Lab Sheet: C# Advanced Language Features

This lab sheet covers the sections on C# Objects, C# Advanced Language Features and Code Contracts from the course on *Industrial Programming* (F21SC): http://www.macs.hw.ac.uk/~hwloidl/Courses/F21SC#slides. The tasks should be performed on the Windows lab (EM 2.45) machines in the department.

These exercises are *optional,* but aimed to deepen your understanding of the advanced language features that we covered in the course. Try at least one exercise from each section, to prepare you for the implementation of CW1.

C# Code Contracts (Week 4)

- Pick-up the "functions with code contracts" example (FunctionsWithCodeContracts.cs) and enable Code Contracts in Visual Studio. Which bugs in this sample code can be spotted through Code Contracts, using run-time checking or static analysis?
- Pick-up the "bank account" example (revision.cs) and add pre- and post-conditions to the methods (as discussed on the "C# Revision" slides).
- Pick-up the final sample source program from the "Case Study of Delegates and Generics" (GenGenSort.cs), and translate all the pre- and post-conditiond that are mentioned in comments, into Code Contracts (using `Contract.Requires` and `Contract.Ensures`).

C# GUIs (Week 5)

- Create a form with two text-areas as input fields, a text-area as output field, and a button to add these two values. Use these slides for basic information on how to build a GUI with Visual Studio.
- Extend the previous example to a simple calculator of basic arithmetic operations.

C# Advanced Language Features (Week 5)

- Extend the `twice` example, and define a method that applies a function (input: integer; output: integer) n-times to an argument, using *delegates*. Use this method to print all powers of 2 from 0 to 20.
- Modify the *binary search tree* example, using generics over the element type, as discussed in class.
- For the binary search tree, implement an indexer, for direct access to the i-th element
- For the binary search tree, implement an enumerator, to enable foreach loops (*Beware: this exercise is more difficult than the rest*).
- Use *delegates* to define a method that applies a method to every element of a binary search tree.