

FUNCTIONAL TESTING

TEST

PBA SOFTWAREUDVIKLING /
BSC SOFTWARE DEVELOPMENT

Christian Nielsen cnls@cphbusiness.dk

Tine Marbjerg tm@cphbusiness.dk

SPRING 2019



TODAY'S TOPICS

- **Test types & levels**
 - Be able to compare the different test levels and their objectives
 - Be able to compare the different test types and their objectives
 - Be able to map terminology between traditional testing and agile
- **Test automation**
 - Be able to use the Test Pyramid as guideline to think about different layers of testing, and how much test automation to put into each layer.
- **Service layer testing: REST service**
 - Be able to use REST Assured to test REST service
- **Presentation layer testing: Automated system testing of Web pages**
 - Be able to automate test of web pages with Selenium WebDriver

TEST LEVELS AND TEST TYPES

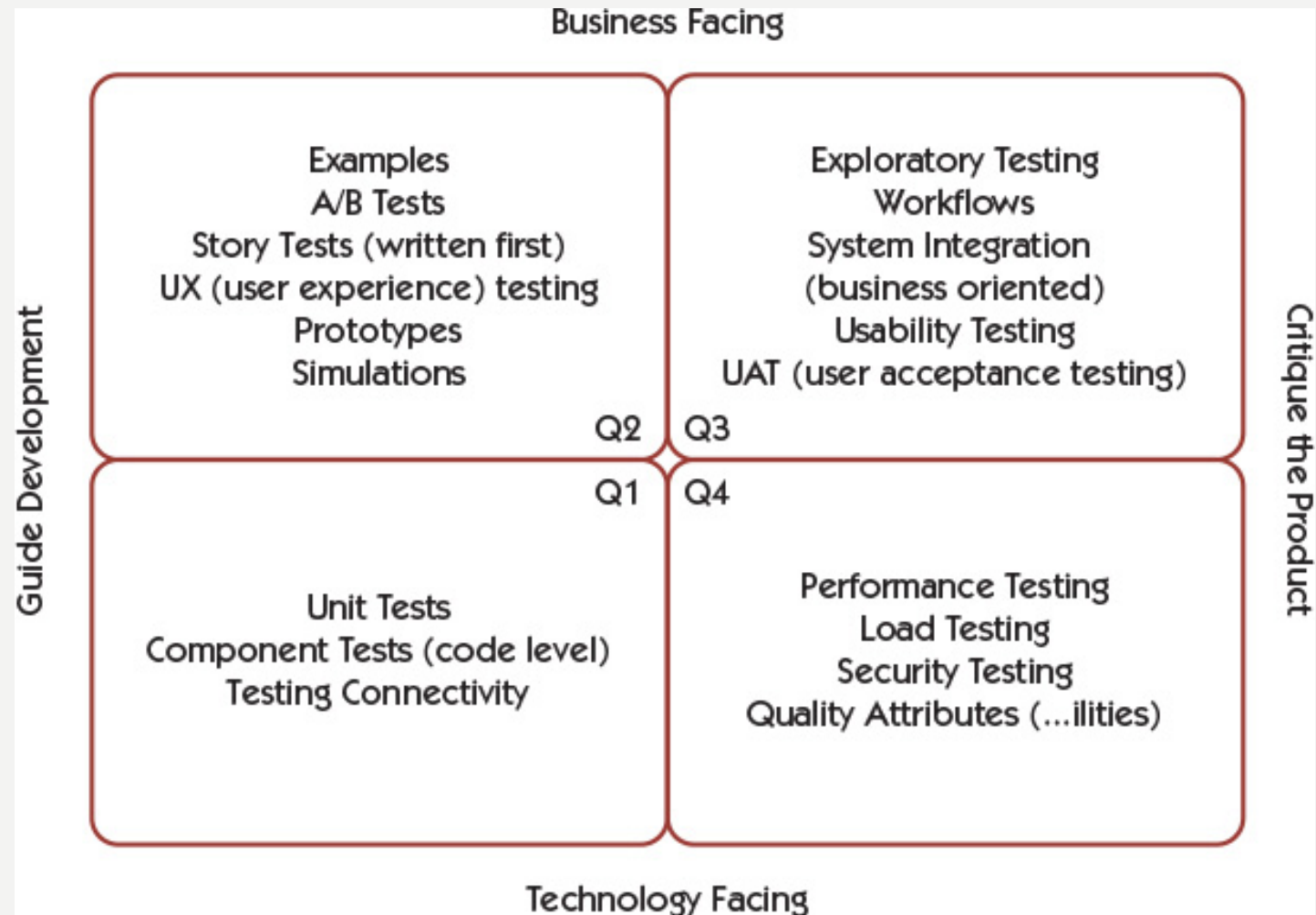
- **Test levels**

- Unit tests
 - Classes and methods
- Integration tests
 - Subsystems, databases, interfaces, API
- System testing (*acceptance testing*)
 - Applications
- Acceptance testing (*user acceptance testing*)
 - System, Configuration, Recovery systems, Converted Data

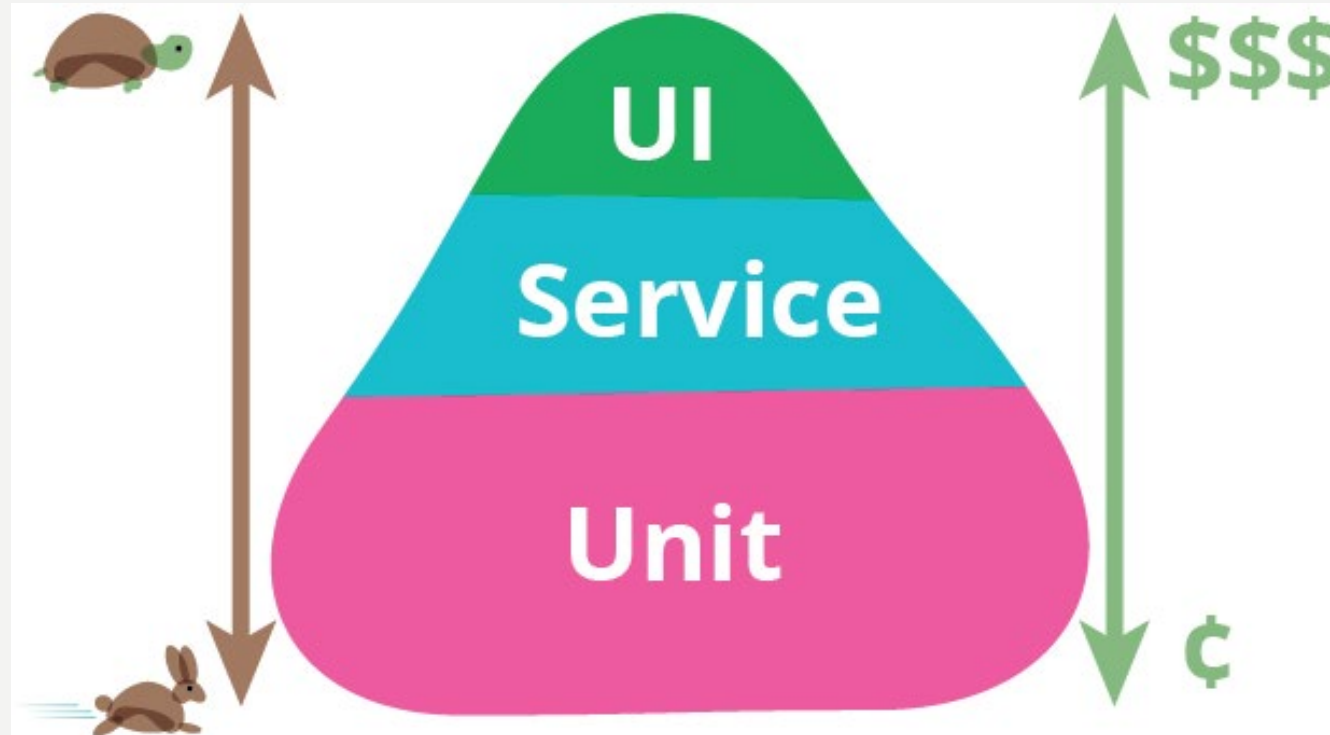
- **Test types**

- Functional tests (black-box)
 - Use cases /stories
 - Functional unit tests(!)
- Non-functional tests
 - “ilities”
- White-box
 - Code coverage, structural coverage
- Change-related tests
 - Confirmation testing (bug fixing)
 - Regression testing (unintended side-effects)

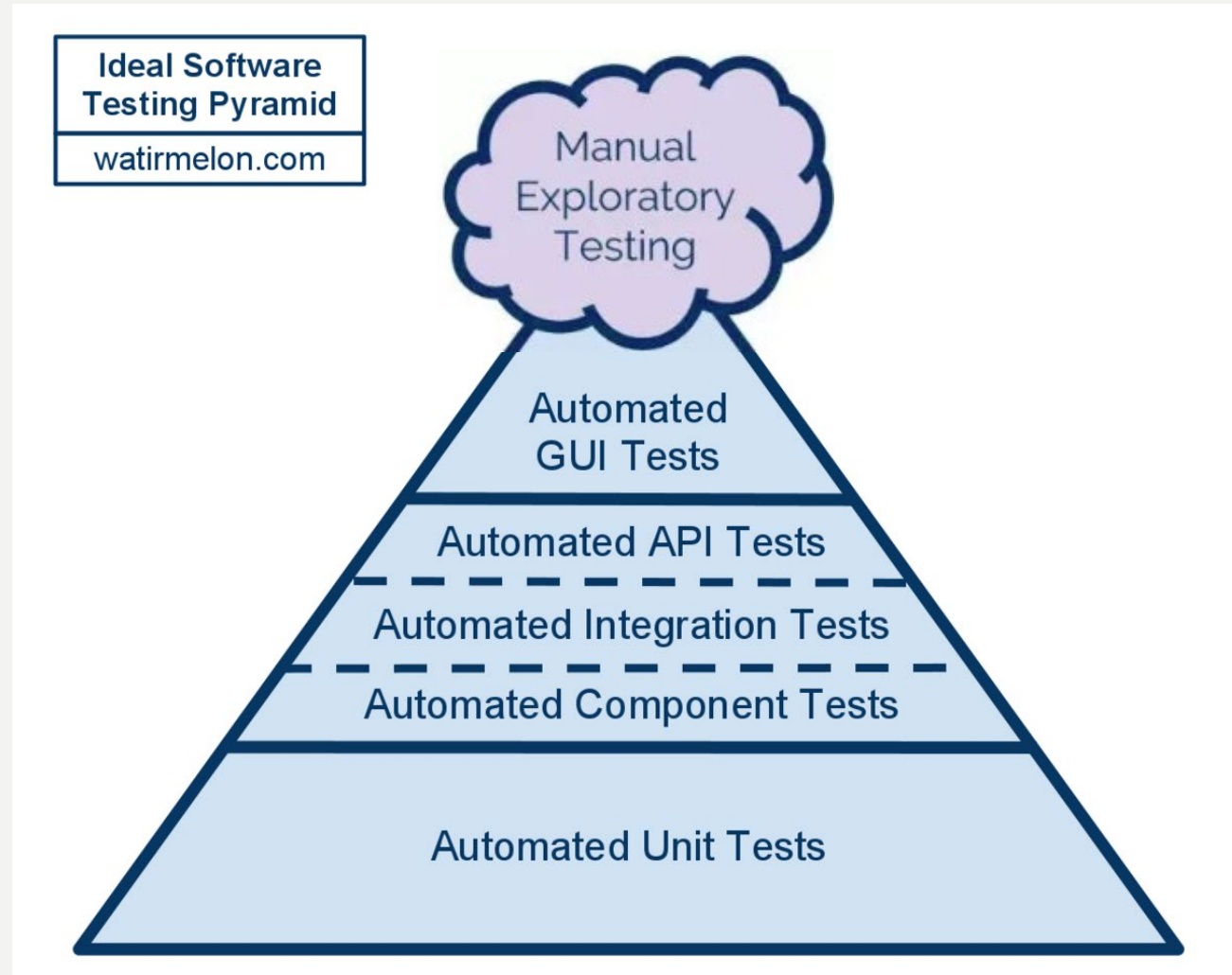
AGILE TESTING QUADRANTS



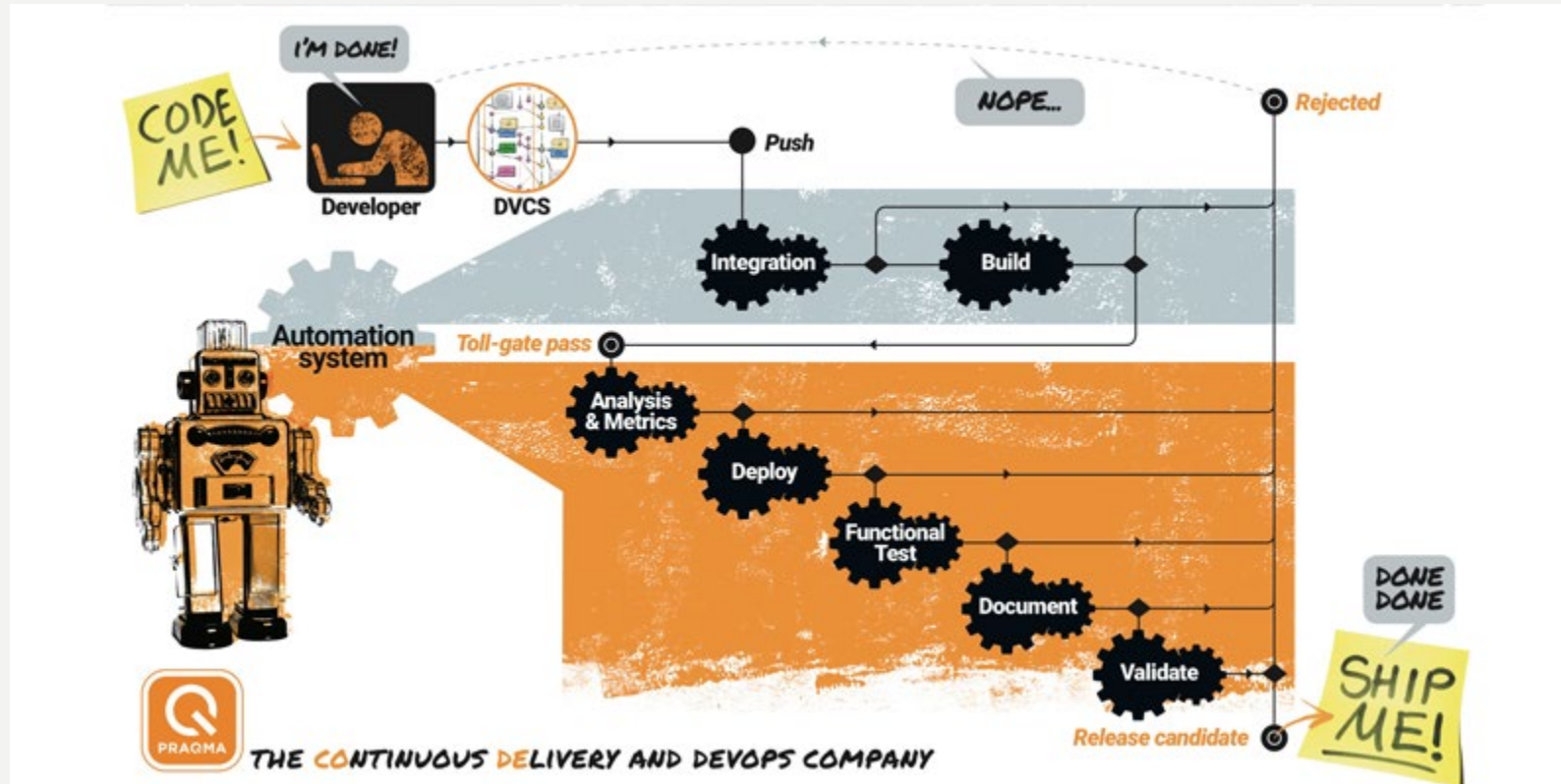
TEST PYRAMID



TEST PYRAMID



RELEASE PIPELINE

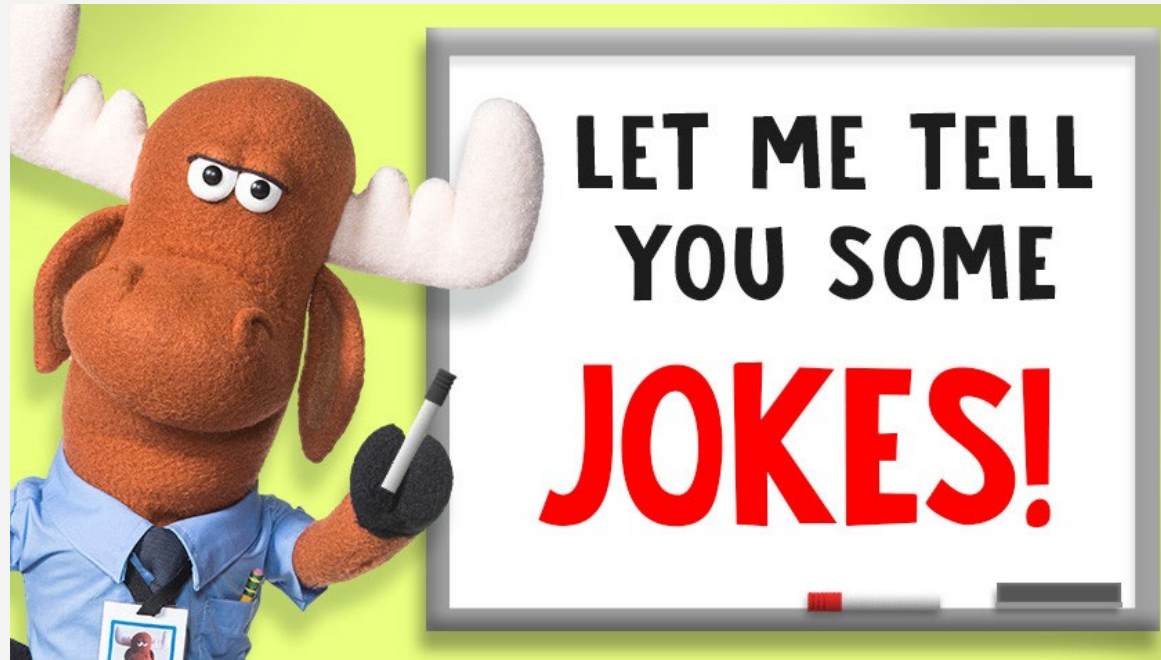


FUNCTIONAL UNIT TEST

```
@Test
public void testChuckNorris() {
    when().
        get("https://api.chucknorris.io/jokes/random").then().
            statusCode(200) ;
}
```


TEST REST SERVICE

- REST Assured Java DSL for easy testing of REST services: <http://rest-assured.io/>

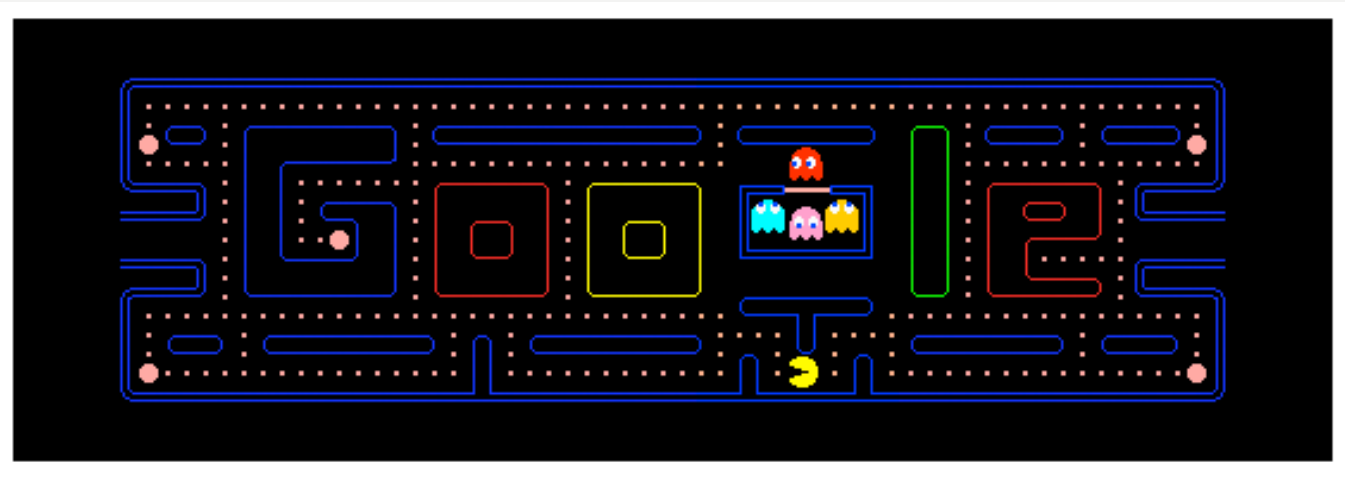


EXERCISE 1 - JOKEFETCHER

- Create a new method in `JokeFetcher` class that uses another Joke Rest API
- Write relevant unit tests
- Write relevant integration tests
- Constraints/issues
 - Unit tests that deal with dates, must use Java 8 Date/Time API
 - Hamcrest matchers should be considered for better readability
 - Implement a strategy for test of private methods
 - Implement a strategy for test to handle time difference of tests at the two test levels

AUTOMATED SYSTEM TESTING

- Selenium automates browsers. That's it! What you do with that power is entirely up to you.
<https://www.seleniumhq.org/>
- Example: Pacman automated



PRESENTATION LAYER TESTING

- We can test
 - the content of web pages to any level of detail (even spelling)
 - the application structure or navigation (following links to their expected destination, for example),
 - the ability to verify user stories with acceptance tests*
 - the site works with required browsers and operating systems.

SETTING UP A SELENIUM-WEBDRIVER PROJECT

- The easiest way to set up a Selenium 2.0 Java project is to use Maven

```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>3.141.59</version>
</dependency>
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>htmlunit-driver</artifactId>
  <version>2.29.0</version>
</dependency>
```

SELENIUM BROWSER SUPPORT

Name of driver	Available on which OS?	Class to instantiate
HtmlUnit Driver	All	org.openqa.selenium.htmlunit.HtmlUnitDriver
Firefox Driver	All	org.openqa.selenium.firefox.FirefoxDriver
Internet Explorer Driver	Windows	org.openqa.selenium.ie.InternetExplorerDriver
Chrome Driver	All	org.openqa.selenium.chrome.ChromeDriver

- HtmlUnit Driver is fast, but not graphical

```
WebDriver driver = new HtmlUnitDriver();
```

- Other drivers render content to a screen (can be used to check information such as the position of an element on a page)

- Firefox

```
WebDriver driver = new FirefoxDriver();
```

- Chrome

```
WebDriver driver = new ChromeDriver();
```

- you'll need a chromedriver binary,

GETTING STARTED WITH SELENIUM

- Goto “Introducing the Selenium-WebDriver API by Example” at https://www.seleniumhq.org/docs/03_webdriver.jsp
- Create a Maven project and copy the sample code from the grey box (Selenium2Example) into your project. You may choose Chrome or IE instead, if you want.
- Run code and see what happens

SELENIUM AND JUNIT

- Manually inspecting `System.out.println()` in an output window doesn't seem efficient
- Change the Google cheese search into a JUnit test

INTERACTING WITH THE PAGE

- Locating UI element (called WebElement):

Having this HTML code:

```
<input type="text" name="passwd" id="passwd-id" />
```

you could find it using any of:

```
WebElement element;  
element = driver.findElement(By.id("passwd-id"));  
element = driver.findElement(By.name("passwd"));  
element = driver.findElement(By.xpath("//input[@id='passwd-id']"));
```

PAGE INTERACTION EXAMPLE

- Let's see a local jsp example (BuyCarTest.java



STUDY POINT EXERCISE

- Create a Web user interface to Marios Pizzabar and relevant automated system tests.
- Push your solution to github, including an image with test results in the root and follow the review hand-in process.