

Think Java

How to Think Like a Computer Scientist

Think Java

How to Think Like a Computer Scientist

Allen B. Downey

5.0.5

Copyright © 2011 Allen Downey.

Permission is granted to copy, distribute, transmit and adapt this work under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License: <http://creativecommons.org/licenses/by-nc-sa/3.0/>.

If you are interested in distributing a commercial version of this work, please contact Allen B. Downey.

The original form of this book is \LaTeX source code. Compiling this \LaTeX source has the effect of generating a device-independent representation of the book, which can be converted to other formats and printed.

The \LaTeX source for this book is available from

`thinkapjava.com`

This book was typeset using \LaTeX . The illustrations were drawn in xfig. All of these are free, open-source programs.

Preface

“As we enjoy great Advantages from the Inventions of others, we should be glad of an Opportunity to serve others by any Invention of ours, and this we should do freely and generously.”

—Benjamin Franklin, quoted in *Benjamin Franklin* by Edmund S. Morgan.

Why I wrote this book

This is the fifth edition of a book I started writing in 1999, when I was teaching at Colby College. I had taught an introductory computer science class using the Java programming language, but I had not found a textbook I was happy with. For one thing, they were all too big! There was no way my students would read 800 pages of dense, technical material, even if I wanted them to. And I didn't want them to. Most of the material was too specific—details about Java and its libraries that would be obsolete by the end of the semester, and that obscured the material I really wanted to get to.

The other problem I found was that the introduction to object oriented programming was too abrupt. Many students who were otherwise doing well just hit a wall when we got to objects, whether we did it at the beginning, middle or end.

So I started writing. I wrote a chapter a day for 13 days, and on the 14th day I edited. Then I sent it to be photocopied and bound. When I handed it out on the first day of class, I told the students that they would be expected to read one chapter a week. In other words, they would read it seven times slower than I wrote it.

The philosophy behind it

Here are some of the ideas that make the book the way it is:

- Vocabulary is important. Students need to be able to talk about programs and understand what I am saying. I try to introduce the minimum number of terms, to define them carefully when they are first used, and to organize them in glossaries at the end of each chapter. In my class, I include vocabulary questions on quizzes and exams, and require students to use appropriate terms in short-answer responses.
- To write a program, students have to understand the algorithm, know the programming language, and they have to be able to debug. I think too many books neglect debugging. This book includes an appendix on debugging and an appendix on program development (which can help avoid debugging). I recommend that students read this material early and come back to it often.
- Some concepts take time to sink in. Some of the more difficult ideas in the book, like recursion, appear several times. By coming back to these ideas, I am trying to give students a chance to review and reinforce or, if they missed it the first time, a chance to catch up.
- I try to use the minimum amount of Java to get the maximum amount of programming power. The purpose of this book is to teach programming and some introductory ideas from computer science, not Java. I left out some language features, like the `switch` statement, that are unnecessary, and avoided most of the libraries, especially the ones like the AWT that have been changing quickly or are likely to be replaced.

The minimalism of my approach has some advantages. Each chapter is about ten pages, not including the exercises. In my classes I ask students to read each chapter before we discuss it, and I have found that they are willing to do that and their comprehension is good. Their preparation makes class time available for discussion of the more abstract material, in-class exercises, and additional topics that aren't in the book.

But minimalism has some disadvantages. There is not much here that is intrinsically fun. Most of my examples demonstrate the most basic use of a language feature, and many of the exercises involve string manipulation

and mathematical ideas. I think some of them are fun, but many of the things that excite students about computer science, like graphics, sound and network applications, are given short shrift.

The problem is that many of the more exciting features involve lots of details and not much concept. Pedagogically, that means a lot of effort for not much payoff. So there is a tradeoff between the material that students enjoy and the material that is most intellectually rich. I leave it to individual teachers to find the balance that is best for their classes. To help, the book includes appendices that cover graphics, keyboard input and file input.

Object-oriented programming

Some books introduce objects immediately; others warm up with a more procedural style and develop object-oriented style more gradually. This book uses the “objects late” approach.

Many of Java’s object-oriented features are motivated by problems with previous languages, and their implementations are influenced by this history. Some of these features are hard to explain if students aren’t familiar with the problems they solve.

It wasn’t my intention to postpone object-oriented programming. On the contrary, I got to it as quickly as I could, limited by my intention to introduce concepts one at a time, as clearly as possible, in a way that allows students to practice each idea in isolation before adding the next. But I have to admit that it takes some time to get there.

The Computer Science AP Exam

Naturally, when the College Board announced that the AP Exam would switch to Java, I made plans to update the Java version of the book. Looking at the proposed AP Syllabus, I saw that their subset of Java was all but identical to the subset I had chosen.

During January 2003, I worked on the Fourth Edition of the book, making these changes:

- I added sections to improve coverage of the AP syllabus.

- I improved the appendices on debugging and program development.
- I collected the exercises, quizzes, and exam questions I had used in my classes and put them at the end of the appropriate chapters. I also made up some problems that are intended to help with AP Exam preparation.

Finally, in August 2011 I wrote the fifth edition, adding coverage of the GridWorld Case Study that is part of the AP Exam.

Free books!

Since the beginning, this book has under a license that allows users to copy, distribute and modify the book. Readers can download the book in a variety of formats and read it on screen or print it. Teachers are free to print as many copies as they need. And anyone is free to customize the book for their needs.

People have translated the book into other computer languages (including Python and Eiffel), and other natural languages (including Spanish, French and German). Many of these derivatives are also available under free licenses.

Motivated by Open Source Software, I adopted the philosophy of releasing the book early and updating it often. I do my best to minimize the number of errors, but I also depend on readers to help out.

The response has been great. I get messages almost every day from people who have read the book and liked it enough to take the trouble to send in a “bug report.” Often I can correct an error and post an updated version within a few minutes. I think of the book as a work in progress, improving a little whenever I have time to make a revision, or when readers send feedback.

Oh, the title

I get a lot of grief about the title of the book. Not everyone understands that it is—mostly—a joke. Reading this book will probably not make you think like a computer scientist. That takes time, experience, and probably a few more classes.

But there is a kernel of truth in the title: this book is not about Java, and it is only partly about programming. If it is successful, this book is about a way of thinking. Computer scientists have an approach to problem-solving, and a way of crafting solutions, that is unique, versatile and powerful. I hope that this book gives you a sense of what that approach is, and that at some point you will find yourself thinking like a computer scientist.

Allen Downey
Needham, Massachusetts
July 13, 2011

Contributors List

When I started writing free books, it didn't occur to me to keep a contributors list. When Jeff Elkner suggested it, it seemed so obvious that I am embarrassed by the omission. This list starts with the 4th Edition, so it omits many people who contributed suggestions and corrections to earlier versions.

If you have additional comments, please send them to feedback@greenteapress.com.

- Ellen Hildreth used this book to teach Data Structures at Wellesley College, and she gave me a whole stack of corrections, along with some great suggestions.
- Tania Passfield pointed out that the glossary of Chapter 4 has some leftover terms that no longer appear in the text.
- Elizabeth Wiethoff noticed that my series expansion of e^{-x^2} was wrong. She is also working on a Ruby version of the book!
- Matt Crawford sent in a whole patch file full of corrections!
- Chi-Yu Li pointed out a typo and an error in one of the code examples.
- Doan Thanh Nam corrected an example in Chapter 3.
- Stijn Debrouwere found a math typo.

- Muhammad Saied translated the book into Arabic, and found several errors.
- Marius Margowski found an inconsistency in a code example.
- Guy Driesen found several typos.

Contents

Preface	v
1 The way of the program	1
1.1 What is a programming language?	1
1.2 What is a program?	3
1.3 What is debugging?	4
1.4 Formal and natural languages	6
1.5 The first program	8
1.6 Glossary	9
1.7 Exercises	11
2 Variables and types	13
2.1 More printing	13
2.2 Variables	15
2.3 Assignment	15
2.4 Printing variables	16
2.5 Keywords	18
2.6 Operators	18

2.7	Order of operations	19
2.8	Operators for Strings	20
2.9	Composition	20
2.10	Glossary	21
2.11	Exercises	22
3	Methods	25
3.1	Floating-point	25
3.2	Converting from double to int	26
3.3	Math methods	27
3.4	Composition	28
3.5	Adding new methods	29
3.6	Classes and methods	31
3.7	Programs with multiple methods	32
3.8	Parameters and arguments	33
3.9	Stack diagrams	34
3.10	Methods with multiple parameters	35
3.11	Methods with results	36
3.12	Glossary	36
3.13	Exercises	37
4	Conditionals and recursion	39
4.1	The modulus operator	39
4.2	Conditional execution	39
4.3	Alternative execution	40

4.4	Chained conditionals	41
4.5	Nested conditionals	42
4.6	The return statement	43
4.7	Type conversion	43
4.8	Recursion	44
4.9	Stack diagrams for recursive methods	46
4.10	Glossary	46
4.11	Exercises	47
5	GridWorld: Part One	51
5.1	Getting started	51
5.2	BugRunner	52
6	Fruitful methods	55
6.1	Return values	55
6.2	Program development	57
6.3	Composition	60
6.4	Overloading	60
6.5	Boolean expressions	62
6.6	Logical operators	63
6.7	Boolean methods	63
6.8	More recursion	64
6.9	Leap of faith	67
6.10	One more example	68
6.11	Glossary	68
6.12	Exercises	69

7	Iteration	75
7.1	Multiple assignment	75
7.2	Iteration	76
7.3	The <code>while</code> statement	76
7.4	Tables	78
7.5	Two-dimensional tables	81
7.6	Encapsulation and generalization	81
7.7	Methods	83
7.8	More encapsulation	83
7.9	Local variables	84
7.10	More generalization	84
7.11	Glossary	86
7.12	Exercises	87
8	Strings and things	91
8.1	Invoking methods on objects	91
8.2	Length	92
8.3	Traversal	93
8.4	Run-time errors	93
8.5	Reading documentation	95
8.6	The <code>indexOf</code> method	96
8.7	Looping and counting	96
8.8	Increment and decrement operators	97
8.9	Strings are immutable	98
8.10	Strings are incomparable	99
8.11	Glossary	100
8.12	Exercises	100

9	Mutable objects	107
9.1	Points and Rectangles	107
9.2	Packages	107
9.3	Point objects	108
9.4	Instance variables	109
9.5	Objects as parameters	110
9.6	Rectangles	110
9.7	Objects as return types	111
9.8	Objects are mutable	111
9.9	Aliasing	113
9.10	null	114
9.11	Garbage collection	114
9.12	Objects and primitives	115
9.13	Glossary	116
9.14	Exercises	117
10	GridWorld: Part 2	123
10.1	Termites	125
10.2	Langton's Termite	129
11	Create your own objects	131
11.1	Class definitions and object types	131
11.2	Time	132
11.3	Constructors	133
11.4	More constructors	134

11.5	Creating a new object	135
11.6	Printing objects	136
11.7	Operations on objects	137
11.8	Pure functions	137
11.9	Modifiers	140
11.10	Fill-in methods	141
11.11	Incremental development and planning	142
11.12	Generalization	143
11.13	Algorithms	144
11.14	Glossary	144
11.15	Exercises	145
12	Arrays	149
12.1	Accessing elements	150
12.2	Copying arrays	151
12.3	for loops	151
12.4	Arrays and objects	152
12.5	Array length	153
12.6	Random numbers	153
12.7	Array of random numbers	154
12.8	Counting	156
12.9	The histogram	157
12.10	A single-pass solution	158
12.11	Glossary	158
12.12	Exercises	159

13 Arrays of Objects	165
13.1 The Road Ahead	165
13.2 Card objects	165
13.3 The <code>printCard</code> method	167
13.4 The <code>sameCard</code> method	169
13.5 The <code>compareCard</code> method	170
13.6 Arrays of cards	171
13.7 The <code>printDeck</code> method	173
13.8 Searching	173
13.9 Decks and subdecks	177
13.10 Glossary	178
13.11 Exercises	178
14 Objects of Arrays	181
14.1 The <code>Deck</code> class	181
14.2 Shuffling	183
14.3 Sorting	184
14.4 Subdecks	184
14.5 Shuffling and dealing	185
14.6 Mergesort	186
14.7 Class variables	189
14.8 Glossary	189
14.9 Exercises	190

Thank You for previewing this eBook

You can read the full version of this eBook in different formats:

- HTML (Free /Available to everyone)
- PDF / TXT (Available to V.I.P. members. Free Standard members can access up to 5 PDF/TXT eBooks per month each month)
- Epub & Mobipocket (Exclusive to V.I.P. members)

To download this full book, simply select the format you desire below

