23. Planned parts of the book

Description

In this chapter, we list chapters and sections we plan for the next editions of this book. Whether or not these parts will be written, depends on the feedback we may get on this first edition and if we still find them to be relevant.

Using other languages than qBeta

For each chapter, we plan to add a section describing in general termes how the code of this chapter may be written in mainstream languages. At the moment, we hope to be able to cover C++, Java, C# and Python, but this may change.

Exercises

For each chapter and/or section, we plan to add a number of exercises.

Historical notes

We plan to have a section at the end of each chapter with a short mentioning of historical facts about the material of the chapter.

References

In this version of the book, there are almost no references to other books or papers. This is of course needed for a future version.

Supplementary properties

In this version, we have introduced access modifiers like %private. Access modifiers belong to a group of language mechanics in qBeta called supplementary properties. Supplementary properties may in general be used specify additional properties of an object, class, and method.

One important use of supplementary properties is to define classes like `MonitorSystem` that are safe with respect to race conditions.

Exceptions

A large part of any software design is the handling of error situations or rare situations that are unlikely to happen and this is called exception handling and the subject for a chapter in a coming version.

Programming style

This chapter will be about conventions for names of attributes, indentation, use of blank space, blank lines, comments, etc. This is usually referring to as programming style or coding style and may be important for large projects or companies to ensure that the programmers use the same style to improve readability, maintainability, etc. across programmers.

Design techniques for structures

This chapter will be about techniques for identification of phenomena and concepts and representing these using objects,

classes, data-items, and methods.

Design techniques for algorithms and data structures

A major part of the book has focus on describing the central language mechanism of object-oriented programming like objects and classes. A major part of programming is also about developing algorithms and data structures for solving a given problem in an efficient way. In , we have recommended taking courses on this subject.

In this chapter, we plan to introduce some of the basic techniques that are useful/necessary for developing algorithms. In section , we have as an example introduced recursion.

We plan tro introduce how to use assertions, which may be viewed as a kind of general snapshots that describe the state of an object that always holds at a given point in the statement part of a method or class. Such assertions may be used to express pre-conditions that must hold when executing a method and post-conditions that must hild after the method invocation. In addition assertions may be usd to express an invariant for a class, which must hold for an object before and after an invocation of one of its methods.