# TCI – week 2 practical

These exercises should be executed individually. All of these are assignments are needed in the final project.

## Unit Tests

### Assignment – Basic mock exercises

Tests with mock objects: Do exercise 5.7.1 from the book.

### Assignment – Creating Tests based on requirements using TDD

### Look at the requirements below and translate them to unit tests one by one. Make sure you use the TDD rhythm as described in chapter 4.

The case is involving modeling of a Book with Chapters

|  |  |
| --- | --- |
| Requirement | Description |
| **A chapter is described by name and number** | * Name is a not null string * number is a string like ‘2.4’, ‘3’, ‘4.3’ etc. 2 levels of chapters is maximum. * If anything is wrong with the parameters, an IllegalArgumentException is thrown. |
| **Chapters can be used in a sorted collection** | * They must implement Comparable and override equals and hashcode. |
| **A Book is described by name and author** | * Both not null, and not empty. Otherwise IllegalArgumentException. |
| **You can add a Chapter to a book** | * Parameters include the name and number of a chapter after which a Chapter Object is created and added, * Otherwise IllegalArgumentException. |
| **You can get a Table Of Contents** | * A Set is returned which contains a clone of the Chapters of the Book, sorted by their natural ordening. |

### Assignment – Creating Tests based on requirements AND using Mock Objects

Do exercise 5.7.2 from the book.

*Reflection points:*

* Did you manage to stick to the TDD rhythm? If not, what happened? What do you need to be able to stay in the TDD rhythm next time?

I didn’t fully stick to TDD rythm. Because I need to know the overview of the program itself and how do they will work. By defining a specific requirement in the test case didn’t help me quite much. If I don’t have any view regarding on how does the program works then probably it’ll be hard for me to follow on afterwards. In order to keep stay in TDD rhythm I think would be good if we understand the program works, what are the requirements then we can start afterwards.

* Are the names of the test methods self-explanatory?

Yes, they are. For me I don’t think it’s ambiguous.

* Do you think that reading the body of your tests would be enough to understand what the requirement was? Are they -in other words- self describing which requirement they are meant to test? If not, what could you do to improve them?

It’s simple body test case and everyone should be understand what the requirement is especially regarding what mentioned in this document (See table description).

* In which tests did you need to use mock objects? Why?

It almost requires a mock object while creating test case in this assignment. Otherwise we can’t check whether the tests are passed and meet the requirements or not by comparing the results.

**When done and having used TDD as a method: Congratulations, you’ve just proved that all requirements are implemented in the code! Also all requirements are documented as tests!**

========================== End of practical. ===============================