An introduction to Object-Oriented Programming as Modeling

Ole Lehrmann Madsen
Aarhus University

Birger Møller-Pedersen
University of Oslo

This book is a general introduction to object-oriented programming with emphasis on modeling.

When creating a program to support a task or solve a problem, you need to construct within the computer a model of that aspect of the problem domain to which the program will be applied. In such a model, relevant phenomena and concepts from the application domain must be represented by some entities within the computer.

The intended readers of this book are students and others that want to learn programming and more specifically object-oriented programming. No previous knowledge of programming is required. We expect the readers to be at the age of 17-19 or older.

The phenomena to be considered may be physical entities like bank accounts, patient record, vehicles, robots, etc. Physical entities are characterized by properties like weight and color of a vehicle. Physical entities may also carry out activities that may change the physical entities or observe properties of the physical entities. A concept is a generalized idea of a collection of phenomena, based on knowledge of common properties of elements in the collection.

Phenomena are represented by objects, which are the computer entities considered in this book. Objects may hold data and carry out computations. Concepts are represented by classes, which technically are templates for generating objects.

A program is described in a programming language and may be executed on a computer which then generates a so-called *program execution*, which is considered a model of the aspect of the application domain being considered.

Programming languages are complex entities and learning the details of such a language may be difficult and take time. There is great variation in the complexity of different languages. In this book we use a language with a relatively low complexity, but still the technicalities of this language may be a hindrance for some readers.

We introduce a conceptual framework for organizing and understanding knowledge about application domains. This includes understanding phenomena and concepts and conceptual means such as classification, composition and contextualization.

There are many books on object-oriented programming, just as there are many books on object-oriented modeling. Most textbooks on object-oriented programming do not consider modeling, but focus on the technical aspects of a programming language. The books on object-oriented modelling include weak links to programming and often lacks a proper conceptual framework.

In this book, we provide an introduction to object-oriented programming as modeling. We thus emphasize modeling, and we demonstrate that modeling can be done as part of programming.

A a description in a programming language is almost always made in the form of a text whereas a description in a modeling language is mainly made as a graphical diagram – the Unified Modeling Language (UML) is an example of this. For this reason, many people often take for granted that you have to use a graphical language to do object-oriented modeling.

In this book, we claim that you can use your programming language for modelling. You may use UML as well, but there is no need to. We do use graphical diagrams to illustrate our programs elements, whenever useful, but only as illustrations. If you prefer to use UML or any other graphical notation, this is of course possible.

The first object-oriented language, SIMULA, was intended for programming as well as modeling, but unfortunately this is not central in mainstream object-oriented programming.

We believe that the SIMULA approach is the right way to do it. The example programs in this book are therefore made in a simple modern language that is intended for programming as well as modeling, that is we go back to the future as pioneered by SIMULA.

The language being used is called qBeta and is derived from the Beta language. A preliminary description of qBeta may be found here.

The plan is to supply descriptions of how to make such programs in mainstream languages like, C++, Java, C#, and Python. We do, however, think that the current version may be a useful supplement to most courses on object-oriented programming and we think most instructors should be able to build the bridge between this book and textbooks about a given programming language.

Some people may find it confusing that students may have to deal with syntax of the language in this book and then in addition the syntax of a language like C++, Java, C#, or Python. We do, however, think that it is essential that students are trained in understanding language mechanisms independent of syntax of mainstream languages.

Students should start reading the introduction and subsequent chapters on objects and classes. They may then alternate between reading about similar mechanism of a mainstream language and further chapters in this book. It is essential that they try out the examples provided in this book.

We recommend starting with 1. Introduction.

The preface is only relevant if you want to know the background/motivation for writing this book.

The field below is a simple preliminary search field that search the pages in this book.

This is the first version of this book. Most parts are complete with a few exceptions mainly regarding formatting of the pages, resolution and size of illustrations, and lack of streamlining of diagrams.
For the next versions of the book, some new parts are planned – see chapter 23. Planned parts of the book.
Since this is the first version, we have not received feedback from any users. This means that the text including example code may contain errors or be hard to understand.
There may language elements we have forgotten to explain just as we may use terms that are not defined or need a better definition.
At the end of chapter 1. Introduction, we have listed a number of inconveniences that we hope to get rid of.