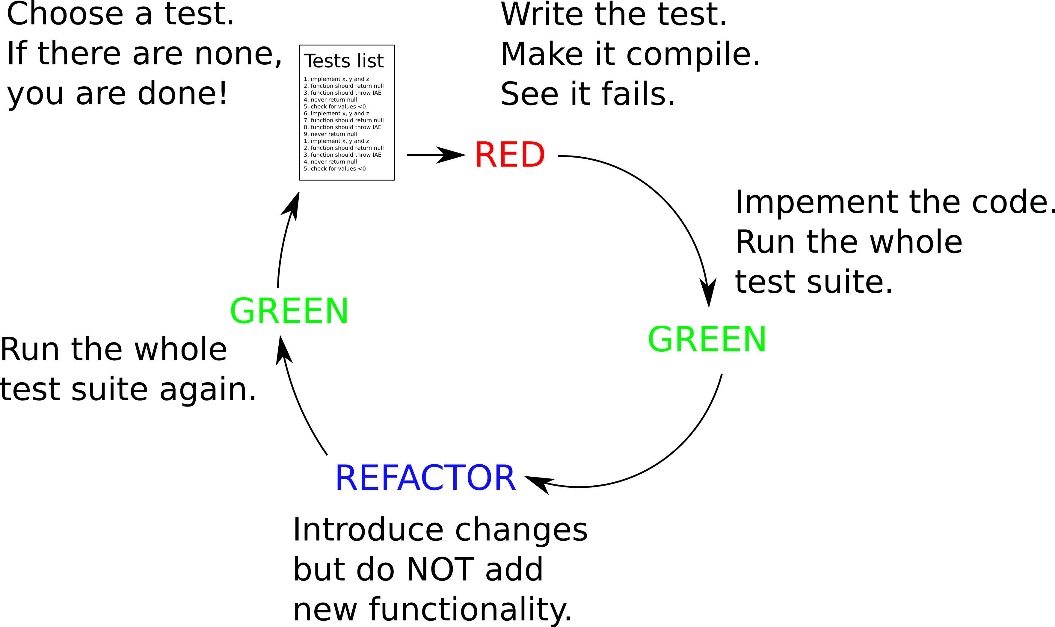
# TCI – week 2 practical

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

## Part A: TDD

### Assignment 1 – Using TDD to create justifiably correct code.

**Description**



Make sure you understand TDD! See chapter 4 of the book, sources in sheets and below.

Given is the description below of the requirements of a piece of software. It’s your job to use TDD to create the code. You can determine the order in which you will implement the requirements, explain your decision.

COMMIT TO GIT AFTER EACH TDD CYCLE!   
The history of your commits is used as proof you used TDD as a development approach. No history, no result (this will also be used this way in the final project)

You are going to build software for a school administration. The UML is given below:

Course objects function are placeholders for the name of the course, the start date and end date of the course. (This means a Course class is immutable: it only has a constructor to set values, and getters to access them.)

It has ONE requirement:

* The end date should be after the begin date, otherwise a CourseDateException is thrown.

Since it is immutable, and you can have IntelliJ create most code for it, you can assume this generated code is correct. So you only have to have tests for the one requirement above you need to implement!

Currently the following requirements for the School are stated:

A School has a name, an opening date and a collection of courses.

Name and opening date are not allowed to be null.

You can add a course to the school.

Course begin dates are after the school begin date. Throw a CourseException when it’s wrong.

The name of the course within a school is unique, otherwise a DuplicateCourseException is thrown.

You can get a course by name.

You can get a list of all course names.

You can get a list of copies of all courses

Use TDD to create tests & implement each requirement one by one. Commit to git after each creation of a test, and after each implementation belonging to that test.

Question for yourself (not associated with TDD, but OO development):

What would change in the requirements above when this should apply:



*…………………………………………………..*

*Sources:*

*WHAT & WHY of TDD:*

*Test-driven development: Write better code in less time* [*https://www.youtube.com/watch?v=HhwElTL-mdI*](https://www.youtube.com/watch?v=HhwElTL-mdI) *: till 15:30.*

*HOW to use TDD:*

*Uncle Bob demo:* [*https://www.youtube.com/watch?v=qkblc5WRn-U&t=1450*](https://www.youtube.com/watch?v=qkblc5WRn-U&t=1450)

*Extensive and really insightful example of using TDD, unit testing, parameterized tests etc. on Game of Life:* [*https://www.youtube.com/watch*](https://www.youtube.com/watch)[*v=2oa0U53B278&list=PLwAX\_Bwbts\_fI71bo62NwIMc05-uGigza*](https://www.youtube.com/watch?v=2oa0U53B278&list=PLwAX_Bwbts_fI71bo62NwIMc05-uGigza)

*TDD WITH Java and/or IntelliJ:*

*using TDD in IntelliJ:* [*https://www.youtube.com/watch?v=QDFI19lj4OM*](https://www.youtube.com/watch?v=QDFI19lj4OM)

<https://www.jetbrains.com/help/idea/tutorial-test-driven-development.html>

*Java tutorial: Test-Driven development with Junit:* [*https://www.youtube.com/watch?v=2Ekty7t621k*](https://www.youtube.com/watch?v=2Ekty7t621k)

## Part B: Continuous building

## Assignment 2 – Continuous building using Gradle

Gradle can perform continuous builds: it will scan your source directories for changes and automatically executes a build and tests: <https://blog.gradle.org/introducing-continuous-build> .

Try this out on your own computer.

## Assignment 3 – Configuring your local environment

This week, we are going to configure gradle to automate the tedious parts of being a software developer:

* Configure gradle to commit changes to your local repo automatically only when all of your unit tests are succesfully run. Document how you have done this, so you can share it within your group assignment. (you can find plugins for Gradle which add tasks to Gradle do do this: plugins.gradle.org)

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

## Appendix A

# Project Title

One Paragraph of project description goes here

## Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. See deployment for notes on how to deploy the project on a live system.

### Prerequisites

What things you need to install the software and how to install them

```

Give examples

```

### Installing

A step by step series of examples that tell you have to get a development env running

Say what the step will be

```

Give the example

```

And repeat

```

until finished

```

End with an example of getting some data out of the system or using it for a little demo

## Running the tests

Explain how to run the automated tests for this system

### Break down into end to end tests

Explain what these tests test and why

```

Give an example

```

### And coding style tests

Explain what these tests test and why

```

Give an example

```

## Deployment

Add additional notes about how to deploy this on a live system

## Built With

\* [Dropwizard](http://www.dropwizard.io/1.0.2/docs/) - The web framework used

\* [Maven](https://maven.apache.org/) - Dependency Management

\* [ROME](https://rometools.github.io/rome/) - Used to generate RSS Feeds

## Contributing

Please read [CONTRIBUTING.md](https://gist.github.com/PurpleBooth/b24679402957c63ec426) for details on our code of conduct, and the process for submitting pull requests to us.

## Versioning

We use [SemVer](http://semver.org/) for versioning. For the versions available, see the [tags on this repository](https://github.com/your/project/tags).

## Authors

\* \*\*Billie Thompson\*\* - \*Initial work\* - [PurpleBooth](https://github.com/PurpleBooth)

See also the list of [contributors](https://github.com/your/project/contributors) who participated in this project.

## License

This project is licensed under the MIT License - see the [LICENSE.md](LICENSE.md) file for details

## Acknowledgments

\* Hat tip to anyone who's code was used

\* Inspiration

\* etc

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