrue( message: "Check if empty element is visible",
            driver.findElement(By.className("logo")).isDisplayed());
}
```

- We will use the depricated Cucumber classes

Good or Bad?

```
Funktionalität: Als Benutzer möchte ich mich bei MyAccount anmelden

Szenario: Der Benutzer meldet sich in seinem MyAccount an
  Gegeben sei   der Benutzer befindet sich auf der Homepage
  Wenn          sich der Benutzer bei seinem MyAccount anmeldet
  Dann          sollte der Benutzer zu seiner Kontoseite weitergeleitet werden
```

Good or Bad?

```
Funktionalität: Als Benutzer möchte ich mich bei MyAccount anmelden

Szenario: Der Benutzer meldet sich in seinem MyAccount an
  Gegeben sei   der Benutzer befindet sich auf der Homepage
  Wenn          sich der Benutzer bei seinem MyAccount anmeldet
  Dann          sollte der Benutzer zu seiner Kontoseite weitergeleitet werden
```

- ✔ Funktionalität: Als Benutzer möchte ich mich bei MyAccount anmelden
  - ✔ Scenario: Der Benutzer meldet sich in seinem MyAccount an
    - ✔ der Benutzer befindet sich auf der Homepage
    - ✔ sich der Benutzer bei seinem MyAccount anmeldet
    - ✔ sollte der Benutzer zu seiner Kontoseite weitergeleitet werden

Good or Bad?

```
Scenario: Buy a banana with a credit card
  Given The user is on the homepage
  When I select the fruit section
  Then I should see all the fruit
  When I select a banana in quickview
  And The user selects a banana
  Then The site should see take me to the cart screen
  Given I don't add any more bananas
  When The user buys the banana
  Then I should see a banana purchase receipt
```

Good or Bad?

```
Scenario: Buy a banana with a credit card
  Given The user is on the homepage
  When I select the fruit section
  Then I should see all the fruit
  When I select a banana in quickview
  And The user selects a banana
  Then The site should see take me to the cart screen
  Given I don't add any more bananas
  When The user buys the banana
  Then I should see a banana purchase receipt
```

- ✔ Feature: As a user I want to buy a banana
- ▼ ✔ Scenario: Buy a banana with a credit card
  - ✔ Hook: before
  - ✔ The user is on the homepage
  - ✔ I select the fruit section
  - ✔ I should see all the fruit
  - ✔ I select a banana in quickview
  - ✔ The user selects a banana
  - ✔ The site should see take me to the cart screen
  - ✔ I don't add any more bananas
  - ✔ The user buys the banana
  - ✔ I should see a banana purchase receipt
  - ✔ Hook: after

Good or Bad?

```
Scenario: Buy a banana and login
  Given The user is on the homepage
  When    The user buys a banana
  Then    The user should be shown a receipt
  Given   The user is on the homepage
  When    The user logs into his MyAccount
  Then    The user should be taken to his Account page
```

Good or Bad?

```
Scenario: Buy a banana and login
  Given The user is on the homepage
  When    The user buys a banana
  Then    The user should be shown a receipt
  Given   The user is on the homepage
  When    The user logs into his MyAccount
  Then    The user should be taken to his Account page
```

- ✔ Feature: As a user I want to buy a banana and login
- ▼ ✔ Scenario: Buy a banana and login
  - ✔ Hook: before
  - ✔ The user is on the homepage
  - ✔ The user buys a banana
  - ✔ The user should be shown a receipt
  - ✔ The user is on the homepage
  - ✔ The user logs into his MyAccount
  - ✔ The user should be taken to his Account page
  - ✔ Hook: after

Good or Bad?

- All tests ran and reported green
- Green is good, right?

Implementation is key

- What is the correct way to implement Cucumber into a testing framework?

Implementation is key

- What is the **TECHNICAL** correct way to implement Cucumber into a testing framework?

Implementation is key

- What is the TECHNICAL correct way to implement Cucumber into a testing framework?
  - No compile errors
  - Tests can be executed
  - No deprecation warnings
  - No pending exceptions
  - Test results should either be positive or negative

Implementation is key

- Today is NOT about technical implementation ☺
- Today I will share you my own truth regarding implementation
- My experiences throughout the years bundled in this workshop
  - Use Cucumber to its full advantage (in my humble opinion ☺)

Disclaimer

- Michel Lalmohamed does not work for Cucumber.io
- Michel Lalmohamed is not a shareholder of Cucumber.io
- Michel Lalmohamed is not a stakeholder in Cucumber.io

Disclaimer

- Michel Lalmohamed does not work for Cucumber.io
- Michel Lalmohamed is not a shareholder of Cucumber.io
- Michel Lalmohamed is not a stakeholder in Cucumber.io
- Michel Lalmohamed **is a big fan** of Cucumber / SpecFlow!

What is Cucumber?

- Cucumber == BDD?

What is Cucumber?

- Cucumber == BDD?
  - No

***BDD** gives a **clear understanding** as to **what the system should do** from the **perspective** of the **developer** and the **customer**. Meaning BDD is **not only customer focused** but also that BDD **focuses** on the **behavioural aspect of the system**. This **allows** much **easier collaboration** with **non-technical stakeholders** because BDD **tests** are **written in an English-like language**. The test describes the implementation rather than exposing the code level tests.*

What is Cucumber?

- Cucumber == BDD?
  - No
- We use BDD so we have to use Cucumber!

What is Cucumber?

- Cucumber == BDD?
  - No
- We use BDD so we have to use Cucumber!
  - No

What is Cucumber?

- Cucumber == BDD?
  - No
- We use BDD so we have to use Cucumber!
  - No
- Cucumber is a tool

What is Cucumber?

- Cucumber == BDD?
  - No
- We use BDD so we have to use Cucumber!
  - No
- Cucumber is a tool
  - Yes, it’s an open source collaboration tool

What is Cucumber?

- Cucumber == BDD?
  - No
- We use BDD so we have to use Cucumber!
  - No
- Cucumber is a tool
  - Yes, it’s an open source collaboration tool
- Cucumber makes my test framework more accessible

What is Cucumber?

- Cucumber == BDD?
  - No
- We use BDD so we have to use Cucumber!
  - No
- Cucumber is a tool
  - Yes, it’s an open source collaboration tool
- Cucumber makes my test framework more accessible
  - Yes, when implemented correctly

What is Cucumber?

- Cucumber == BDD?
  - No
- We use BDD so we have to use Cucumber!
  - No
- Cucumber is a tool
  - Yes, it's an open source collaboration tool
- Cucumber makes my test framework more accessible
  - Yes, when implemented correctly
- Cucumber means more code for me to write

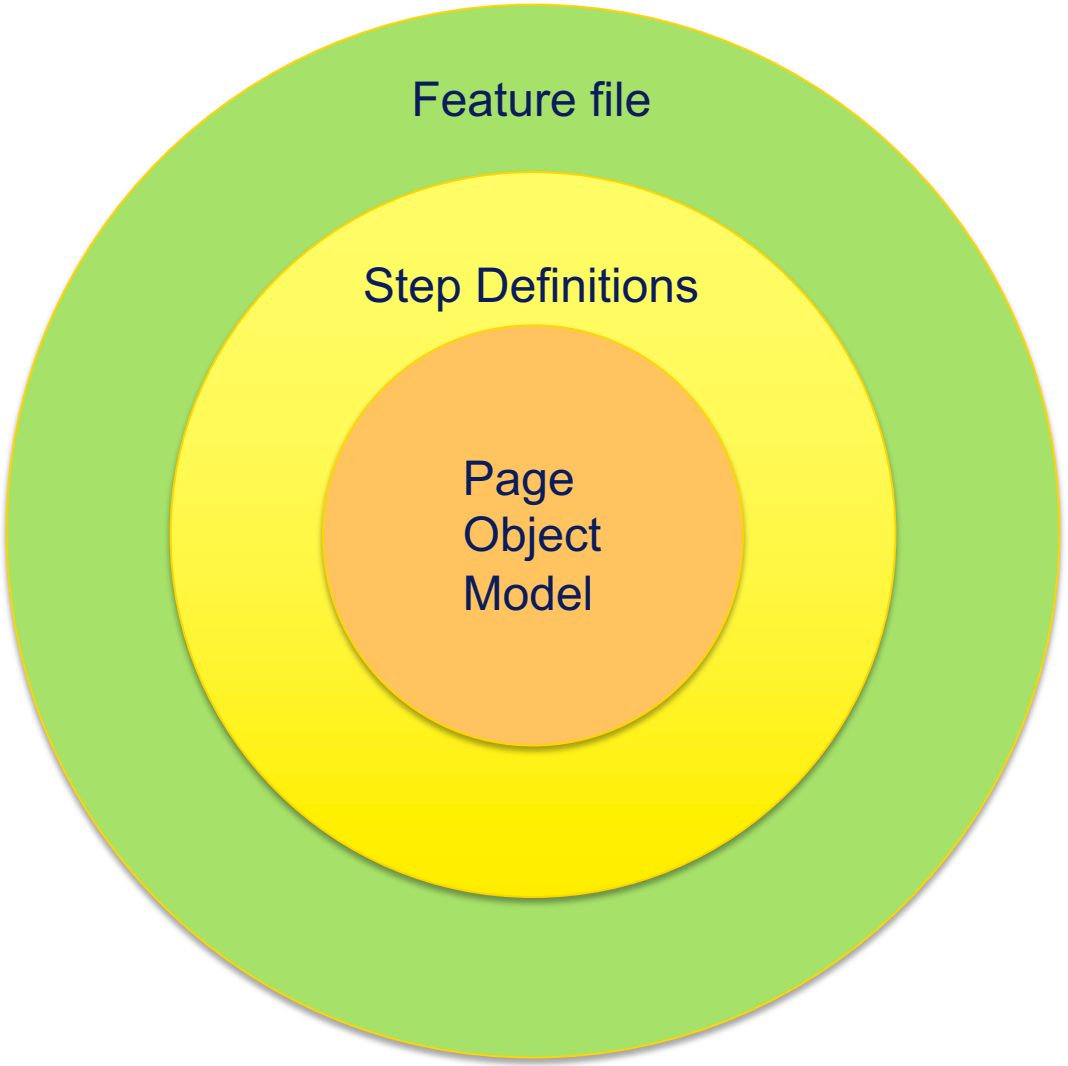
What is Cucumber?

- Cucumber == BDD?
  - No
- We use BDD so we have to use Cucumber!
  - No
- Cucumber is a tool
  - Yes, it's an open source collaboration tool
- Cucumber makes my test framework more accessible
  - Yes, when implemented correctly
- Cucumber means more code for me to write
  - Yes…wait what…..

Cucumber means more code to write

- You could write **methods** to mimic Gherkin
  - Test will then be build with OOP
- More code == more effort

Cucumber means more code to write



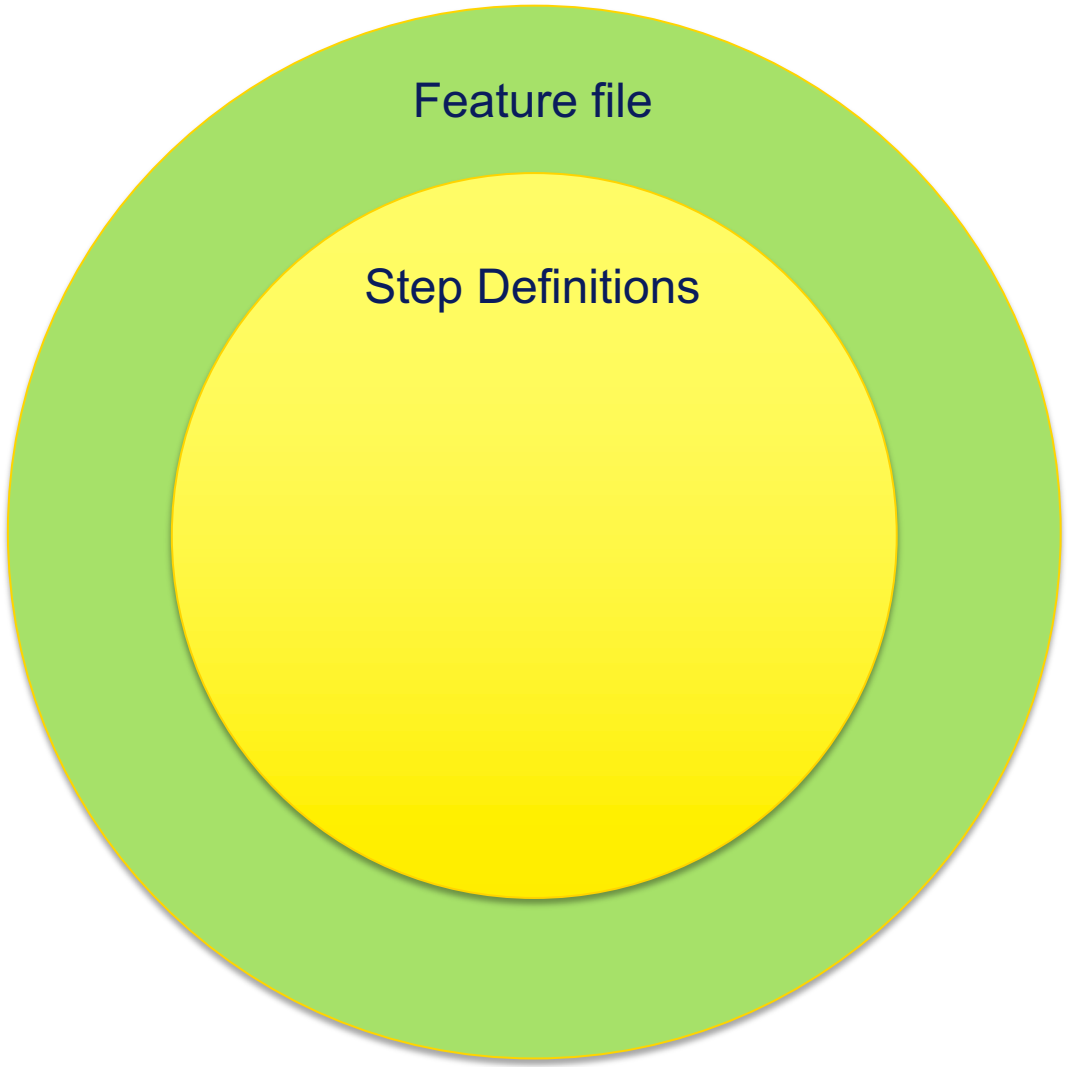
Cucumber enhances accessibility

- Correct implementation will ensure this
- Enhanced accessibility is worth writing more code!

Cucumber enhances accessibility

- Correct implementation will ensure this
- Enhanced accessibility is worth writing more code!
- Scope for today: improve Cucumber implementation

Technical scope of today

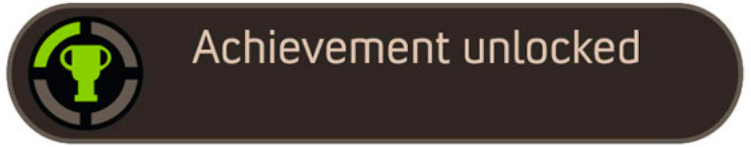


Page Object Model

Technical scope of today

- Provided with a working framework containing a full Page Object Model, step definitions and the (empty) feature files
- Throughout the workshop we will improve this framework
- After the workshop the full framework with the “correct” code will be made available

And that was the long introduction



Overview

- Get the code
- Quick and dirty login
- Login the “right” way
- Generic becomes the enemy*
- Shorten the test
- Code reset for all
- Contradiction: Let the assertions rise up

* = theory only

Exercise 0.1 – Get the code

- URL: https://github.com/MichelAmin47/AgileTdWorkshop.git
- Navigate to your preferred folder
- git clone https://github.com/MichelAmin47/AgileTdWorkshop.git

Exercise 0.1 – Get the code

- From the USB copy the contents from the folder
  - ThisOneAtTheBeginning
- Into a new folder locally (e.g. AgileTdWorkshop)

Exercise 0.1 – Get the code

- Select the pom.xml to open the project as a Maven project
- <your folder>/AgileTdWorkshop/src/pom.xml
  - Wait for all the maven dependencies to load

Exercise 0.1 – Get the code

- Run the smoke test:
- AgileTdWorkshop/src/test/resources/features/SmokeTest.feature
- AgileTdWorkshop/src/test/java/runners/RunCukesSmokeTest

Exercise 0.2 – Create an account

- Please go to: https://webshop.mobiletestautomation.nl/login

- Create an account
  - No email validation

Exercise 1 – Quick and dirty login

- Create a quick login test together
  - test/resources/features/exercises/FirstExercise.feature
- The step definition code
  - Test/java/stepdefinitions/exercises/StepdefsLogin

Exercise 1 – Quick and dirty login

FirstExercise.feature

```
1  Feature: Login
2
3    @exerciseOne
4    Scenario: User log in
5      Given   User is on the homepage
6      When
```

```
@exerciseOne
Scenario: User log in
  Given   User is on the homepage
  When    U
```

User is on the homepage
User click login button
User clicks sign in button
User fills in email
User fills in password
The user is on the homepage
The user logs into his MyAccount
The user logs into his MyAccount with "<string>" and "<string>" as cr…
The user should be taken to his Account page
The user submits his first address
MyAccount elements visible
The new address should be show on the address page

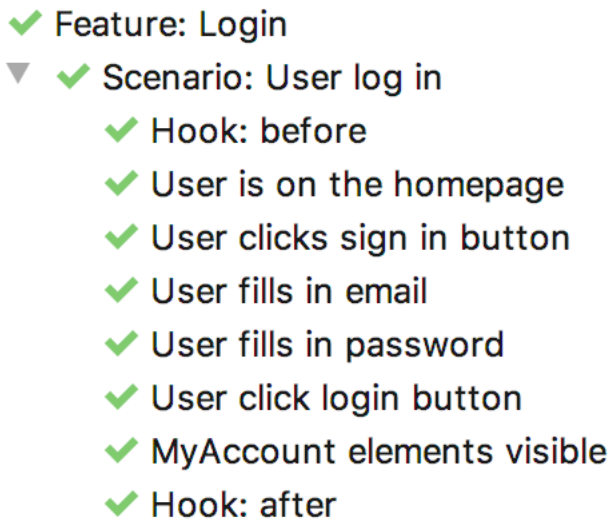
Press ⏎ to insert, ⇥ to replace Next Tip

Let’s code!

VALORI

Exercise 1 – Quick and dirty login

- Not written in an English-like language.
- This is your report



Exercise 1 – Quick and dirty login

- Not written in an English-like language.
- Each step describes an action…

```
@exerciseOne
Scenario: User log in
  Given   User is on the homepage
  When    User clicks sign in button
  And     User fills in email
  And     User fills in password
  And     User click login button
  Then    MyAccount elements visible
```

Exercise 1 – Quick and dirty login

- Resulting in Stepdef code like this:

```
@And("User fills in email")
public void userFillsInEmail() {
    authenticationPage.fillInEmail("test@tester.com");
}

@And("User fills in password")
public void userFillsInPassword() {
    authenticationPage.fillInPassword("1qazxsw2");
}

@And("User click login button")
public void userClickLoginButton() {
    authenticationPage.clickSubmitButton();
}
```

Exercise 1 – Quick and dirty login

- This extra code which does not benefit your framework

```
@And("User fills in email")
public void userFillsInEmail() {
    authenticationPage.fillInEmail("test@tester.com");
}

@And("User fills in password")
public void userFillsInPassword() {
    authenticationPage.fillInPassword("1qazxsw2");
}

@And("User click login button")
public void userClickLoginButton() {
    authenticationPage.clickSubmitButton();
}
```

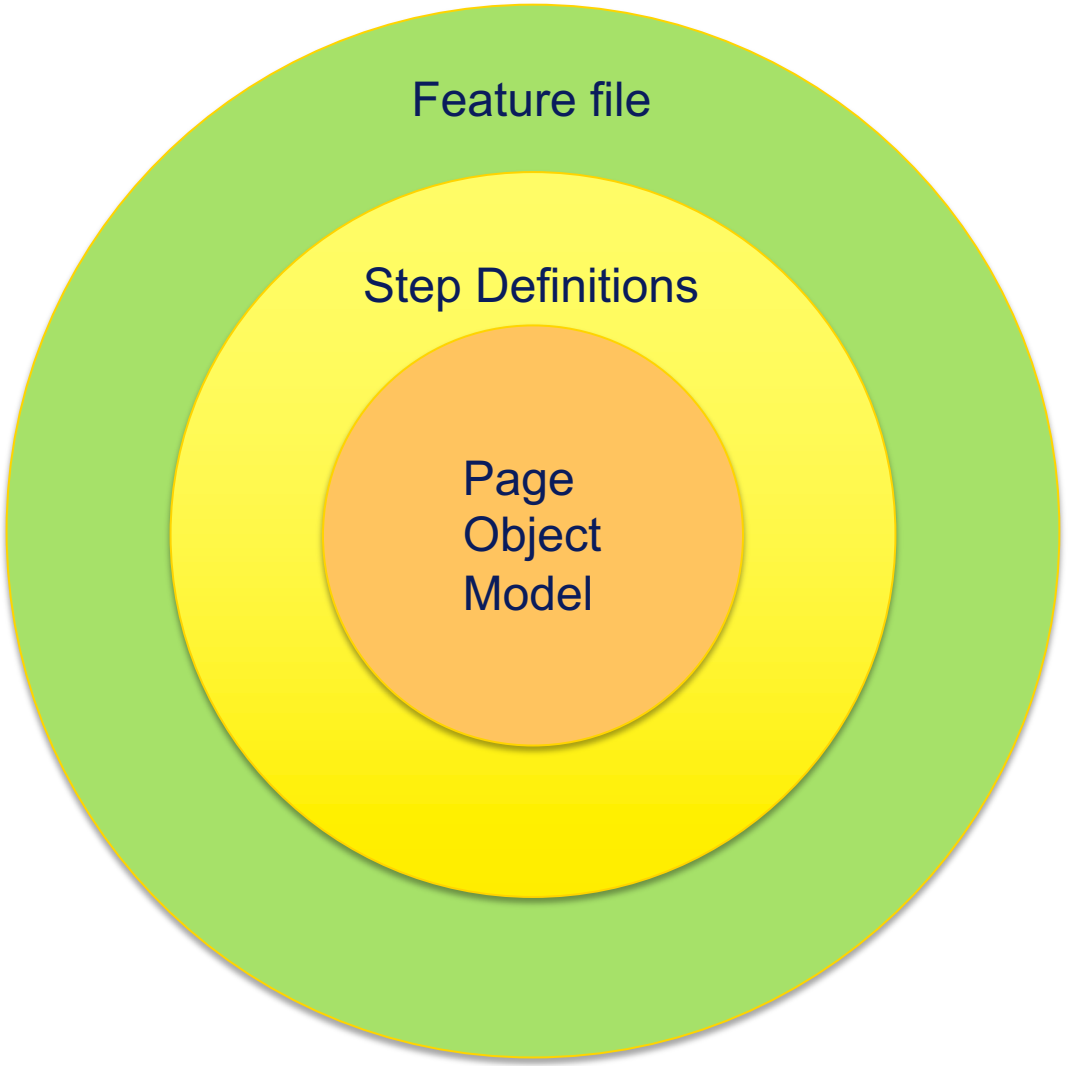
Exercise 1 – Quick and dirty login

- This extra code creates more code on a Page Object Model level

```
@And("User fills in email")
public void userFillsInEmail() {
    authenticationPage.fillInEmail("test@tester.com");
}

@And("User fills in password")
public void userFillsInPassword() {
    authenticationPage.fillInPassword("1qazxsw2");
}

@And("User click login button")
public void userClickLoginButton() {
    authenticationPage.clickSubmitButton();
}
```

```
public void fillInEmail(String email){
    emailTextfield.sendKeys(email);
}

public void fillInPassword(String password){
    passwordTextfield.sendKeys(password);
}

public void clickSubmitButton(){
    loginButton.click();
}
```

Exercise 1 – Quick and dirty login



Exercise 1 – Quick and dirty login

- Let the Page Object Model handle the actions

```
@When("The user logs into his MyAccount with \"([^\"]*)\" and \"([^\"]*)\" as credentials")
public void theUserLogsIntoHisMyAccountWithAndAsCredentials(String email, String password) {
    homePage = new HomePage(driver);
    homePage.clickLogIn();
    authenticationPage.login(email,password);
}
```

```
public void login(String email, String password) {
    emailTextfield.sendKeys(email);
    passwordTextfield.sendKeys(password);
    loginButton.click();
}
```

Exercise 2 – Login the “right” way

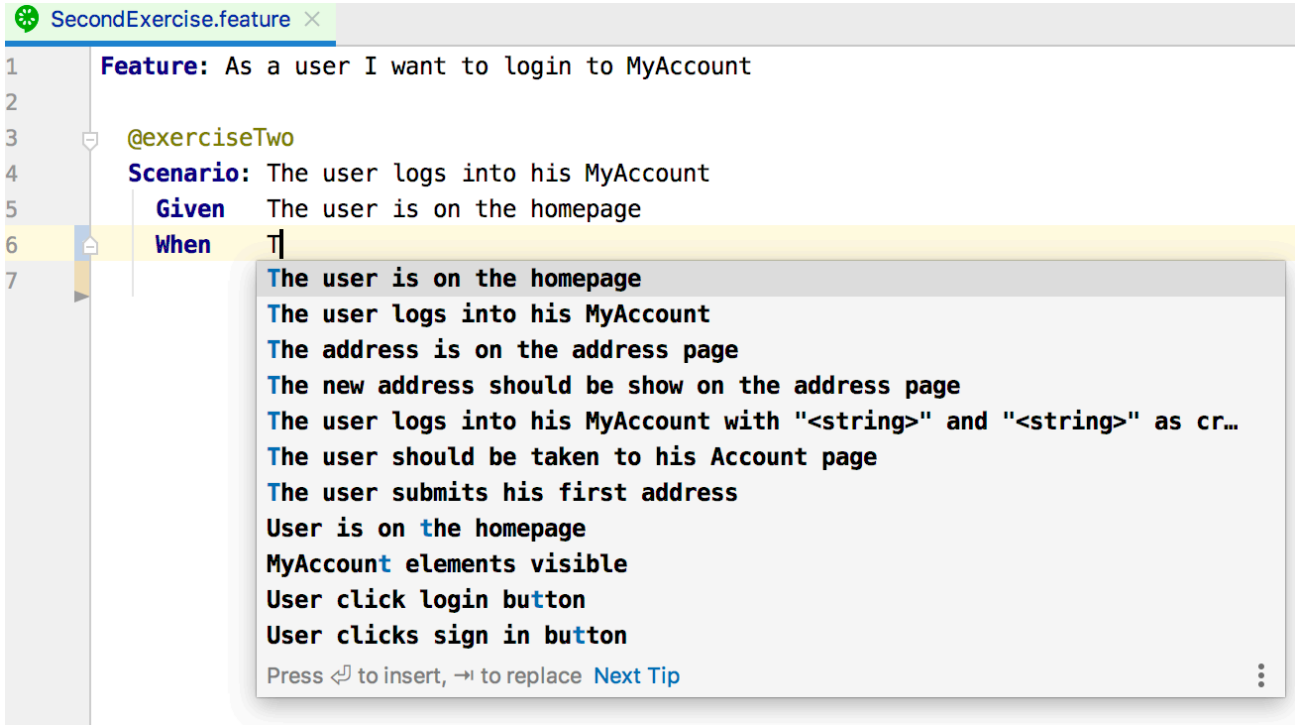
- Lets try that again ☺

- Rewrite the login test by using more behavior driven step defs
  - The IDE will show you what’s available
  - The methods for these “better” Stepdefs are empty
  - Use the statements from the Stepdefs from the previous exercise to create a test

- Use your own credentials

Exercise 2 – Login the “right” way

- The IDE will show you what’s available



- Feature file:
  - test/resources/features/exercises/SecondExercise.feature

Exercise 2 – Login the “right” way

- The methods for these “better” Stepdefs are empty

- The step definition code
  - Test/java/stepdefinitions/exercises/StepdefsLogin

Exercise 2 – Login the “right” way

- Use the statements from the Stepdefs from the previous exercise to create a test and fill the new Stepdef methods

```
@And("User fills in email")
public void userFillsInEmail() {
    authenticationPage.fillInEmail("test@tester.com");
}

@And("User fills in password")
public void userFillsInPassword() {
    authenticationPage.fillInPassword("1qazxsw2");
}

@And("User click login button")
public void userClickLoginButton() {
    authenticationPage.clickSubmitButton();
}
```

Use your own credentials!

Exercise 2 – Login the “right” way (Extra)

- Create a test in which the credentials can be used as parameters in the stepDef method
- Arguments will then be your email and password

Let’s code!

VALORI

Exercise 2 – Login the “right” way

- written in an English-like language

```
Feature: As a user I want to login to MyAccount

  @exerciseTwo
  Scenario: The user logs into his MyAccount
    Given   The user is on the homepage
    When    The user logs into his MyAccount
    Then    The user should be taken to his Account page

  @exerciseTwo
  Scenario: The user logs into his MyAccount with credentials
    Given   The user is on the homepage
    When    The user logs into his MyAccount with "test@tester.com" and "1qazxsw2" as credentials
    Then    The user should be taken to his Account page
```

Exercise 2 – Login the “right” way

- Even more important

```
@When("The user logs into his MyAccount")
public void theUserLogsIntoHisMyAccount() {
    homePage = new HomePage(driver);
    homePage.clickLogIn();
    authenticationPage.fillInEmail("test@tester.com");
    authenticationPage.fillInPassword("1qazxsw2");
    authenticationPage.clickSubmitButton();
}

@When("The user logs into his MyAccount with \"([^\"]*)\" and \"([^\"]*)\" as credentials")
public void theUserLogsIntoHisMyAccountWithAndAsCredentials(String email, String password) {
    homePage = new HomePage(driver);
    homePage.clickLogIn();
    authenticationPage.login(email,password);
}
```

- Multiple actions are combined to describe the behavior of the system and the end user

Generic becomes the enemy

- My first step def

```
When    Driver clicks element with "css" locator "div.generic_ftw"
```

Generic becomes the enemy

- Duplicate code is the enemy
  - Object-Oriented Design Principle: DRY (Don’t Repeat Yourself)
- Keep it generic == reusable
  - Avoid code duplication
  - Build tests quicker
- But when are we taking it to far?

Generic becomes the enemy

- When an input is directly used, we're good

```
When    The user logs into his MyAccount with "test@tester.com" and "1qazxsw2" as credentials
```

Generic becomes the enemy

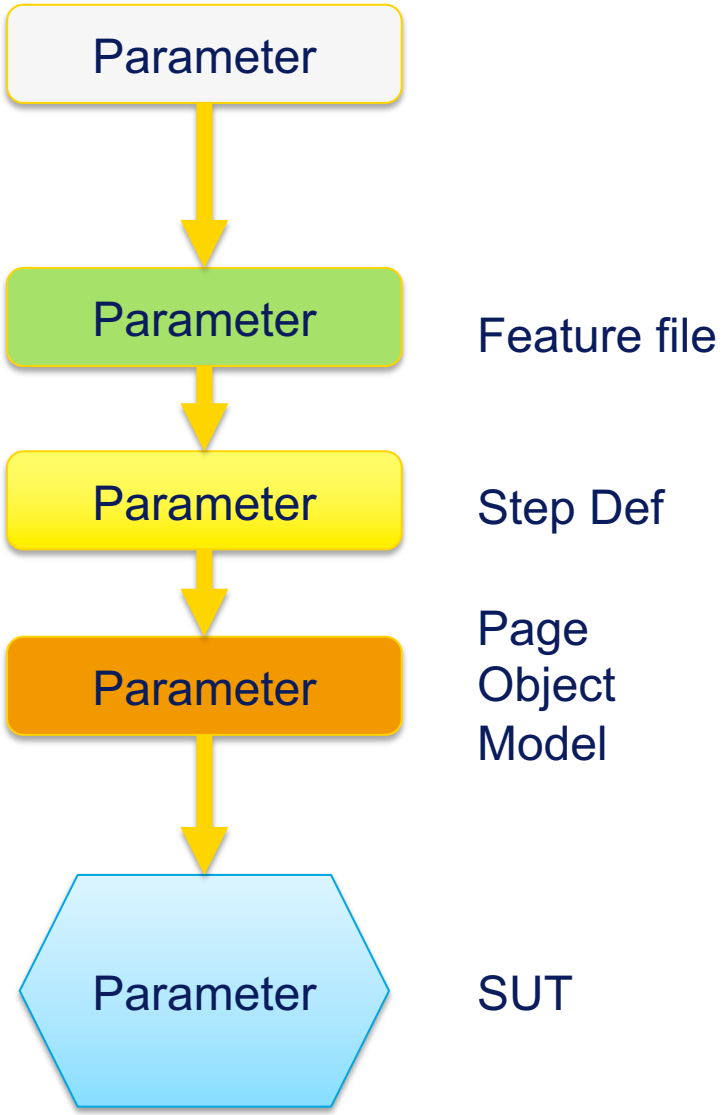
- When an input is directly used, we're good

```
@When("The user logs into his MyAccount with \"([^\"]*)\" and \"([^\"]*)\" as credentials")
public void theUserLogsIntoHisMyAccountWithAndAsCredentials(String email, String password) {
    homePage = new HomePage(driver);
    homePage.clickLogIn();
    authenticationPage.login(email,password);
}
```

```
public void login(String email, String password) {
    driver.findElement(By.cssSelector("input[type='email']")).sendKeys(email);
    driver.findElement(By.cssSelector("input[type='password']")).sendKeys(password);
    loginButton.click();
}
```

Generic becomes the enemy

- When an input is directly used, we're good



Generic becomes the enemy

- When an input has to be used for conditional logic…
- …we're on a slippery slope

```
When    The user logs into his MyAccount with "tester" credentials
```

Generic becomes the enemy

```
@When("The user logs into his MyAccount with \"([^\"]*)\" credentials")
public void theUserLogsIntoHisMyAccountWithCredentials(String userType) {
    if (userType.equals("tester")){
        authenticationPage.login( email: "tester@test", password: "1qazxsw2");
    }
    else if (userType.equals("regular")){
        authenticationPage.login( email: "regular@test", password: "1qazxsw2");
        authenticationPage.checkNoAdminConsole();
    }
    else if (userType.equals("admin")){
        authenticationPage.login( email: "admin@test", password: "1qazxsw2");
        authenticationPage.checkAdminConsole();
    }
    else if (userType.equals("BA")){
        authenticationPage.login( email: "BA@test", password: "1qazxsw2");
        authenticationPage.checkNoAdminConsole();
        authenticationPage.checkBAConsole();
    }
    else if (userType.equals("no_user")){
        authenticationPage.login( email: "blabla@test", password: "1qazxsw2");
        authenticationPage.noEntry();
    }
    else if (userType.equals("blocked")){
        authenticationPage.login( email: "blocked@test", password: "1qazxsw2");
        authenticationPage.noEntry();
        authenticationPage.explicitWarning();
    }
}
```

People will need to have a look at this code to know what they can use as input

Generic becomes the enemy

```
@When("The user logs into his MyAccount with \"([^\"]*)\" credentials")
public void theUserLogsIntoHisMyAccountWithCredentials(String userType) {
    if (userType.equals("tester")){
        authenticationPage.login( email: "tester@test", password: "1qazxsw2");
    }
    else if (userType.equals("regular")){
        authenticationPage.login( email: "regular@test", password: "1qazxsw2");
        authenticationPage.checkNoAdminConsole();
    }
    else if (userType.equals("admin")){
        authenticationPage.login( email: "admin@test", password: "1qazxsw2");
        authenticationPage.checkAdminConsole();
    }
    else if (userType.equals("BA")){
        authenticationPage.login( email: "BA@test", password: "1qazxsw2");
        authenticationPage.checkNoAdminConsole();
        authenticationPage.checkBAConsole();
    }
    else if (userType.equals("no_user")){
        authenticationPage.login( email: "blabla@test", password: "1qazxsw2");
        authenticationPage.noEntry();
    }
    else if (userType.equals("blocked")){
        authenticationPage.login( email: "blocked@test", password: "1qazxsw2");
        authenticationPage.noEntry();
        authenticationPage.explicitWarning();
    }
}
```

Does not really improve accessibility, right?

Generic becomes the enemy

- No harm in splitting it up

```
When    The user logs into his MyAccount as a regular user
```

```
When    The user logs into his MyAccount as a BA user
```

Generic becomes the enemy

- The step definition executes code, there is are no conditions that need to be met

```
@When("The user logs into his MyAccount as a regular user")
public void theUserLogsIntoHisMyAccountAsARegularUser() {
    authenticationPage.login( email: "regular@test", password: "1qazxsw2");
    authenticationPage.checkNoAdminConsole();
}

@When("The user logs into his MyAccount as a BA user")
public void theUserLogsIntoHisMyAccountAsABAUser() {
    authenticationPage.login( email: "BA@test", password: "1qazxsw2");
    authenticationPage.checkNoAdminConsole();
    authenticationPage.checkBAConsole();
}
```

Let’s take it up a notch

Time to write own step defs! ☺

Exercise 3 – Shorten the test

- Exercise 3 contains a test which is too long and too implicit
  - Use your own credentials
- Rewrite the steps in this feature, make it shorter and more behavior driven
  - test/resources/features/exercises/ThirdExercise.feature
- Rewrite the Step definitions to fit the new shorter feature
  - Test/java/stepdefinitions/exercises/StepDefAddress
  - The IDE can help you
  - You can use the methods from addressPage, no need to adjust the Page Object Model (unless you want too ☺)
  - Main/java/pages/AddressPage
- Question before you begin: “Does the test really need the input for address to be parameters?”

Exercise 3 – Shorten the test

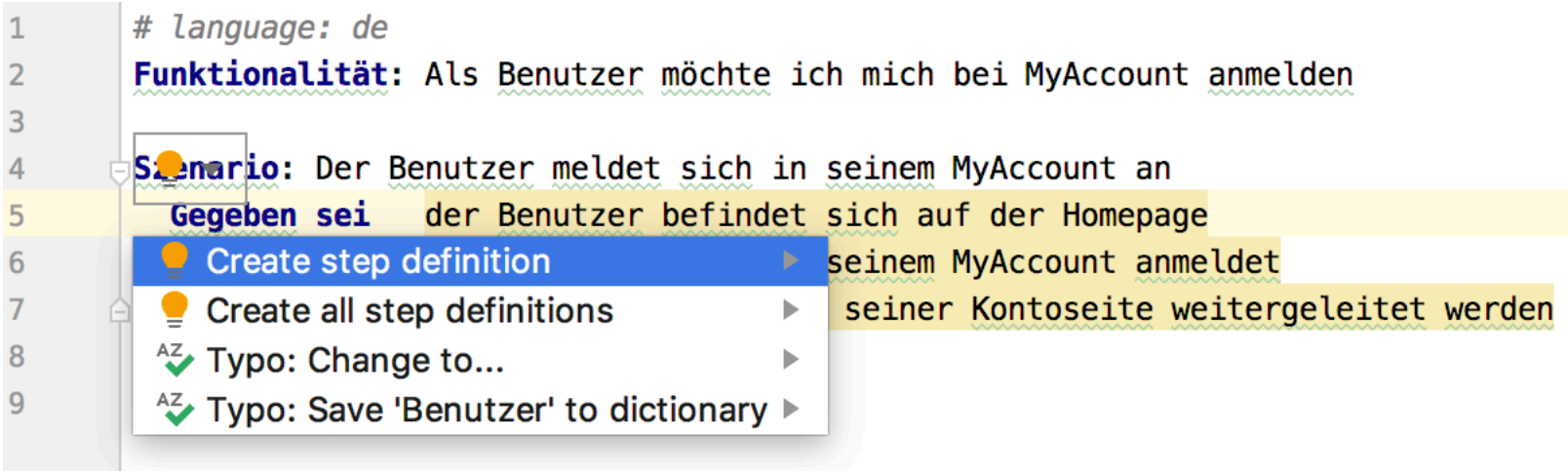
- Exercise 3 contains a test which is too long and too implicit
  - Use your own credentials
- Rewrite the steps in this feature
- Make it shorter and more behavior driven
  - test/resources/features/exercises/ThirdExercise.feature

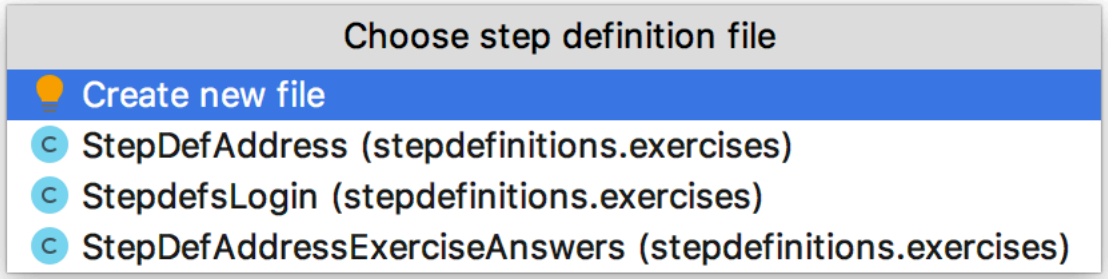
```
@exerciseThree
Scenario: The user adds his first adress to MyAccount
  Given   The user is on the homepage
  When    User clicks sign in button
  And     User fills in email
  And     User fills in password
  And     User click login button
  And     clicks add first address
  And     checks for the new address page
  And     fills in Alias "retseT"
  And     fills in First Name "Mister"
  And     fills in Last Name "Test"
  And     fills in Company "Tester.io"
  And     fills in VAT "4321"
  And     fills in Address "Testlane 23"
  And     fills in Address Complement "3A"
  And     fills in Postal Code "1111 AA"
  And     fills in City "Testdam"
  And     fills in Country ""
  And     fills in Phone "0031123456"
  And     clicks SAVE
  And     checks for the new address save message
  Then    The address is on the address page
  And     delete the address
  And     checks for deleted address message
```

Exercise 3 – Shorten the test

- Rewrite the Step definitions to fit the new shorter feature
  - Test/java/stepdefinitions/exercises/StepDefAddress
  - The IDE can help you
  - You can use the methods from addressPage, no need to adjust the Page Object Model (unless you want too ☺)
  - Main/java/pages/AddressPage

Exercise 3 – Shorten the test





Exercise 3 – Shorten the test

```
@And("checks for the new address page")
public void checksForTheNewAddressPage() { addressPage.validateAddressPage(); }

@And("fills in Alias \"([^\"]*)\"")
public void fillsInAlias(String alias) { addressPage.fillInAlias(alias); }

@And("fills in First Name \"([^\"]*)\"")
public void fillsInFirstName(String firstName) { addressPage.fillInFirstName(firstName); }

@And("fills in Last Name \"([^\"]*)\"")
public void fillsInLastName(String lastName) { addressPage.fillInLastName(lastName); }

@And("fills in Company \"([^\"]*)\"")
public void fillsInCompany(String company) { addressPage.fillInCompany(company); }

@And("fills in VAT \"([^\"]*)\"")
public void fillsInVAT(String vat) { addressPage.fillInVAT(vat); }

@And("fills in Address \"([^\"]*)\"")
public void fillsInAddress(String address) { addressPage.fillInAddress(address); }
```

Exercise 3 – Get the code (Git)

- git pull to get the code from the updated master branch

Exercise 3 – Get the code (USB)

- From the USB copy the contents from the folder
  - ThisOneForTheThirdExercise
- Into the folder with your code.
  - Yes, overwrite everything

Let’s code!

VALORI

Exercise 3 – Shorten the test

- Exercise 3 shorter story

```
@exerciseThree
Scenario: The user adds his first adress to MyAccount
  Given   The user is on the homepage
  When    The user logs into his MyAccount
  And     The user submits his first address
  Then    The new address should be show on the address page
```

Option 1

Option 2

```
@exerciseThree
Scenario: The user adds his first address to MyAccount
  Given   The user is on the homepage
  When    The user logs into his MyAccount
  And     The user submits his first address
  Then    The new address should be shown on the address page
  And     The newly added address is deleted
```

Exercise 3

- Exercise 3 with less stepDefs

```
@And("The user submits his first address")
public void theUserSubmitsHisFirstAddress() {
    myAccountPage = new MyAccountPage(driver);
    myAccountPage.clickToAddFirstAddres();
    addressPage.validateAddressPage();

    addressPage.fillInAlias("retseT");
    addressPage.fillInFirstName("Mister");
    addressPage.fillInLastName("Test");
    addressPage.fillInCompany("Tester.io");
    addressPage.fillInVAT("4321");
    addressPage.fillInAddress("Testlane 23");
    addressPage.fillInAddressCompl("3A");
    addressPage.fillInPostalCode("1111 AA");
    addressPage.fillInCity("Testdam");
    addressPage.fillInPhoneNumber("0031123456");
    addressPage.clickSaveButton();
    addressPage.validateNewAddressSaveMessage();
}
```

Exercise 3 – Shorten the test

- Exercise 3 shorter story

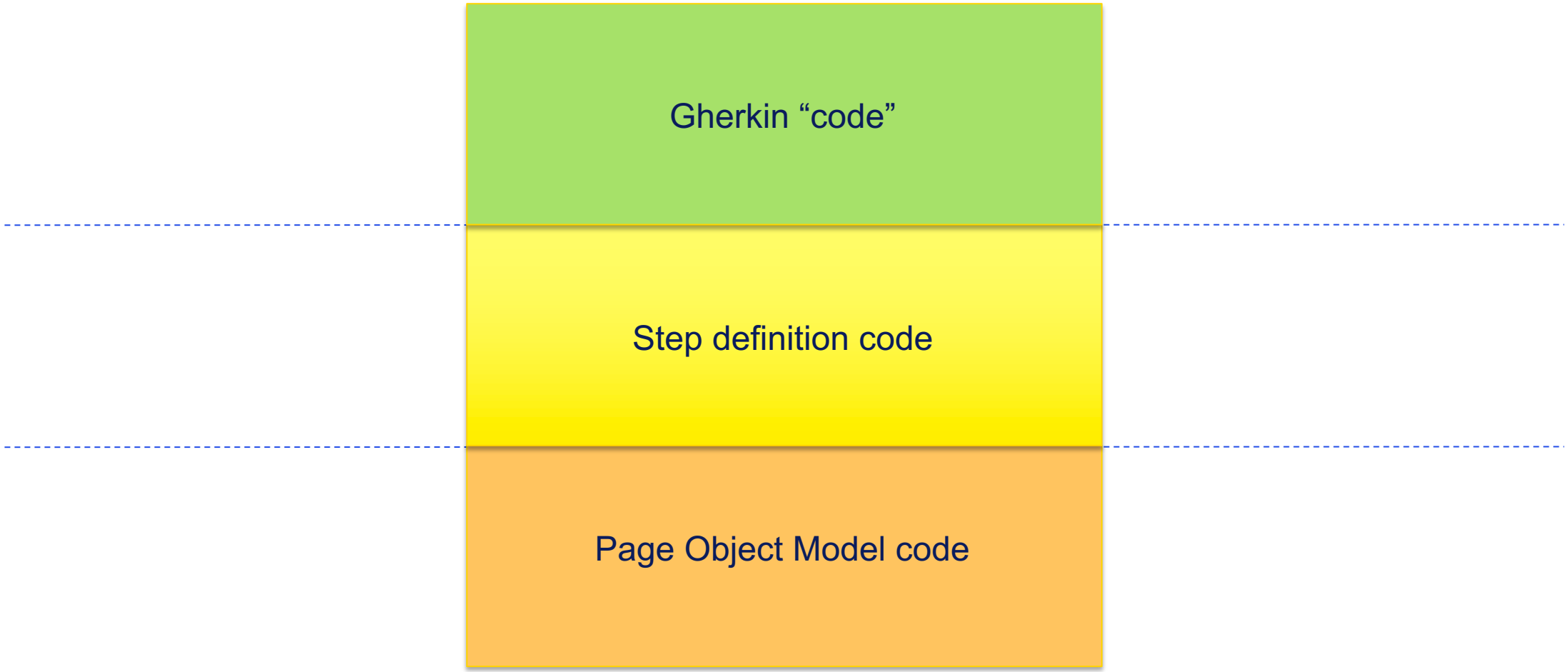
Option 1

```
@Then("The new address should be shown on the address page")
public void theNewAddressShouldBeShowOnTheAddressPage() {
    addressPage.validateNewlyAddedAddress();
    addressPage.deleteNewlyAddress();
    addressPage.validateDeletedAddressMessage();
}
```

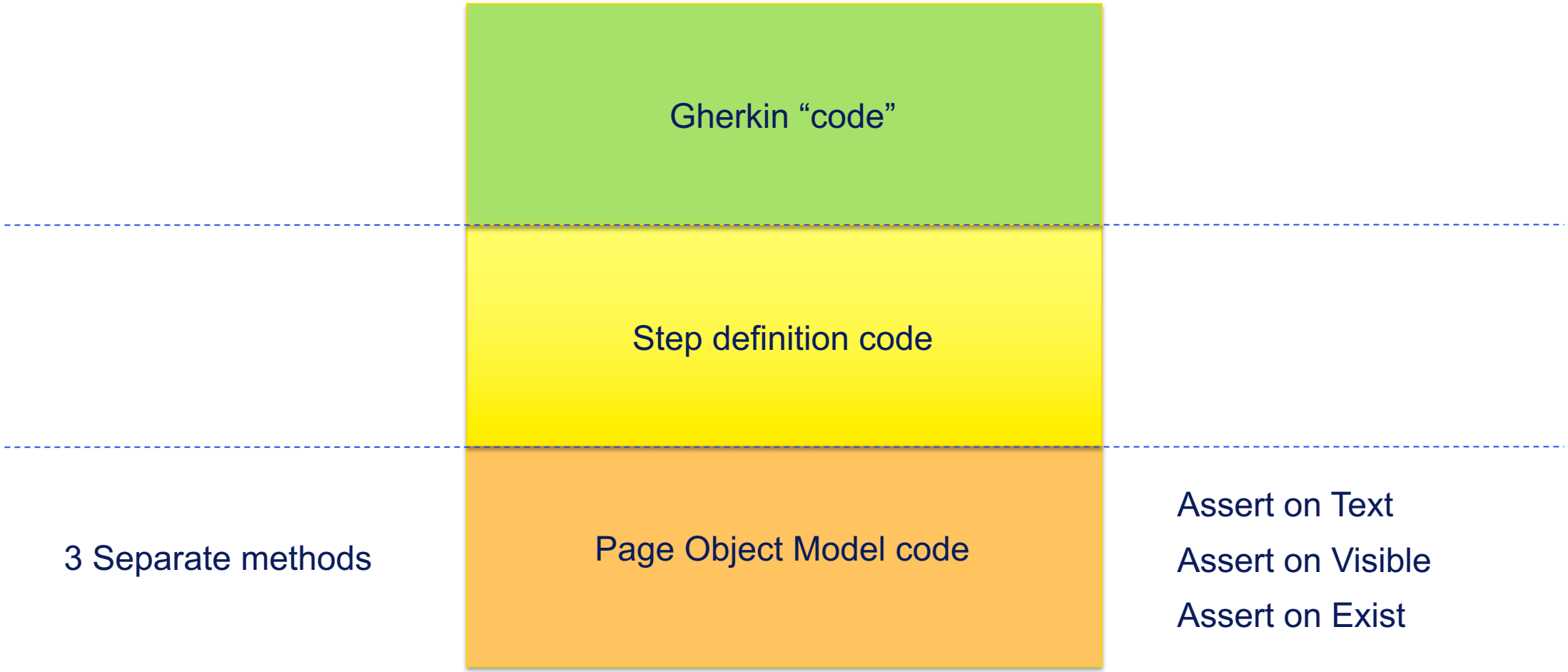
Option 2

```
@Then("The new address should be shown on the address page")
public void theNewAddressShouldBeShowOnTheAddressPage() {
    addressPage.validateNewlyAddedAddress();
}

@Then("The newly added address is deleted")
public void theNewlyAddedAddressIsDeleted() {
    addressPage.deleteNewlyAddress();
    addressPage.validateDeletedAddressMessage();
}
```

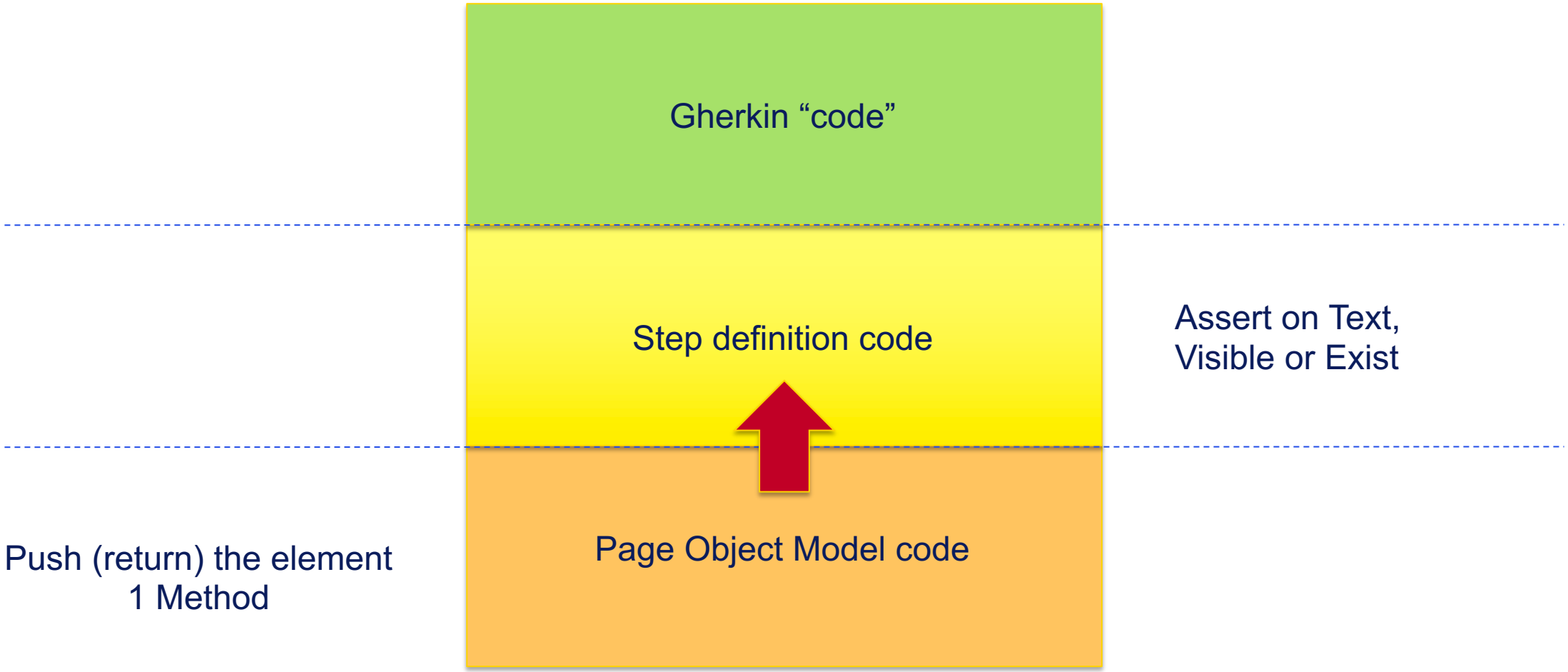
Contradictions



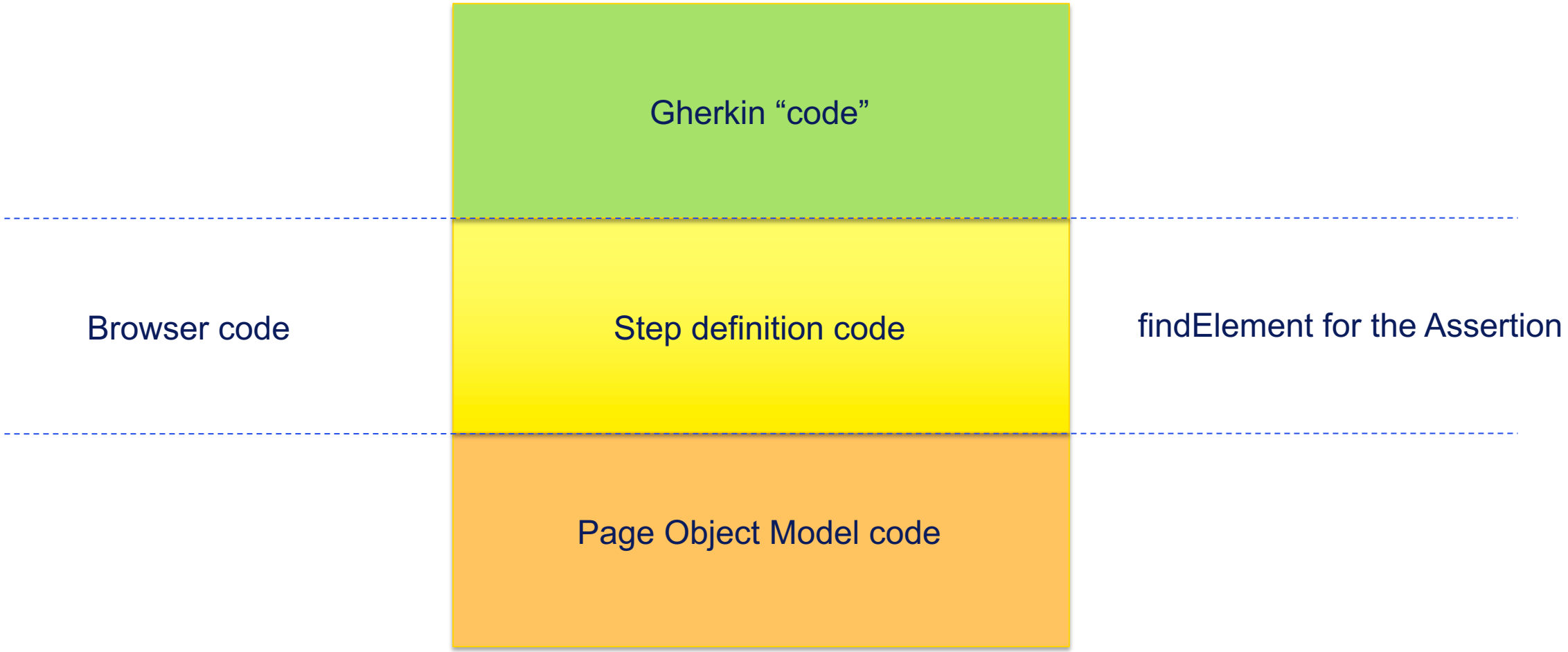
Contradictions



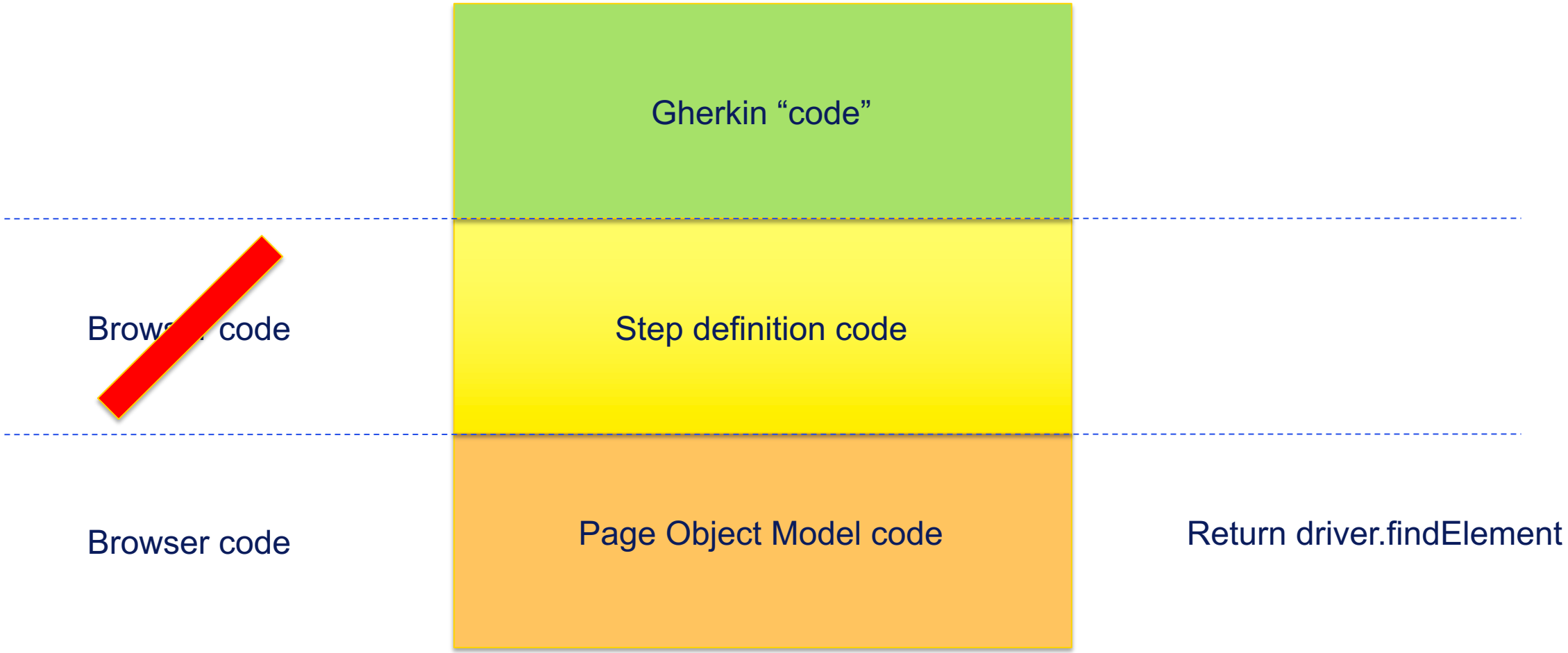
Contradictions



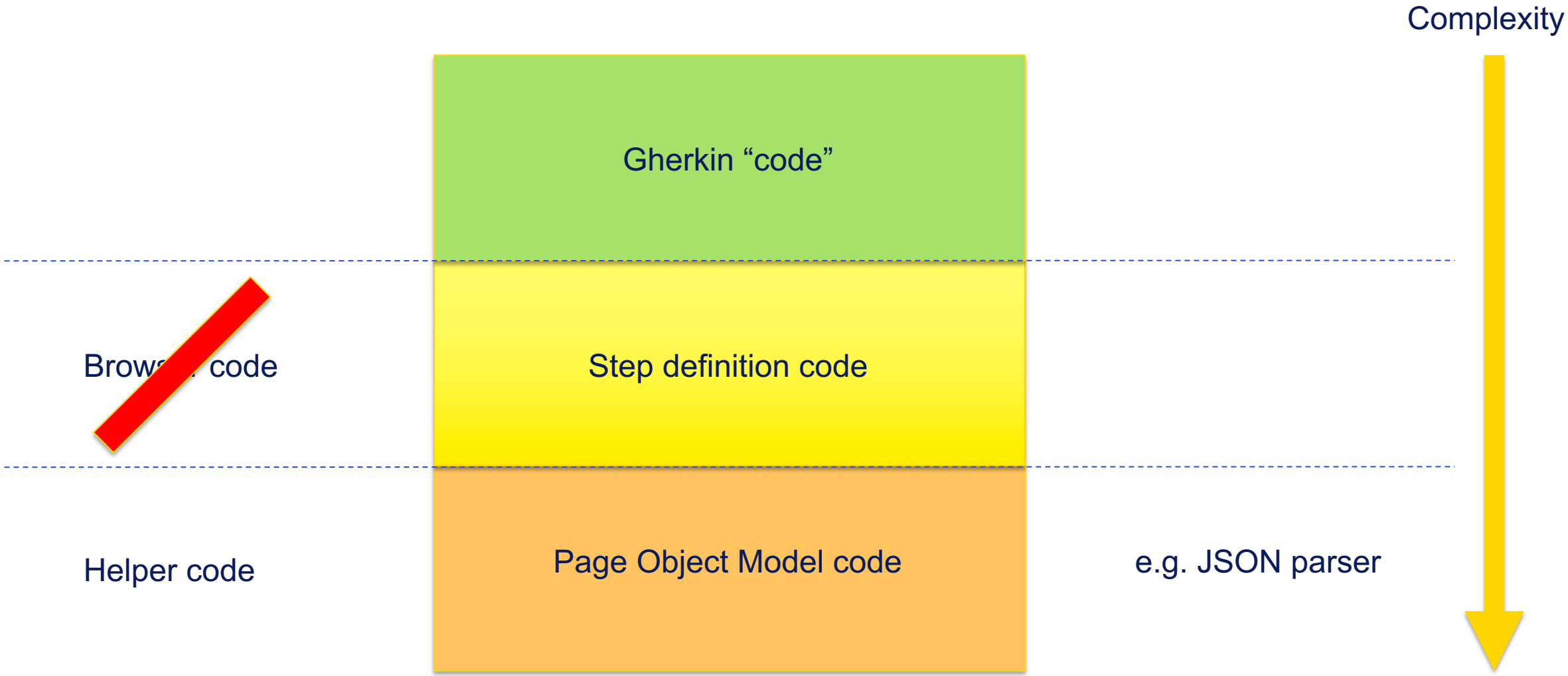
Contradictions



Contradictions



Contradictions



Contradictions

- Don’t Assert too low in your code.
- Hide the browser code as low as possible in your code

Exercise 4 – Get the code (Git)

- git pull to get the code from the updated master branch

Exercise 4 – Get the code (USB)

- From the USB copy the contents from the folder
  - ThisOneForTheFourth
- Into the folder with your code.
  - Yes, overwrite everything

Exercise 4 – Contradiction: Let the assertions rise up

- Exercise four is a copy of the outcome of exercise three

- All the assertions are in the Page Object Model
  - Main/java/pages/AddressPage

```
public void validateDeletedAddressMessage() {
    Assert.assertEquals( message: "Address was not deleted succesfully",  expected: "Address successfully deleted!",
            driver.findElement(successMessage).getText());
}
```

Exercise 4 – Contradiction: Let the assertions rise up

- Use the methods in the Page Object Model to push the element into the Stepdefs
  - Main/java/pages/AddressPage
  - (getter methods)

```
public WebElement getAddAddressElement() { return driver.findElement(newAddressHeader); }

public WebElement getNewAddressSavedMessageElement() { return driver.findElement(successMessage); }

public WebElement getNewlyAddedAddressElement() { return driver.findElement(addressBlock); }

public WebElement getNewAddressAliasElement() { return driver.findElement(addressAlias); }

public WebElement getAddressDeletedMessageElement() { return driver.findElement(successMessage); }
```

Exercise 4 – Contradiction: Let the assertions rise up

- And create Assertions in the StepDef methods
- Don’t forget to remove the calls to “Page Object Model assertion methods”
  - Double validation ☺
- And use your own credentials!

Exercise 4 – Contradiction: Let the assertions rise up

- Assert in the Page Object Model

```
myAccountPage = new MyAccountPage(driver);
myAccountPage.clickToAddFirstAddres();
addressPage.validateAddressPage();
```

- Assert in the stepDef

```
myAccountPage = new MyAccountPage(driver);
myAccountPage.clickToAddFirstAddres();

Assert.assertEquals( message: "Add new address page is not shown",  expected: "New address",
        addressPage.getAddAddressElement().getText());
```

Let’s code!

VALORI

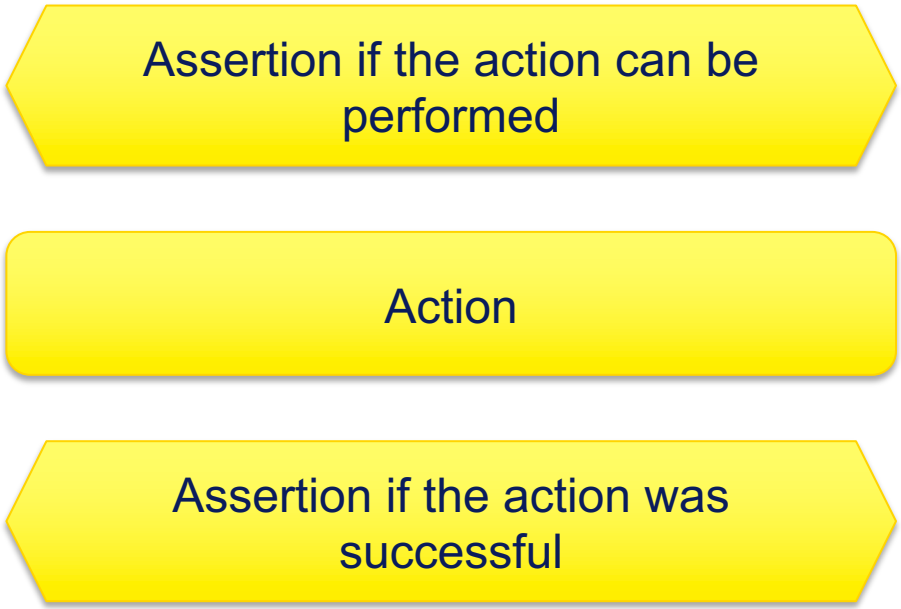
Exercise 4 – Contradiction: Let the assertions rise up

```
@And("The user submits his first address")
public void theUserSubmitsHisFirstAddress() {
    myAccountPage = new MyAccountPage(driver);
    myAccountPage.clickToAddFirstAddres();

    Assert.assertEquals( message: "Add new address page is not shown",  expected: "New address",
            addressPage.getAddAddressElement().getText());

    addressPage.fillInAlias("retseT");
    addressPage.fillInFirstName("Mister");
    addressPage.fillInLastName("Test");
    addressPage.fillInCompany("Tester.io");
    addressPage.fillInVAT("4321");
    addressPage.fillInAddress("Testlane 23");
    addressPage.fillInAddressCompl("3A");
    addressPage.fillInPostalCode("1111 AA");
    addressPage.fillInCity("Testdam");
    addressPage.fillInPhoneNumber("0031123456");
    addressPage.clickSaveButton();

    Assert.assertEquals( message: "Address was not saved succesfully",  expected: "Address successfully added!",
            addressPage.getNewAddressSavedMessageElement().getText());
}
```

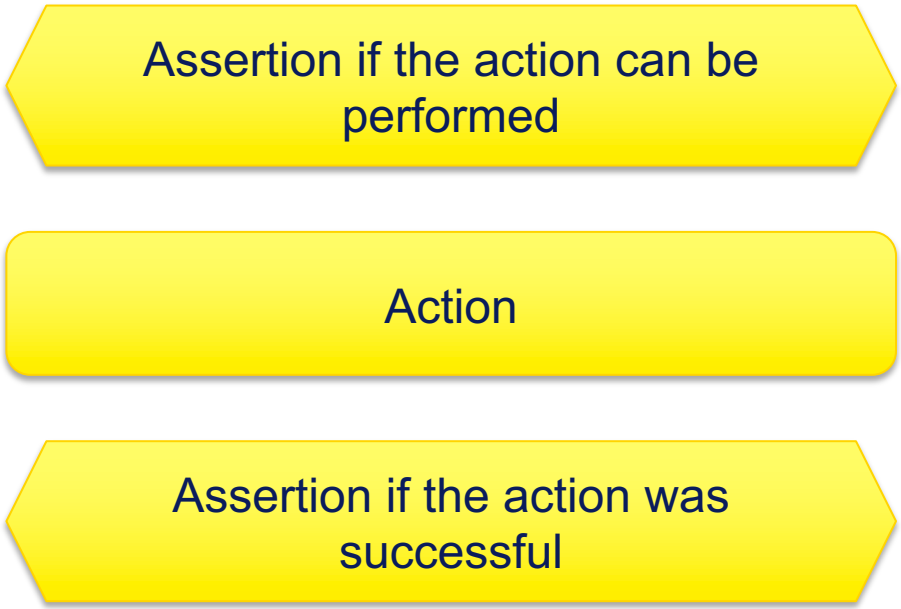
Exercise 4 – Contradiction: Let the assertions rise up

```
@Then("The new address should be show on the address page")
public void theNewAddressShouldBeShowOnTheAddressPage() {
    Assert.assertTrue( message: "Check if address block is visible",
            addressPage.getNewlyAddedAddressElement().isDisplayed());

    Assert.assertEquals( message: "Address alias not present",  expected: "retseT",
            addressPage.getNewAddressAliasElement().getText());

    addressPage.validateDeletedAddressMessage();

    Assert.assertEquals( message: "Address was not deleted succesfully",  expected: "Address successfully deleted!",
            addressPage.getAddressDeletedMessageElement().getText());
}
```

Contradictions – where to Assert

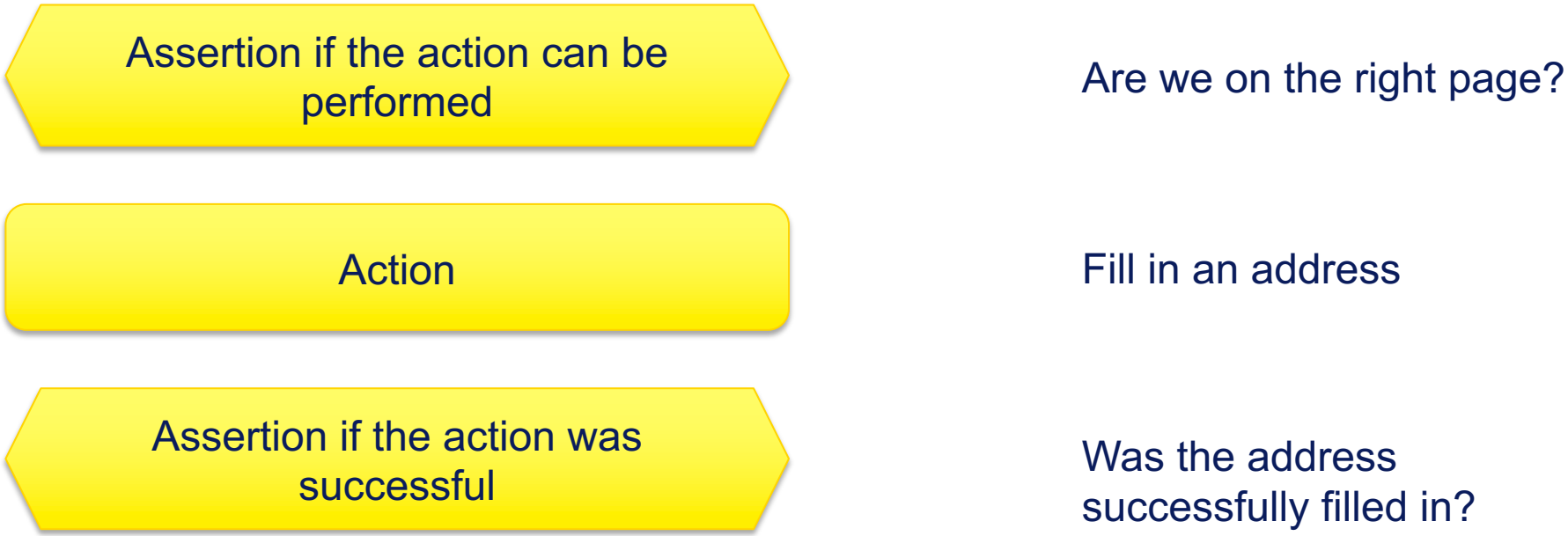


Contradictions – where to Assert

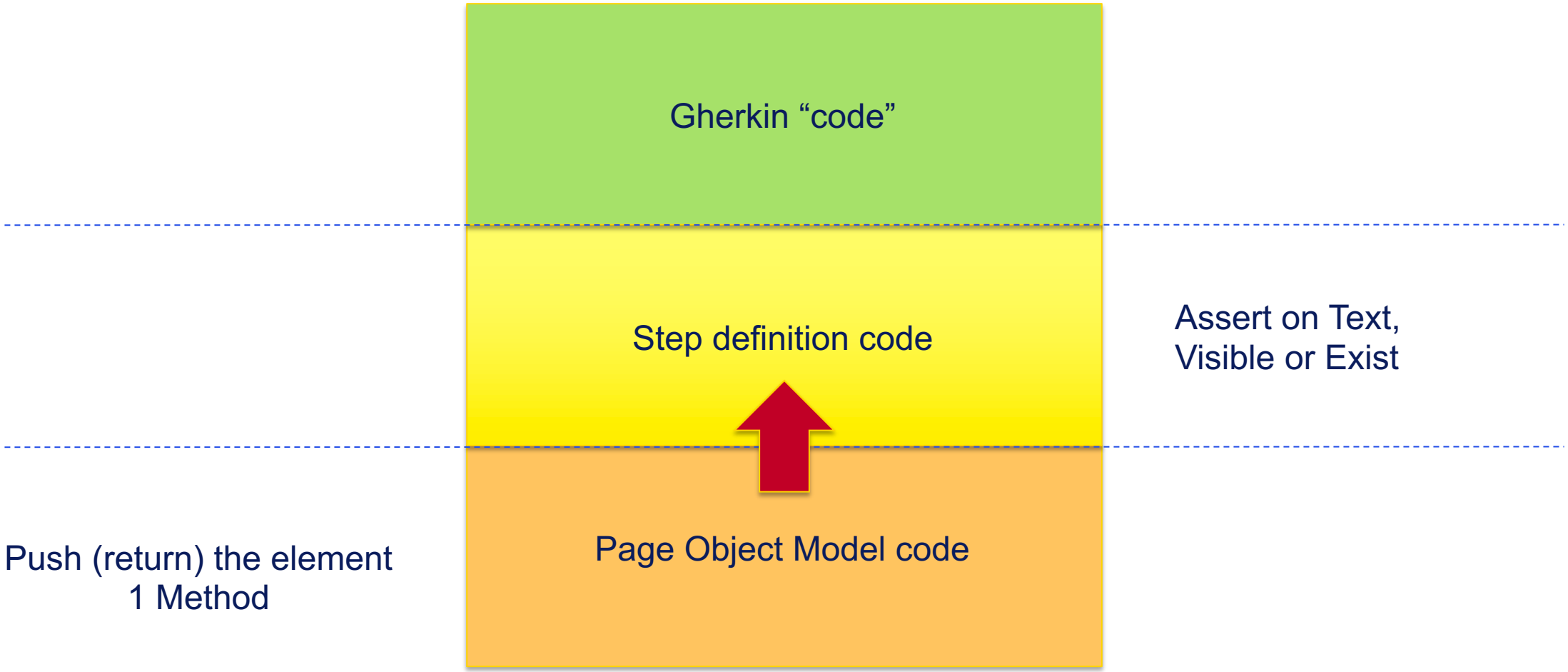


Fill in an address

Contradictions – where to Assert

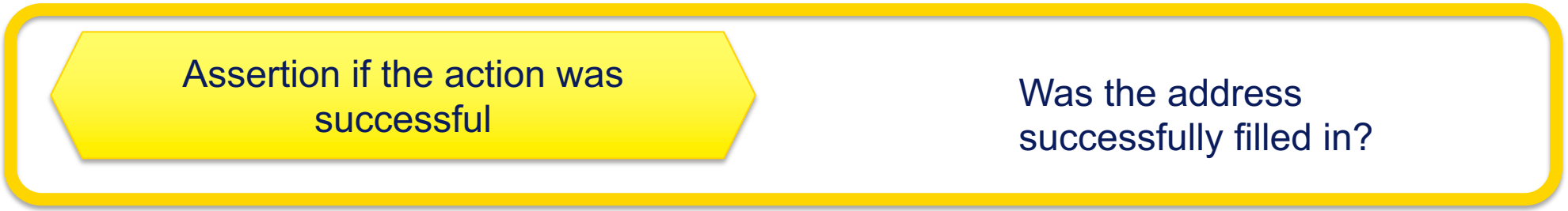


Contradictions

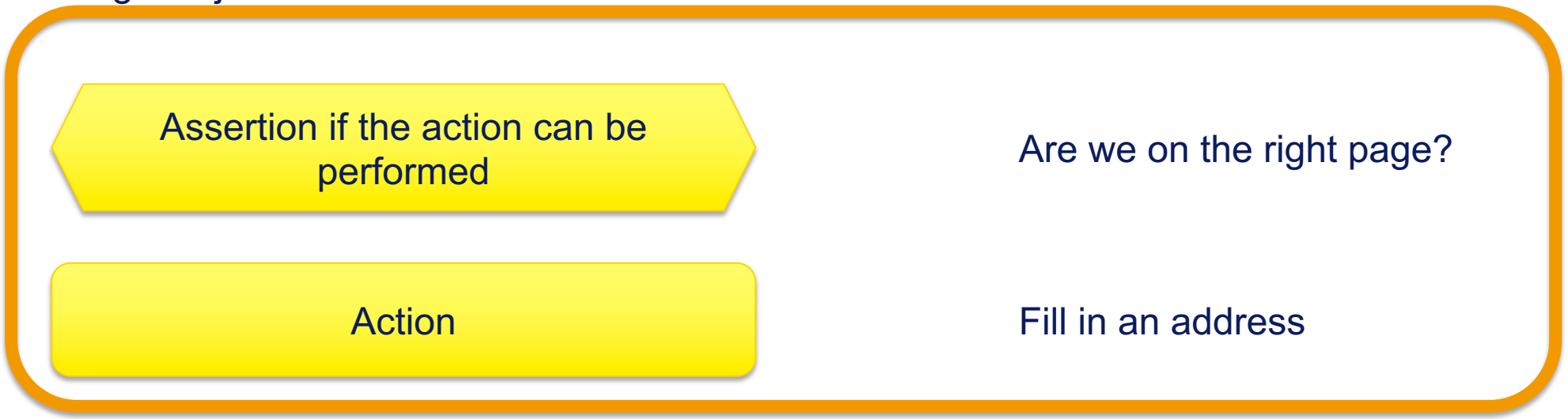


Contradictions – where to Assert

Step definition code



Page Object Model code



Take aways

- Cucumber is more work but can be worth it!
- Scenario’s should make sense, keep it as short as possible
- Don’t expose everything in a scenario
- Don’t create generic just so it’s generic
- Assert not to deep
- Hide the complex code as deep down as you can

Take aways

Cucumber implementation == Accessibility

Take aways

Cucumber implementation == Accessibility

Thank you for your attention.

Feedback welcome!

Go to agiletestingdays.com/session-ratings and give your rating!

Questions?
Twitter: @MichelAmin47
Email: michellalmohamed@valori.nl