Learning to program with F#

Jon Sporring

Department of Computer Science,
University of Copenhagen

July 19, 2017

Contents

Chapter 16

Object-oriented programming

- Object-oriented programming
- objects
- attributeds
- methods
- models
- object-oriented analysis
- object-oriented design

Object-oriented programming is a programming paradigm that focusses on *objects* such as a person, place, thing, event, and concept relevant for the problem. Objects may contain data and code, which in the object-oriented paradigm are called *attributeds* and *methods*. Object-oriented programming is an extension of data types, in the sense that objects contains both data and functions in a similar manner as a module, but object-oriented programming emphasizes the semantic unity of the data and functions. Thus, objects are *models* of real world entities, and object-oriented programming leads to a particular style of programming analysis and design called *object-oriented analysis and design*.

An object is a variable of a class type. A class is defined using the "type" keyword, and there are allways parantheses after the class name. Consider the following problem.

Problem 16.1:

A complex number is a pair of real numbers called the real and the imaginary part and a set of operators. In particular, addition of two complex numbers is the the addition of their real parts and of their imaginary parts. Define a class for complex numbers including the addition operator.

A solution to this problem is as follows.

Listing 16.1, complex.fsx:
A class implementing complex numbers and the addition operator.

```
type complex(aReal,anImaginary) =
  member this.re = aReal
  member this.im = anImaginary
  member this.add (a : complex) : complex =
    new complex(this.re + a.re, this.im + a.im)

let x = new complex(1.0,2.0)
let y = new complex(2.5,-1.2)
let z = x.add(y)
printfn "(%A, %A) + (%A, %A) = (%A, %A)" x.re x.im y.re y.im z.re z.im
```

```
(1.0, 2.0) + (2.5, -1.2) = (3.5, 0.8)
```

Things to remember:

- upcast and downcast "upcast", ":>", "downcast", ":?>"
- boxing `(box 5) :?> int;;`, see Spec-4.0 chapter 18.2.6.
- obj type Spec-4.0 chapter 18.1
- boxing Spec-4.0 Section 18.2.6

[1]

[1] Todo: **In object oriented programming: functions and data are combined. Contrast the Anemic Domain Model (`https://www.martinfowler.com/bliki/AnemicDomainModel.html`)**

Appendix F

To Dos

- Remove EBNF from main body of the text, possibly extend the appendix
- Add appendix on regular expressions
- Add Torben's notes on functional programming
- Rewrite list chapter (add sequences?)
- Add a chapter comparing the 3 paradigms
- Write structured programming part
- Write chapter on pattern matching (if not already in Torben's notes)
- Add something on piping (if not already in Torben's notes)
- Add abstraction of computer: places <-> memory/disk. Mutable objects are abstractions of places `https://www.infoq.com/presentations/Value-Values`. Facts does not rime with set and get.
- Hickey: Difference between syntax and semantics. Values or locations, add a good figure. Functional programming: All values are freely shareable.
- something about organising stuff: `https://fsharpforfunandprofit.com/posts/organizing-functions/`, `https://fsharpforfunandprofit.com/posts/recipe-part3/`

Bibliography

[1] M. Auer, J. Poelz, A. Fuernweger, L. Meyer, and T. Tschurtschenthaler. Umlet, free uml tool for fast uml diagrams. `http://www.umlet.com`, Present version 14.2, version 1.0 was released June 21, 2002.

[2] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2):346–366, 1932.

[3] European Computer Manufacturers Association (ECMA). Standard ecma-335, common language infrastructure (cli). `http://www.ecma-international.org/publications/standards/Ecma-335.htm`.

[4] International Organization for Standardization. Iso/iec 23271:2012, common language infrastructure (cli). `https://www.iso.org/standard/58046.html`.

[5] Object Management Group. Uml version 2.0. `http://www.omg.org/spec/UML/2.0/`.

[6] Programming Research Group. Specifications for the ibm mathematical formula translating system, fortran. Technical report, Applied Science Division, International Business Machines Corporation, 1954.

[7] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM*, 3(4):184–195, 1960.

[8] X3: ASA Sectional Committee on Computers and Information Processing. American standard code for information interchange. Technical Report ASA X3.4-1963, American Standards Association (ASA), 1963. `http://worldpowersystems.com/projects/codes/X3.4-1963/`.

[9] George Pólya. *How to solve it*. Princeton University Press, 1945.

Index