Aalto University
School of Science

Automated unit & integration testing - Behavior Driven Testing frameworks: Spock

CS-E4960 Software Testing and Quality Assurance

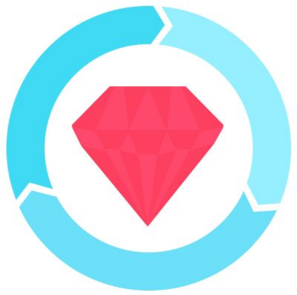
16.10.2018

Antti Ahonen

Department of Computer Science

Contents

Automated Low-level Testing with BDD-frameworks



- Continue from last week: Test data creating
- Motivation for going BDD-style
- **BDD Gherkin:** Spock
- Continue with general concepts using BDD-frameworks:
    - Isolation
    - Testability
    - Readability
    - Maintainability

How to build test objects for low level testing?

- Fixtures
- Named constructor parameters
- Factory-pattern: https://factoryboy.readthedocs.io/en/latest/
- **Builder-pattern**

- Always build the minimal test data for test method under run
- **Avoid SQL-scripts for integration test data seed!**

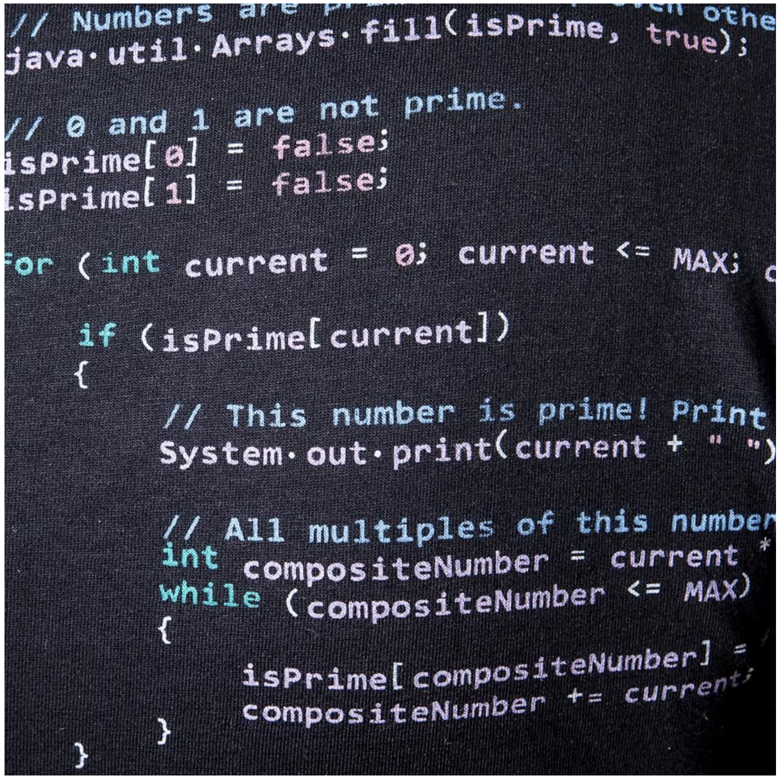
Integration testing - JUnit: Builder-pattern vs Sql scripts example

- Bringing visibility of context to tests
  - Removing magic assertions against context created outside of tests
- **“Visualizing”** the test data creating and **associations** with graph-structured builders

source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/main/java/fi/aalto/testingandqa/review/ReviewService.java

test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/tree/master/src/test/java/fi/aalto/testingandqa/review/reviewservice [AddReactionITest.java, AddReactionSqlITest.java]

commented test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/tree/master/src/test/java/fi/aalto/testingandqa/review/reviewservice [CommentedAddReactionITest.java, CommentedAddReactionSqlITest.java]



```
// Numbers are                       othe
java.util.Arrays.fill(isPrime, true);

// 0 and 1 are not prime.
isPrime[0] = false;
isPrime[1] = false;

or (int current = 0; current <= MAX;

    if (isPrime[current])
    {
        // This number is prime! Print
        System.out.print(current + " ")

        // All multiples of this number
        int compositeNumber = current *
        while (compositeNumber <= MAX)
        {
            isPrime[compositeNumber] =
            compositeNumber += current;
        }
    }
```

Problems with traditional xUnit testing?

Starting point

- Majority of developers find unit tests helpful in producing higher quality code [26]
- Majority of developers find unit tests helpful in understanding other people’s code [26]

BUT…

Problems

- Developers are mainly trying to find realistic scenarios on what to test [5]
- Developers finding isolating of unit under test hard [5]
- Only half of the survey respondents enjoy writing unit tests [5, 6]
- Maintaining unit tests was found hard [5, 6]
- For 60.4% of developers, understanding unit tests is at least moderately difficult [46]
- Developers find updated documentation and comments in test cases useful, but writing comments to unit tests is rarely or never done [46]

Behavior Driven Testing frameworks for (JVM) Low-level testing

Gherkin style: Spock

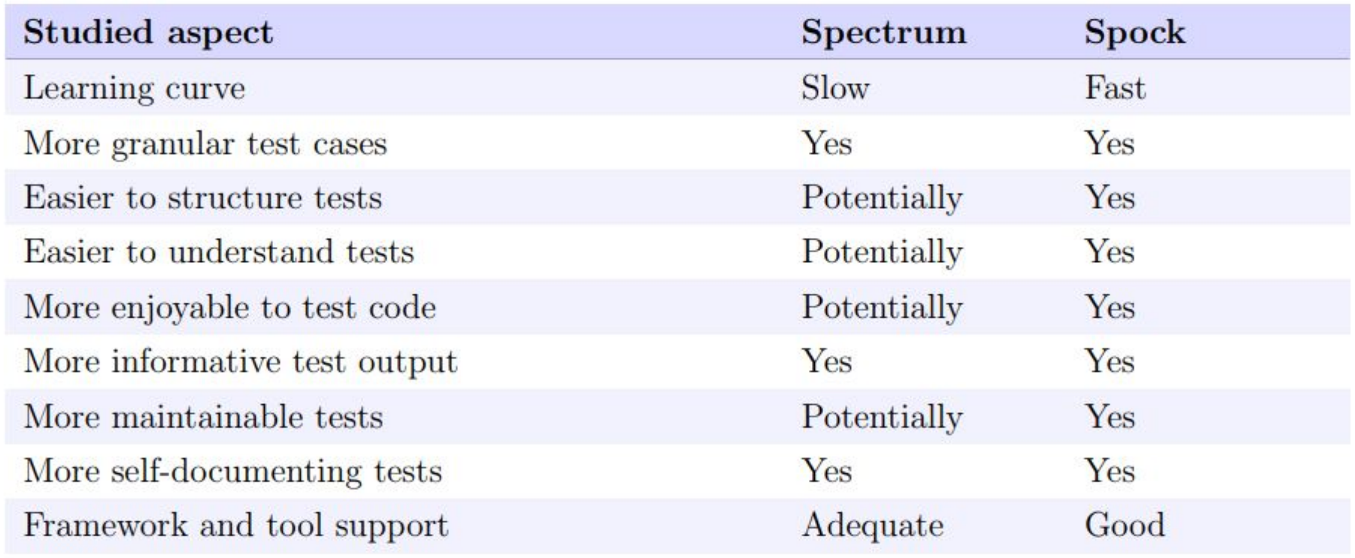
- Specification framework
    - Describe the desired behavior of system under test
- Produces Spec files
    - with feature methods
        - constructed with **Gherkin** blocks
            - **Given, When, Then**
- Dynamic features from Groovy-language
    - Data-Driven Testing
    - Mocks & stubs
    - Debug prints
    - Dynamic builders etc..

Spec style: Spectrum

- Specification framework
- Produces Spec files
    - with example groups
        - with runnable code examples
- Expectations instead of assertions
- Lifecycle hooks instead of fixtures
- Supports nested contexts with example groups

Expectations	$\Longrightarrow$	Assertions
Code Example	$\Longrightarrow$	Test Method
Example Group	$\Longrightarrow$	Test Case
Spec File	$\Longrightarrow$	Test Suite
Lifecycle Hook	$\Longrightarrow$	Test Fixture

Behavior Driven Testing frameworks for common unit testing problems?

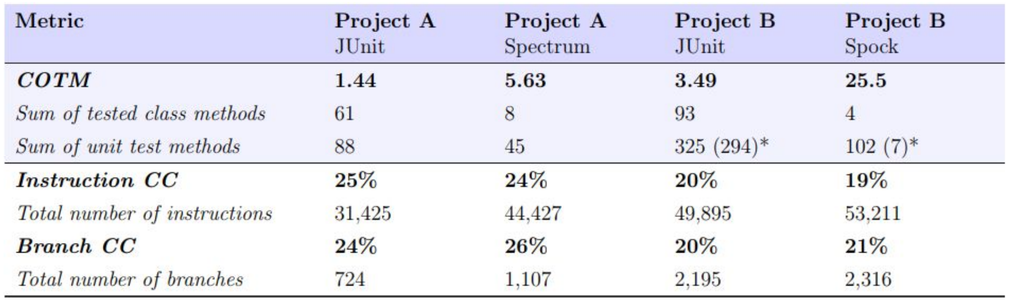


Studied aspect	Spectrum	Spock
Learning curve	Slow	Fast
More granular test cases	Yes	Yes
Easier to structure tests	Potentially	Yes
Easier to understand tests	Potentially	Yes
More enjoyable to test code	Potentially	Yes
More informative test output	Yes	Yes
More maintainable tests	Potentially	Yes
More self-documenting tests	Yes	Yes
Framework and tool support	Adequate	Good

A. Ahonen, “Unit and Integration Testing of Java:
JVM Behavior-Driven Development testing frameworks vs. JUnit” 2017.

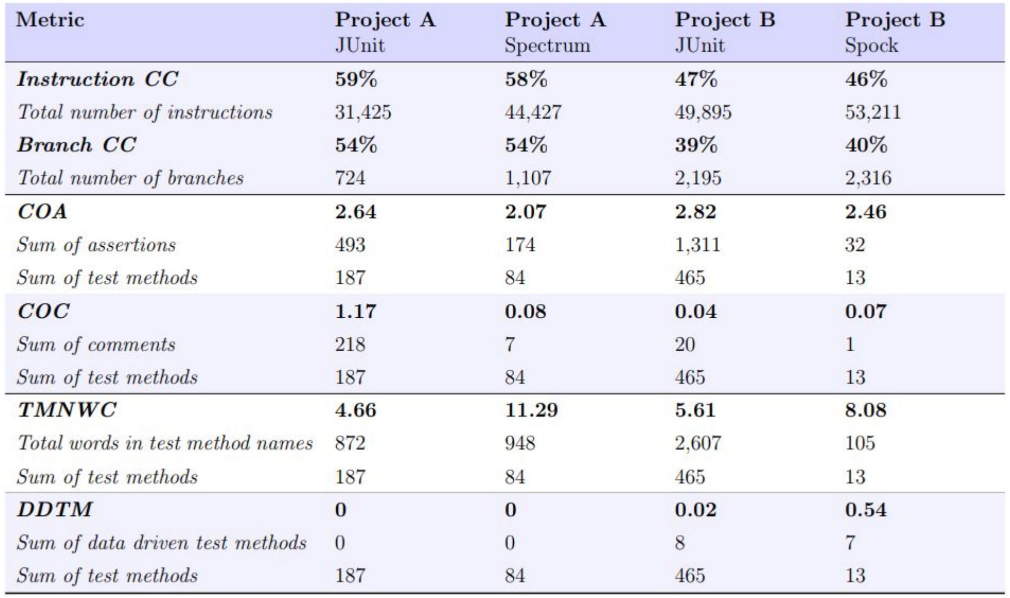
Behavior Driven Testing framework vs JUnit test code metrics

Pure unit test metrics



Metric	Project A JUnit	Project A Spectrum	Project B JUnit	Project B Spock
COTM	**1.44**	**5.63**	**3.49**	**25.5**
Sum of tested class methods	61	8	93	4
Sum of unit test methods	88	45	325 (294)*	102 (7)*
Instruction CC	**25%**	**24%**	**20%**	**19%**
Total number of instructions	31,425	44,427	49,895	53,211
Branch CC	**24%**	**26%**	**20%**	**21%**
Total number of branches	724	1,107	2,195	2,316

Both unit & integration test metrics



Metric	Project A JUnit	Project A Spectrum	Project B JUnit	Project B Spock
Instruction CC	**59%**	**58%**	**47%**	**46%**
Total number of instructions	31,425	44,427	49,895	53,211
Branch CC	**54%**	**54%**	**39%**	**40%**
Total number of branches	724	1,107	2,195	2,316
COA	**2.64**	**2.07**	**2.82**	**2.46**
Sum of assertions	493	174	1,311	32
Sum of test methods	187	84	465	13
COC	**1.17**	**0.08**	**0.04**	**0.07**
Sum of comments	218	7	20	1
Sum of test methods	187	84	465	13
TMNWC	**4.66**	**11.29**	**5.61**	**8.08**
Total words in test method names	872	948	2,607	105
Sum of test methods	187	84	465	13
DDTM	**0**	**0**	**0.02**	**0.54**
Sum of data driven test methods	0	0	8	7
Sum of test methods	187	84	465	13

A. Ahonen, “Unit and Integration Testing of Java:
JVM Behavior-Driven Development testing frameworks vs. JUnit” 2017.

Spock: Gherkin

- Like traditional xUnit-testing framework, but enhanced for readability and maintainability
- **Gherkin**-blocks
  - Separating different parts of test (feature method)
  - **Given:** For test context creating
  - **When:** For tested action
  - **Then:** For assertions against action results
- and a couple of extra blocks
  - **And:** Can be applied after any block to continue using the previous block
  - **Expect:**
    - For doing assertions on the initial context before action
    - or combining **when** and **then** in into one in concise action + assertion
  - **Where:** Data-Driven testing
- Check primer for more: http://spockframework.org/spock/docs/1.2/spock_primer.html

Aalto University
School of Science

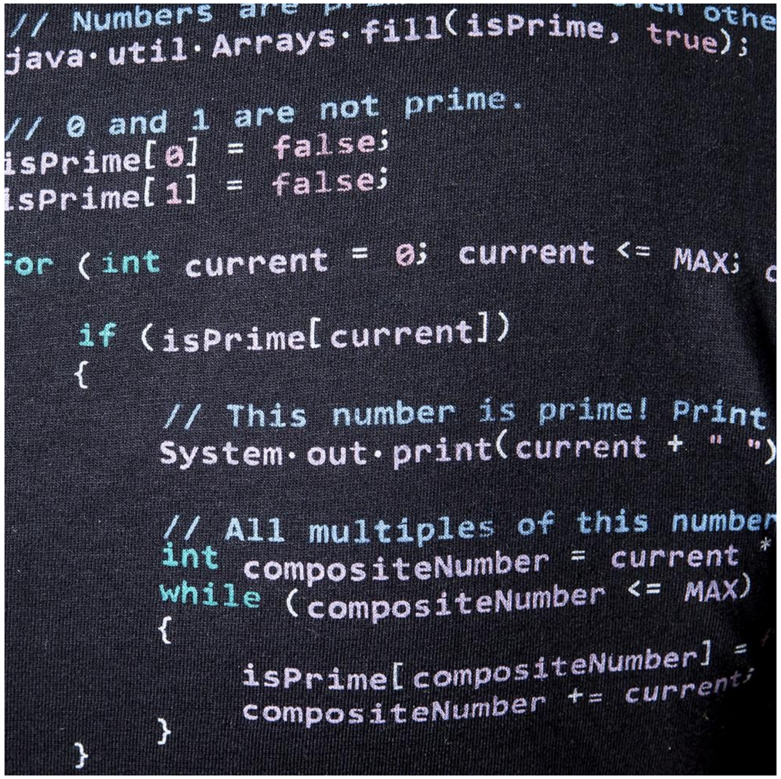
Spock - Simple example with Gherkin blocks

- Gherkin in action
- Exception handling

source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/main/java/fi/aalto/testingandqa/review/ReviewService.java

test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/groovy/fi/aalto/testingandqa/reviewservice/AddCommentISpec.groovy

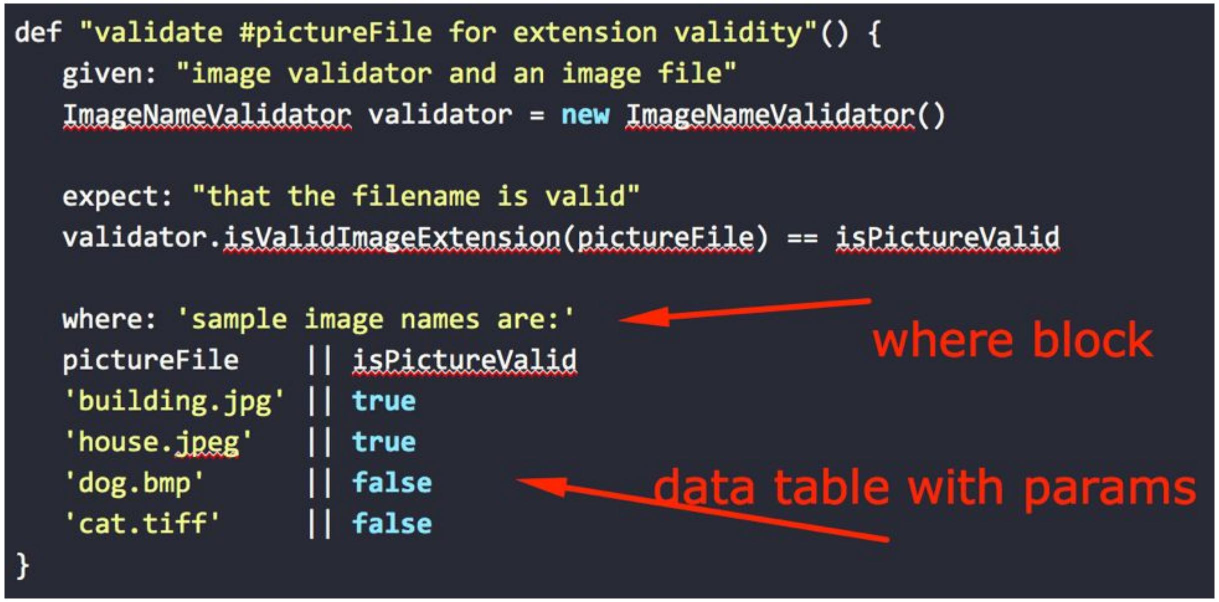
commented test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/groovy/fi/aalto/testingandqa/reviewservice/CommentedAddCommentISpec.groovy



```
// Numbers are                                  othe
java.util.Arrays.fill(isPrime, true);

// 0 and 1 are not prime.
isPrime[0] = false;
isPrime[1] = false;

for (int current = 0; current <= MAX; c

    if (isPrime[current])
    {
        // This number is prime! Print
        System.out.print(current + " ")

        // All multiples of this number
        int compositeNumber = current *
        while (compositeNumber <= MAX)
        {
            isPrime[compositeNumber] =
            compositeNumber += current;
        }
    }
```

Spock: Data-Driven Testing

- Tabular format readable domain specific language (DSL) [45]
- Remove repetition from code
- Test different parameter variations easily



```
def "validate #pictureFile for extension validity"() {
   given: "image validator and an image file"
   ImageNameValidator validator = new ImageNameValidator()

   expect: "that the filename is valid"
   validator.isValidImageExtension(pictureFile) == isPictureValid

   where: 'sample image names are:'
   pictureFile     || isPictureValid
   'building.jpg'  || true
   'house.jpeg'    || true
   'dog.bmp'       || false
   'cat.tiff'      || false
}
```

Aalto University
School of Science

Code sample originally from: https://www.blazemeter.com/blog/api-test-parameterization-with-spock-learn-how

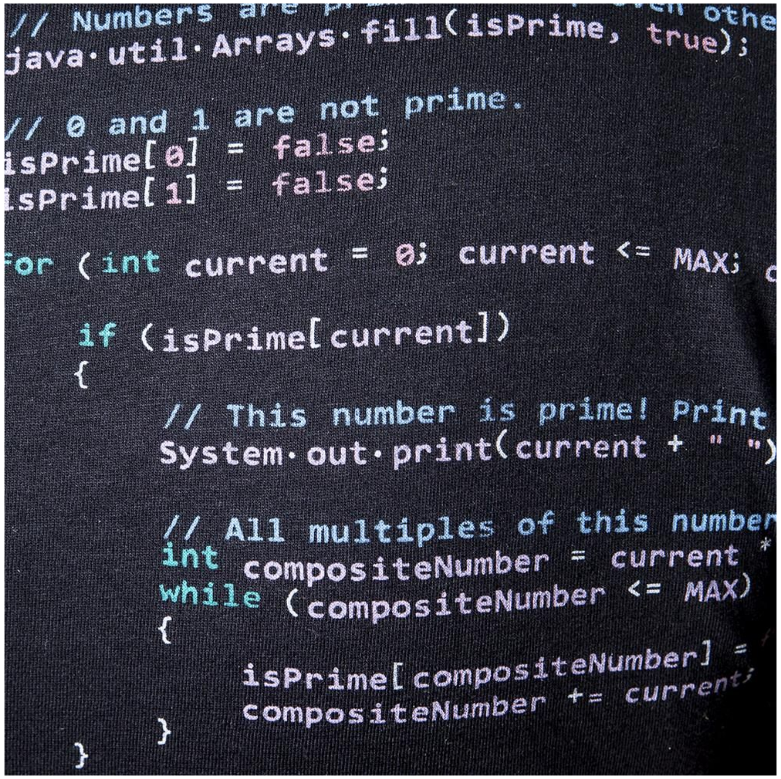
Spock - Data Driven Testing examples

- Removing repetition
- Testing parameter variations
- Injecting test output information through parametrization
- More from here: http://spockframework.org/spock/docs/1.2/data_driven_testing.html

source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/main/java/fi/aalto/testingandqa/review/ReviewService.java and
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/main/java/fi/aalto/testingandqa/algorithm/CurlyBracesChecker.java

test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/groovy/fi/aalto/testingandqa/reviewservice/AddCommentDataDrivenISpec.groovy and
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/groovy/fi/aalto/testingandqa/algorithm/CurlyBracesCheckerSpec.groovy

commented test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/groovy/fi/aalto/testingandqa/reviewservice/CommentedAddCommentDataDrivenISpec.groovy and
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/groovy/fi/aalto/testingandqa/algorithm/CommentedCurlyBracesCheckerSpec.groovy



```
// Numbers are                       othe
java.util.Arrays.fill(isPrime, true);

// 0 and 1 are not prime.
isPrime[0] = false;
isPrime[1] = false;

for (int current = 0; current <= MAX; c

    if (isPrime[current])
    {
        // This number is prime! Print
        System.out.print(current + " ")

        // All multiples of this number
        int compositeNumber = current *
        while (compositeNumber <= MAX)
        {
            isPrime[compositeNumber] =
            compositeNumber += current;
        }
    }
```

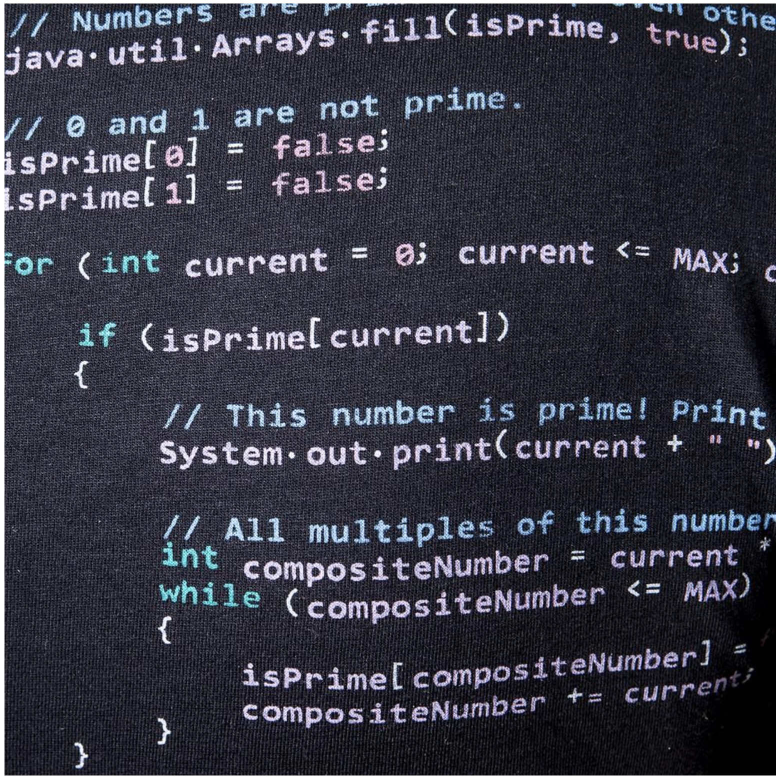
Spock - Mocking and stubbing example

- Mocking
- Stubbing
- Verifying mock object interactions
- More from here: http://spockframework.org/spock/docs/1.2/interaction_based_testing.html

source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/main/java/fi/aalto/testingandqa/review/ReviewService.java

test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/groovy/fi/aalto/testingandqa/reviewservice/AddCommentSpec.groovy

commented test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/groovy/fi/aalto/testingandqa/reviewservice/CommentedAddCommentSpec.groovy



```
// Numbers are                othe
java.util.Arrays.fill(isPrime, true);

// 0 and 1 are not prime.
isPrime[0] = false;
isPrime[1] = false;

for (int current = 0; current <= MAX; c

    if (isPrime[current])
    {
        // This number is prime! Print
        System.out.print(current + " ")

        // All multiples of this number
        int compositeNumber = current *
        while (compositeNumber <= MAX)
        {
            isPrime[compositeNumber] =
            compositeNumber += current;
        }
    }
```

How to make testable code?

"Untangling the spaghetti"

- Layered architecture
- Decoupling
- Single responsibility

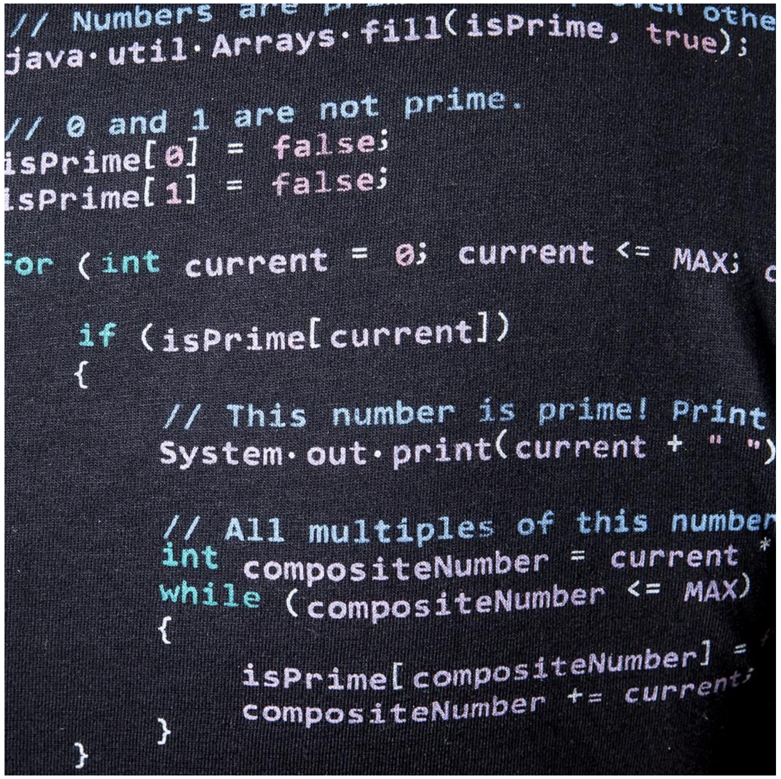
Spock: Unit & Integration testing - Refactoring production code for testability

- Unit testability
  - Easier test context creating
- Unit & Integration testing with external dependencies
  - Isolation
- Spock debug prints

source-codes:
https://github.com/anttiahonen/junit-spock-testing-examples/tree/master/src/main/java/fi/aalto/testingandqa/chucknorris

test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/tree/master/src/test/groovy/fi/aalto/testingandqa/chucknorris
[ChuckJokeTransformerSpec, ChuckNorrisControllerWithMocksRestSpec, ChuckNorrisControllerWithSpyRestSpec]

commented test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/tree/master/src/test/groovy/fi/aalto/testingandqa/chucknorris
[CommentedChuckJokeTransformerSpec, CommentedChuckNorrisControllerWithMocksRestSpec, CommentedChuckNorrisControllerWithSpyRestSpec]



```
// Numbers are                              othe
java.util.Arrays.fill(isPrime, true);

// 0 and 1 are not prime.
isPrime[0] = false;
isPrime[1] = false;

or (int current = 0; current <= MAX;

    if (isPrime[current])
    {
        // This number is prime! Print
        System.out.print(current + " ")

        // All multiples of this number
        int compositeNumber = current *
        while (compositeNumber <= MAX)
        {
            isPrime[compositeNumber] =
            compositeNumber += current;
        }
    }
```

Aalto University
School of Science

References

[5] E. Daka and G. Fraser, “A survey on unit testing practices and problems,” in Software Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium on, pp. 201–211, IEEE, 2014.

[6] P. Runeson, “A survey of unit testing practices,” IEEE software, vol. 23, no. 4, pp. 22–29, 2006.

[26] L. Williams, G. Kudrjavets, and N. Nagappan, “On the effectiveness of unit test automation at microsoft.,” in ISSRE, pp. 81–89, 2009.

[45] P. Niederwieser, “Spock framework reference documentation.” http://spockframework.org/spock/docs/1.1-rc-3/all_in_one.html , 2017. [Online; accessed 28-March-2017].

[46] B. Li, C. Vendome, M. Linares-Vásquez, D. Poshyvanyk, and N. A. Kraft, “Automatically documenting unit test cases,” in Software Testing, Verification and Validation (ICST), 2016 IEEE International Conference on, pp. 341–352, IEEE, 2016.