Aalto University
School of Science

Automated unit & integration testing

CS-E4960 Software Testing and Quality Assurance

9.10.2018

Antti Ahonen
Department of Computer Science

Contents

Automated Low-level Testing

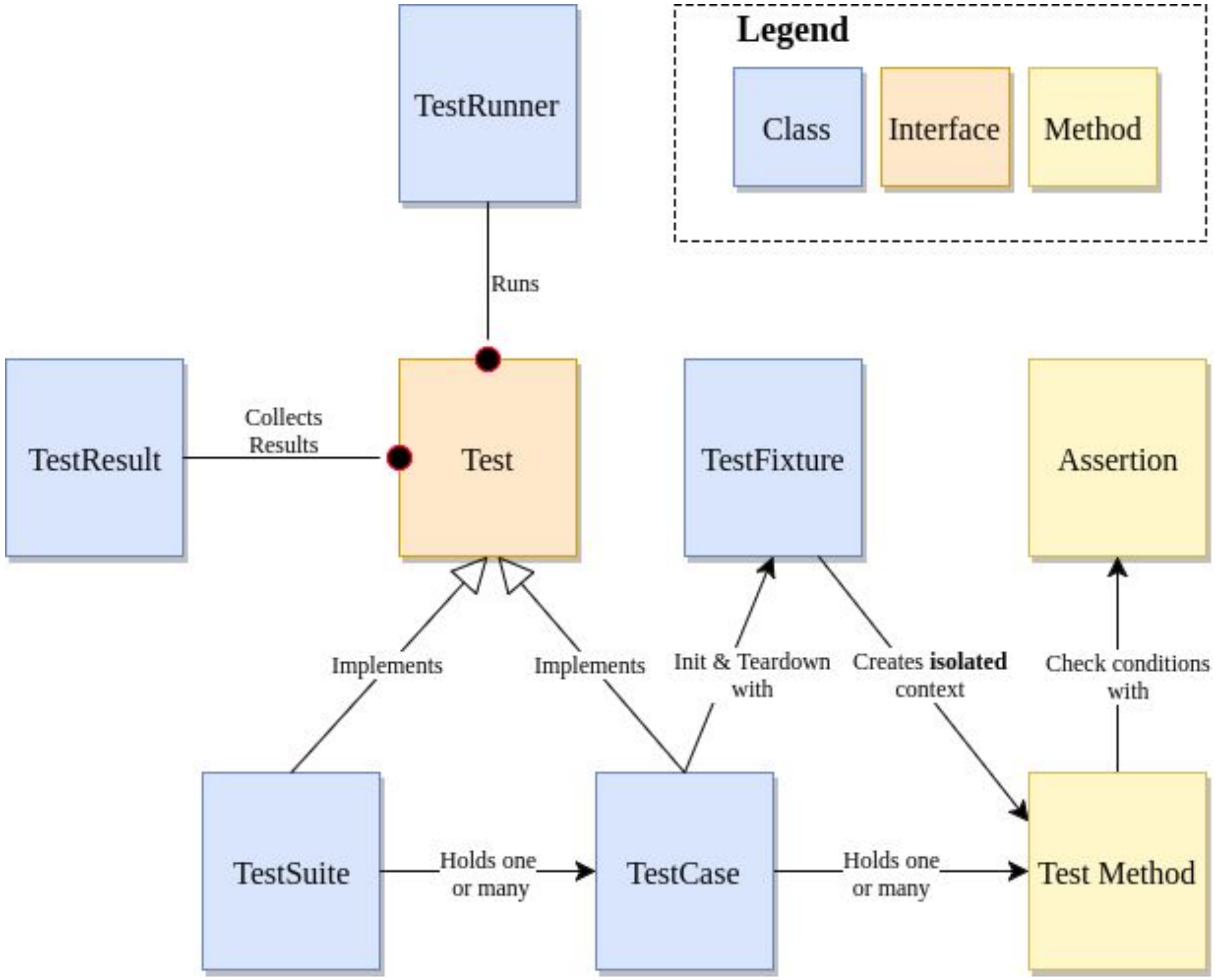
- Unit testing
- Integration testing
- Integration vs Unit testing
- Test data creating
- Isolation
- Testability
- Readability
- Maintainability

Unit test

- Tests individual unit or collection of these units working as one **[6, 22]**

- A good unit test is **[21]**
  - maintainable
  - readable
  - isolated
  - single concern
  - minimal amount of repetition

Automated test structure with JUnit



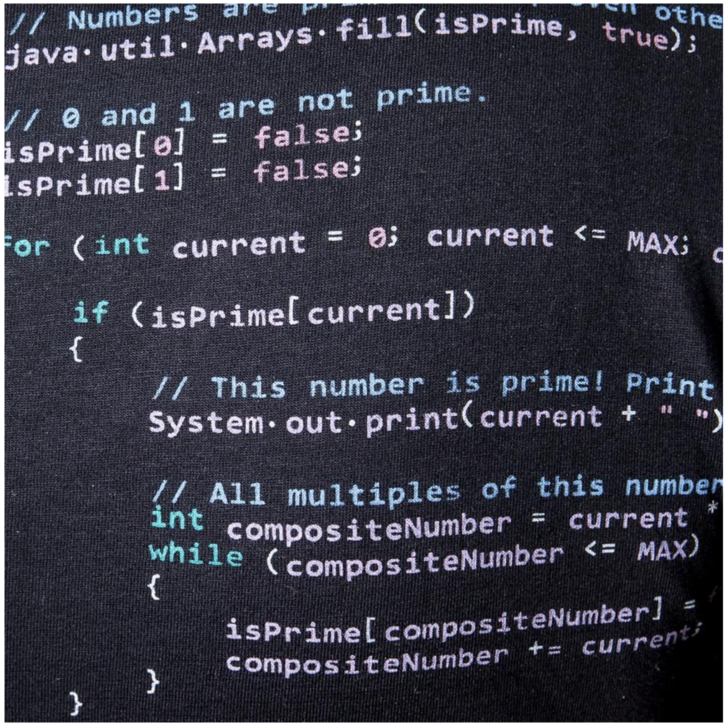
Unit testing - JUnit: simple example

- Adding tests to existing algorithm
  - Gathering test coverage
    - Testing exceptions

source-code:
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/java/fi/aalto/testingandqa/algorithm/CurlyBracesCheckerTest.java

commented test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/java/fi/aalto/testingandqa/algorithm/CommentedCurlyBracesCheckerTest.java

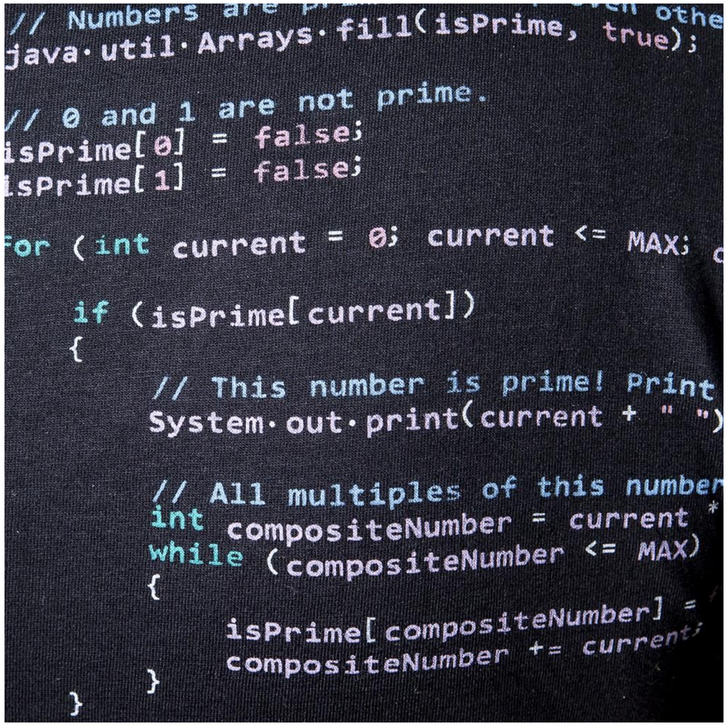


```
// Numbers are pr                othe
java.util.Arrays.fill(isPrime, true);

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

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

    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;
        }
    }
```

Coverage as test code quality meter?

Unit testing - Pytest: refactoring example

- Maintainability:
  - Removing repetition
    - Using fixture methods
    - Using helper methods
- Readability
  - Separating concerns
  - Naming things
  - Get rid of magic constants
  - Creating your own test DSL



```
// Numbers are pr[illegible] 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;
        }
    }
```

source-code:
https://github.com/anttiahonen/python-unit-testing-example/tree/master/example

test source-code:
https://github.com/anttiahonen/python-unit-testing-example/tree/master/example/tests (files without the word _commented_)

commented test source-code:
https://github.com/anttiahonen/python-unit-testing-example/tree/master/example/tests (files with the word _commented_)

Integration testing

- Testing activity which involves multiple components [21, 22]
- Testing a unit of work with real dependencies in place [21]:
  - database
  - networking etc...
- Usually not as fast as unit test
  - **Context loading** is slow, for example dependency injection containers such as *Spring Framework*

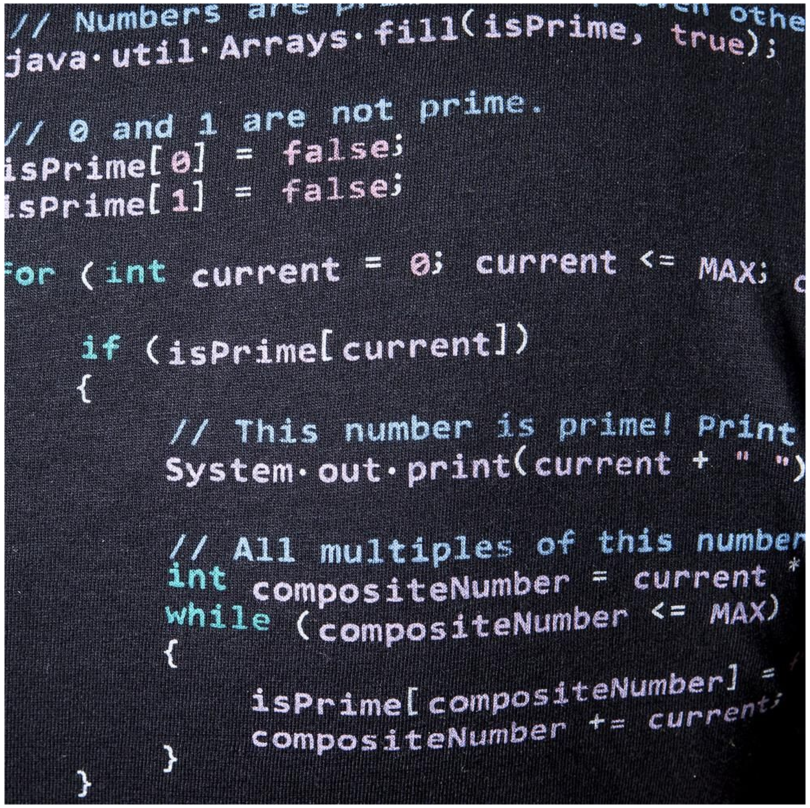
Integration testing - JUnit: SpringBoot example

- Context loading
- Testing with memory db

source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/tree/master/src/main/java/fi/aalto/testingandqa/review
 (ReviewService.java is the top class under test)

test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/java/fi/aalto/testingandqa/review/reviewservice/AddCommentITest.java
and:
https://github.com/anttiahonen/junit-spock-testing-examples/tree/master/src/main/java/fi/aalto/testingandqa/review

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

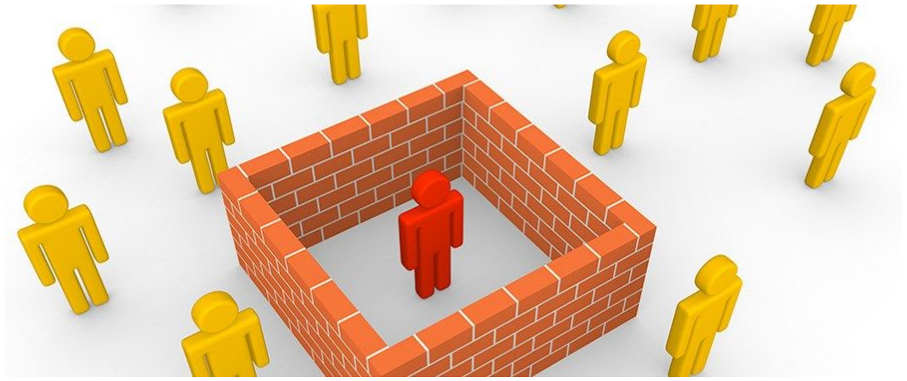


```
// Numbers are                          othe
java.util.Arrays.fill(isPrime, true);

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

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

    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;
        }
    }
```

Isolation [4]

- Isolation with
  - **Mocking**: substituting real objects with limited functionality provided by mocks
  - **Stubbing:** injecting outputs for mocked object behaviors

- Isolation provides
  - Determinism
  - Enables TDD/BDD



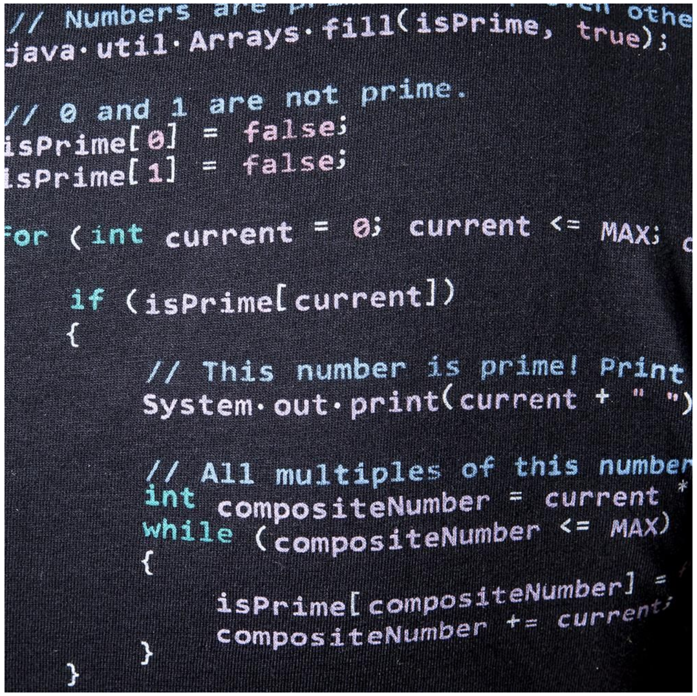
Unit vs. Integration testing - JUnit: mocking & stubbing example

- Mockito
  - Substituting real dependencies with mock objects
  - Stubbing values from mock object method calls
  - Verifying actions on mock objects

source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/tree/master/src/main/java/fi/aalto/testingandqa/review (ReviewService.java is the top class under test)

test source-code:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/java/fi/aalto/testingandqa/review/reviewservice/AddCommentTest.java
and:
https://github.com/anttiahonen/junit-spock-testing-examples/blob/master/src/test/java/fi/aalto/testingandqa/review/ReviewServiceBase.java

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



```
// Numbers are [illegible] 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 build test objects for low level testing?

- Fixtures
- Named constructor parameters
- **Factory-pattern**
- **Builder-pattern**

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

How to make testable code?

"Untangling the spaghetti"

- Layered architecture
- Decoupling
- Single responsibility

References

[4] D. Chelimsky, D. Astels, Z. Dennis, A. Hellesøy, B. Helmkamp, and D. North, The RSpec Book: Behaviour-driven Development with RSpec, Cucumber, and Friends. Pragmatic Bookshelf Series, Pragmatic Bookshelf, 2010.

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

[21] R. Osherove, The Art of Unit Testing, Second Edition. Manning Publications Company, 2013.

[22] J. A. Whittaker, “What is software testing? and why is it so hard?,” IEEE software, vol. 17, no. 1, pp. 70–79, 2000.