Unit Testing

Unit Tests

- A Unit Test is just a Java method that calls a method and asserts the result
- Unit Tests test small units of code in isolation, typically a method
- If the result is what is expected then the test passes
- If the result is not what is expected then the test fail

JUnit

- JUnit is a framework for unit tests in Java
- JUnit is integrated in IntelliJ IDEA and makes it easy to create unit tests
- JUnit uses a @Test annotation to make a method a test method
- JUnit also has classes that make it easy to assert the result of the test

TDD - Test Driven Development

- TDD is about creating the test before the implementation
- The Test drives development
- By always creating the test first the test will not be forgotten
- Also, it forces the developer to think differently, by first thinking of how a new feature should be used and called and then think about the implementation

Red-Green-Refactor

- Red - First create a failing test
- Green - Then make the test pass by implementing the code
- Refactor - Then refactor until you are satisfied with the code (still with a passing test)

Why use TDD?

- We want to develop code fast and efficiently
- To work fast we need clean code
- To keep the code clean we need to refactor
- To refactor we need conficence (that we don’t break anything)
- Tests give us that confidence (we can check that we don’t break anything)

Demo 1 - Unit Testing

- Adding Unit Tests in the Spring Boot Test class
- Use Dependency Injection to get access to the object to be tested
- Assert that the expected value is what is returned

Exercise 1 - Unit Testing

- Use the UnitTestingDemo project
- Create tests to make sure that the FizzBuzzService is returning the expected result when being sent different numbers as input
- Create one test for each type of expected result, that is one test for when the number is expected to be returned, one test for fizz, one test for buzz and one test for fizzbuzz
- Create several Assertions in each test to assert a few different values will have the same result

Exercise 2 - Unit Testing

- If you have time, create a new class in the Spring Boot project. Add a method that returns something based on the input. Annotate the class with @Service
- Autowire an object of the class into the Spring Boot test class
- Create a test and assert that the method really returns the expected value based on the input