Programming – “Pretty Hair” #01

## Exercise 1: 70 minutes

You are going to implement part of the system for “Pretty Hair” which you have worked on previously.

Solve the following exercise using Boss-Secretary/Chef-Sekretær:

* Create Design Class Diagram, Sequence Diagrams and System Operations Contracts for the two Use Cases “Check stock” and “Receive order”.
* In Visual Studio create a Class Library for the source code, a Console project (for the application using the Class Library) and a Unit Test Project for the “Pretty Hair” system.
* Break down the Use Cases into tasks (use whatever artefacts you have from you prior work with “Pretty Hair”).
* Implement at least 8 test-methods for each Use Case.
* Implement the source code to get “Green lights” from all test-methods.

You can use MS unit test framework, NUnit or another test framework of your own choice, but you *must* use Test Driven Development!!

You must use collections / repositories at will, but you must be able to justify your choices.

You should not make your data persistent yet, as we will soon be using a relational database for that!

## Exercise 2: 20 minutes

Part 1 (10 minutes): Join with another pair and present your solution (VS setup, test methods and source code) for them. Listen to their feedback - do not discuss or argue in any way.

Part 2 (10 minutes): Join with another pair and watch their presentation. Give feedback to them using the criteria below and SOLID principles as inspiration. If the other pair tries to argue or defend their solution, tell them politely that the purpose is not to argue their solution but to give them another point of view. If they do not agree they are free to ignore your feedback.

## 

Quality criteria:

* Use Intention-Revealing / Meaningful Names
* Use Solution/Problem Domain Names
* Classes should be small!
* Functions should be small (Do one Thing)!
* Don't Repeat Yourself (Avoid Duplication)
* Explain yourself in code
* Make sure the code formatting is applied
* Use Exceptions rather than Return codes
* Make class final if not being used for inheritance
* Restrict privileges: Use the least privilege mode required (classes, interfaces, methods, and fields)
* Release (Close) resources (Streams, Connections, etc.) in all cases
* Validate inputs (for valid data, size, range, boundary conditions, etc.)
* Make public static fields final (to avoid caller changing the value)
* Beware the performance of string concatenation
* Avoid creating unnecessary objects
* Don't ignore exceptions
* Return empty arrays or collections, not nulls
* In public classes, use accessor methods, not public fields
* Refer to objects by their interfaces
* Always override ToString()
* Use enums instead of int constants
* Use marker interfaces to define types
* Valid test-classes and -methods exist