Noroff
School of technology
and digital media

Thursday - Week 2
JUnit

JUnit Framework

- This a framework for building unit tests
- We import it as part of the project
- We are then able to write “unit tests”
    - Units -> a measure of something
    - Tests -> a test of a particular piece of code

What does it do for us?

- We define ‘test cases’
- These are pairs of values:

  Expected vs. **Actual**
- We write tests to match the specifications of the software

Test classes

- For every class we write we can write a test class
- We can write multiple classes per method
  - Each must have a unique test name (method in the test class)
  - Each must be marked with the @Test Annotation
  - Each must make an assertion

Assertions

- Can be true / false
- Can throw exceptions
- Write one test per expected behaviour

Annotations

- Start with an @ sign
- They tell the compiler how to work with the method that follows
- We use @Test for build unit tests,
- Later, when using Spring we will use different ones

Test cases and memory

- Each time we run a test method we must initialise ANY variable or objects that are needed

- We can set up a once off global - @BeforeAll
- We can run a method before each - @BeforeEach
- Similarly we can run methods after All/Each

Running Unit Tests

- After we run the test we get a report of pass/fails

- We can isolate malfunctioning code without the need for System.out.println()
- We can run tests in isolation

Testing with dummy objects

- It is possible to build Objects to test more complex functionality
- We don’t cover this, but it is possible

- For those interested look into “Mockito”

Further Reading

- Check out the JUnit5 User Guide:
  - https://junit.org/junit5/docs/current/user-guide/

Noroff
School of technology
and digital media

Thursday - Week 2

Automated Testing in IntelliJ

The *quick* how-to (part 1)

- Start by creating a “tests” folder at the same level as “src”
- Mark it as a test folder
  Project Structure -> Modules -> Sources
  or Right-Click the folder
- Start with a normal class that you want to test

The *quick* how-to (part 2)

- Highlight the class name
- Press: Ctrl + Shift + T (Shift + ⌘ + T on **Mac**)
- Select create a new test
- Write your test*
- You can run them by right clicking individual test classes or the whole tests folder

Noroff
School of technology and digital media

Thursday - Week 2

Test Driven Development

Test Driven Development

- We can develop software by building tests first and writing code second
- This will seem VERY tedious
- It saves time in the long run

Always passing

- By building software in this “reverse” fashion we can very quickly revert to code that “works”
- This is particularly helpful on larger projects
    - We can also always track the progress of work
    - It’s very obvious where we need to start working

Noroff
School of technology
and digital media

Thursday - Week 2

Tasks

Task (No hand-in):

- Watch Uncle Bob’s Three Laws of TDD
- Only watch to approx. 45min
- https://www.youtube.com/watch?v=qkblc5WRn-U

Task 15: PetRock example

- This is built using JUnit4 in an older IntelliJ
- Convert it to JUnit5 - Some things will need to be updated
- Part 1 (12min):
  - https://www.youtube.com/watch?v=Bld3644bIAo
- Part 2 (17min) - Cuts off before the ends:
  - https://www.youtube.com/watch?v=xHk9yGZ1z3k

Task 15: PetRock example

- Hand in your version of the pet rock code with 100% passing tests for all the tests described in the videos

- You may stop BEFORE adding the global timeout rule