# TCI – week 2 practical

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

Note: It is still assumed you create a Git repository for all your code. 1 repo per IntelliJ Project.

## Unit Tests

### 2.1 Assignment – Implement the Student and Group tests using mocks

Implement the teacher's demo in class for using mock objects for Student and Group tests yourself.

### 2.2 Assignment book: User Service Tested

Practice creating tests with mock objects: Do exercise 5.7.1 from the book.

Create test specifications using TestCherry first, before you create test code!

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

### Look at the requirements below and translate them into test specifications using TestCherry.

### Determine which test code to create first by looking at the test specifications. Create this test first, and the implementation code next. Create just enough implementation code to fulfil the test. (as meant by TDD)

Make sure you use the TDD rhythm as described in chapter 4. Proceed part by part.

The case is involving modeling of a Book with Chapters:

Part A:

A book always has a name and author (otherwise an IllegalArgumentException should be thrown) and it consists of zero or more Chapters.

Part B:

A Chapter is uniquely described by a name and number, and it can have a reference to one parent chapter when created. A chapter also has a collection of sub-chapters. When a reference to a parent chapter is present during creation of a chapter, the chapter is added to the parents' collection of subchapters. (This way each chapter knows their parent chapter (if they have one), and all chapters know their sub-chapters). Each collection of sub-chapters contains unique elements.

Part C (challenging!):

A table of contents (TOC) of a book can be created, which prints 1 line per chapter: The line consists of the chapter id and chapter name. A chapter id are the chapter numbers of all parent chapters of a chapter, and the chapter number itself, separated by a '. ' . For instance: chapter 3 "ch3 title", which is a subchapter of chapter 1 "the beginning", and chapter 1 is a chapter in the book, gets a line in the TOC like: 1.3 "ch3 title"

**Note: don’t get lost in creating code! Creating tests, based on requirements is the most important part!**

### 2.4 Assignment book: Creating Tests based on requirements AND using Mock Objects

Do exercise 5.7.2 from the book (Race Results Enhanced). Again, use TDD. So: test first, code second.

## Reflection points

* Did you manage to stick to the TDD rhythm? If not, what happened?
* How does using TDD feel compared to previous times? Can you see benefits or disadvantages of using TDD
* 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?
* In which tests did you need to use mock objects? Why? Could you see this in advance when looking at the test specification?
* What kind of order of tests do YOU like? easy first, difficult first, or mixed? Make sure group members know this of each other next project.

**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. ===============================