Learning Objectives in O1

Programming I a.k.a. O1 is an introductory programming course. After you complete O1, we hope that you can mean it when you say:

1. Computer programming is a meaningful and fun thing to do.
2. I know the basics of computer programming.

After O1, you should be able to write small application programs of your own. You should also be able to read and modify programs — even somewhat bigger ones — written by others. Moreover, you should have an understanding of various fundamental programming concepts that will help you develop your programming skills further, whether in the follow-on courses at Aalto or independently.

We use the Scala programming language as a tool, but most of what you learn in O1 is applicable when programming in other languages, too.

We've tried to design O1's grading policy (Chapter 1.1) so that a grade of one means you've reached the minimum requirements for passing, a grade of three roughly corresponds to what you need to know to succeed in follow-on courses, and a grade of five indicates a still better level of achievement.

Minimum Requirements

By the time you finish O1, you ought to have the following basic skills at the very least.

- You understand many **key concepts of computer programming**, especially those associated with imperative-style object-oriented programming: program code, variable, class, object, method, reference, data type, parameter, return value, expression, collections (e.g., buffer or vector) and basic methods for processing them (e.g., `map`, `filter`), selection (e.g., `if`), iteration with loops, `null` and `Option`, etc.
- You **are prepared for learning more** from materials that feature those concepts and terms.
- You can apply the programming concepts as you read and write programs. That is:
  - You can **read programs (written in Scala)** that consist of several interacting classes. You can analyze what given programs do, at least if the programs are no more than about a hundred lines of code long and don't feature demanding algorithms or exotic language constructs.
  - You can **implement classes (in Scala)**, assuming that you are given a specification that defines the classes' public interfaces In other words, you can implement the instance variables and methods of a specified class. You can write such implementations from scratch as well as edit a given program so that it meets a specification.
- You are familiar with some of the main **activities of a programmer**, such as reading a specification, writing code, and testing. You are capable of **using tools** that support the programmer (especially: you know how to use an IDE).
- You have at least a rudimentary understanding of **what a computer does as it runs a program**. You have an understanding of objects being created and initialized in memory. You have a sense of the role of the call stack in controlling how a program run unfolds.
- You know what a software library is. You can read the given documentation of a library and **use tools from a library** in your programs.
- You understand that **good programming style** matters. You can format your code so that it's easy for a human to read and understand.
- You have at least a rudimentary understanding of **how a graphical user interface works** in combination with the rest of an application. You have used a library for implementing simple GUIs.

What You'll Also Need in Follow-On Courses

The minimum requirements alone aren't enough if you intend to continue on to other courses at Aalto or if you develop a hunger to learn more about programming (as we expect). The spring courses Programming 2 and Programming Studio 2, for instance, expect that you know more than what was listed above. All students that plan to take those courses should aim for these objectives, too:

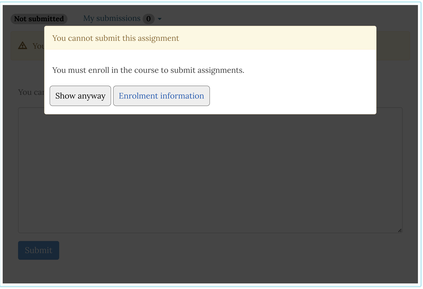
- You know what a compiler and a virtual machine are, and what bytecode and machine language are.
- You know a number of **programming techniques associated with functional programming** and thus have a platform for learning more about that programming paradigm. In particular:
  - You are accustomed to **working with immutable objects**, including immutable collections.
  - You have gained some practice on **using common higher-order methods** (such as `map` and `filter`). You can **make use of anonymous functions** as you use these methods.
- You understand the concepts of **inheritance** and **traits**. You can apply these techniques, too, as you read programs and as you implement programs to a specification.
- You know several ways to store multiple elements in a collection (e.g., vectors, buffers, maps, lazy-lists). At least in simple cases, you can pick a collection type that suits the program at hand.
- You have some practice on reading and editing **somewhat larger programs**. These programs may consist of up to hundreds of lines of code or a dozen or so classes and may feature algorithms that are somewhat more demanding to implement.
- You have at least a rudimentary appreciation of various **criteria for software quality**. For instance, you have seen examples of how the programmer's decisions impact on the modifiability and efficiency of a program. (However, you may still not be skilled in writing programs that are particularly modifiable or efficient.) You understand what refactoring is.

What Else Is on Offer?

There's a lot more that you can learn in O1. Here are some examples of learning objectives that you may go for (and that may boost your grade); below is just a small selection of topics from the optional sections in the ebook.

- You have tried **designing a program** without being given a detailed specification of what each of its components (classes and methods) needs to do. You have implemented a program of your own design.
- You can tell if a given program or program component is typical of **the imperative or the functional style of programming**. You are aware of some of the benefits attributed to these programming paradigms.
- You understand **the concept of recursion** and have seen examples of recursively defined data and recursive method implementations. You can implement a simple recursive class or method.
- You can **use a debugger** for examining a program and locating errors.
- You can use API functions to implement a program that **reads and/or writes (text) files**.
- You have at least an inkling of how a program may operate on other programs and how a programming language may be used for implementing another language.

Feedback



Credits

Thousands of students have given feedback and so contributed to this ebook's design. Thank you!

The ebook's chapters, programming assignments, and weekly bulletins have been written in Finnish and translated into English by Juha Sorva.

The appendices (glossary, Scala reference, FAQ, etc.) are by Juha Sorva unless otherwise specified on the page.

The automatic assessment of the assignments has been developed by: (in alphabetical order) Riku Autio, Nikolas Drosdek, Kaisa Ek, Joonatan Honkamaa, Antti Immonen, Jaakko Kantojärvi, Niklas Kröger, Kalle Laitinen, Teemu Lehtinen, Mikael Lenander, Ilona Ma, Jaakko Nakaza, Strasdosky Otewa, Timi Seppälä, Teemu Sirkiä, Anna Valldeoriola Cardó, and Aleksi Vartiainen.

The illustrations at the top of each chapter, and the similar drawings elsewhere in the ebook, are the work of Christina Lassheikki.

The animations that detail the execution Scala programs have been designed by Juha Sorva and Teemu Sirkiä. Teemu Sirkiä and Riku Autio did the technical implementation, relying on Teemu's Jsvee and Kelmu toolkits.

The other diagrams and interactive presentations in the ebook are by Juha Sorva.

The O1Library software has been developed by Aleksi Lukkarinen and Juha Sorva. Several of its key components are built upon Aleksi's SMCL library.

The pedagogy of using O1Library for simple graphical programming (such as `Pic`) is inspired by the textbooks *How to Design Programs* by Flatt, Felleisen, Findler, and Krishnamurthi and *Picturing Programs* by Stephen Bloch.

The course platform A+ was originally created at Aalto's LeTech research group as a student project. The open-source project is now shepherded by the Computer Science department's edu-tech team and hosted by the department's IT services. Markku Riekkinen is the current lead developer; dozens of Aalto students and others have also contributed.

The A+ Courses plugin, which supports A+ and O1 in IntelliJ IDEA, is another open-source project. It has been designed and implemented by various students in collaboration with O1's teachers.

For O1's current teaching staff, please see Chapter 1.1.

Additional credits appear at the ends of some chapters.