Project Lombok *date of composing: 02/11/'19*

```
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
   <groupId>org.projectlombok</groupId>
   <artifactId>lombok</artifactId>
   <version>1.18.10</version>
   <scope>provided</scope>
</dependency>
```

Goal:

- reducing boilerplate code

Annotations:

- @Getter and @Setter
- @NonNull
- @ToString
- @EqualsAndHashCode
- @Data
- @Cleanup
- @Synchronized
- @SneakyThrows
- @AllArgsConstructor
- @NoArgsConstructor

Pay attention to:

- Constructors which are not using all variables need to be written yourself
- Trouble detecting a superclass' constructor
- Lombok uses the standard JavaBean namingconventions
  - Watch out how you name your variables!!!

Documentation:

https://objectcomputing.com/resources/publications/sett/january-2010-reducing-boilerplate-code-with-project-lombok

Mockito

add the following to your properties inside your pom.xml:

```
<mockito2.version>2.19.0</mockito2.version>
<powermock.version>2.0.0-beta.5</powermock.version>
```

Add the following to your depencies:

```
<dependency>
    <groupId>org.powermock</groupId>
    <artifactId>powermock-core</artifactId>
    <version>${powermock.version}</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.powermock</groupId>
    <artifactId>powermock-module-junit4</artifactId>
    <version>${powermock.version}</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.powermock</groupId>
    <artifactId>powermock-api-mockito2</artifactId>
    <version>${powermock.version}</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>${mockito2.version}</version>
    <scope>test</scope>
</dependency>
```

Goal:

- Eradicating the need of self-written stubs and mocks

Annotations:

- @Mock
- @Spy
- @InjectedMock

Pay attention to:

- In order to use the annotations, set your testClass to `@RunWith(MockitoJUnitRunner.class)`
- Mockito is useable without Spring, but don't forget to import org.mockito.Mockito
- *when()* accepts a mock, and must be used with a following .then…-statement

Documentation: https://www.baeldung.com/mockito-series

assertj

```
<dependency>
    <groupId>org.assertj</groupId>
    <artifactId>assertj-core</artifactId>
    <!-- use 2.9.1 for Java 7 projects -->
    <version>3.11.1</version>
    <scope>test</scope>
</dependency>
```

Goal:

- allowing easier readable assert-statements

Import:

```
import static org.assertj.core.api.Assertions.assertThat;
```

Example:

String message = “test”;

assertThat(message).isEqualTo(stringWrapper.getValue());

Note:

Normally, we would write something like this:

```
Assert.assertEquals("StringWrapper did not return the expected value.", message,
stringWrapper.getValue());
```

How many times did you omit the error-message (which is legal to do), and how many times did you switch the expected and actual argument?

Do your fellow programmers a solid and try to strive towards Clean Code practice. Combined with the given-when-then comments (or Arrange-Act-Assert), someone who reads your tests weeks after you wrote them, will still be able to make something out of it.