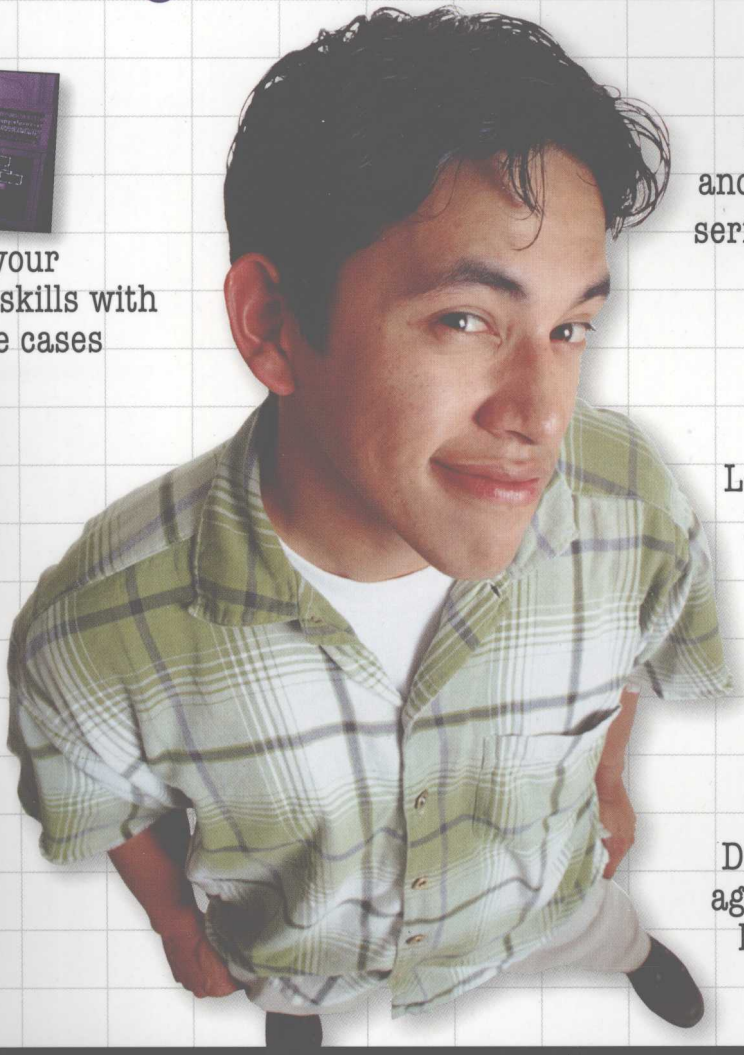
深入浅出面向对象分析与设计（影印版）

Head First Object-Oriented Analysis & Design

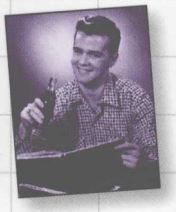
Improve your communication skills with UML and use cases

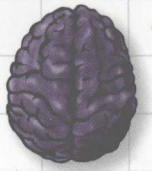
Bend your mind around dozens of OO exercises

Avoid leaving your customers unsatisfied



Turn your requirements and designs into serious software





Load important OO design principles straight into your brain

Discover how abstraction, aggregation, and delegation helped Mary get around Objectville

O'REILLY® 東南大學出版社

Brett D. McLaughlin, Gary Pollice & David West 著

深入浅出面向对象分析与设计(影印版)

Head First Object-Oriented Analysis and Design

Brett D. McLaughlin
Gary Pollice
David West

O'REILLY®

Beijing • Cambridge • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

深入浅出面向对象分析与设计：英文 /（美）麦克劳夫林（McLaughlin,B. D.）等著. —影印本. —南京：东南大学出版社，2007.6

书名原文：Head First Object-Oriented Analysis & Design

ISBN 978-7-5641-0743-7

Ⅰ. 深… Ⅱ. 麦… Ⅲ. 面向对象语言－程序设计－英文 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2007）第 063449 号

江苏省版权局著作权合同登记

图字：10-2007-102 号

书　　名 / 深入浅出面向对象分析与设计（影印版）

责任编辑 / 张烨

封面设计 / Mike Kohnke, Edie Freeman，张健

出版发行 / 东南大学出版社（press.seu.edu.cn）

地　　址 / 南京四牌楼 2 号（邮政编码 210096）

印　　刷 / 扬中市印刷有限公司

开　　本 / 787 毫米 × 980 毫米　12 开本　53 印张

版　　次 / 2007 年 6 月第 1 版　2007 年 6 月第 1 次印刷

印　　数 / 0001-6000 册

书　　号 / ISBN 978-7-5641-0743-7/TP · 113

定　　价 / 98.00 元（册）

Praise for *Head First OOA&D*

"*Head First Object-Oriented Analysis and Design* is a refreshing look at the subject of OOA&D. What sets this book apart is its focus on learning. There are too many books on the market that spend a lot of time telling you why, but do not actually enable the practitioner to start work on a project. Those books are very interesting, but not very practical. I strongly believe that the future of software development practice will focus on the practitioner. The authors have made the content of OOA&D accessible and usable for the practitioner "

— Ivar Jacobson, Ivar Jacobson Consulting

"I just finished reading *HF OOA&D*, and I loved it! The book manages to get across the essentials of object-oriented analysis and design with UML and use cases, and even several lectures on good software design, all in a fast-paced, easy to understand way. The thing I liked most about this book was its focus on why we do OOA&D—to write great software! By defining what great software is and showing how each step in the OOA&D process leads you towards that goal, it can teach even the most jaded Java programmer why OOA&D matters. This is a great 'first book' on design for anyone who is new to Java, or even for those who have been Java programmers for a while but have been scared off by the massive tomes on OO Analysis and Design."

— Kyle Brown, Distinguished Engineer, IBM

"Finally a book on OOA&D that recognizes that the UML is just a notation and that what matters when developing software is taking the time to think the issues through."

— Pete McBreen, Author, *Software Craftsmanship*

"The book does a good job of capturing that entertaining, visually oriented, 'Head First' writing style. But hidden behind the funny pictures and crazy fonts is a serious, intelligent, extremely well-crafted presentation of OO Analysis and Design. This book has a strong opinion of how to design programs, and communicates it effectively. I love the way it uses running examples to lead the reader through the various stages of the design process. As I read the book, I felt like I was looking over the shoulder of an expert designer who was explaining to me what issues were important at each step, and why."

— Edward Sciore, Associate Professor, Computer Science Department
Boston College

"This is a well-designed book that delivers what it promises to its readers: how to analyze, design, and write serious object-oriented software. Its contents flow effortlessly from using use cases for capturing requirements to analysis, design, implementation, testing, and iteration. Every step in the development of object-oriented software is presented in light of sound software engineering principles. The examples are clear and illustrative. This is a solid and refreshing book on object-oriented software development."

— Dung Zung Nguyen, Lecturer
Rice University

Praise for other *Head First* books by the authors

"When arriving home after a 10-hour day at the office programming, who has the energy to plow through yet another new facet of emerging technology? If a developer is going to invest free time in self-driven career development, should it not be at least remotely enjoyable? Judging from the content of O'Reilly's new release *Head Rush Ajax*, the answer is yes...*Head Rush Ajax* is a most enjoyable launchpad into the world of Ajax web applications, well worth the investment in time and money."

— Barry Hawkins, Slashdot.org

"By starting with simple concepts and examples, the book gently takes the reader from humble beginnings to (by the end of the book) where the reader should be comfortable creating Ajax-based websites... Probably the best web designer centric book on Ajax."

— Stefan Mischook, Killersites.com

"Using the irreverent style common of the *Head First/Head Rush* series of books, this book starts at the beginning and introduces you to all you need to know to be able to write the JavaScript that will both send requests to the server and update the page with the results when they are returned...One of the best things about this book (apart form the excellent explanations of how the code works) is that it also looks at security issues...If you learn Ajax from this book you are unlikely to forget much of what you learn."

— Stephen Chapman, JavaScript.About.com

"*Head Rush Ajax* is the book if you want to cut through all the hype and learn how to make your web apps sparkled...your users will love you for it!"

— Kristin Stromberg, Aguirre International

"If you know some HTML, a dollop of CSS, a little JavaScript, and a bit of PHP, but you're mystified about what all the Ajax hype is about, this book is for you...You'll have a blast learning Ajax with *Head Rush Ajax*. By the time you've reached the end of the book, all those web technologies that didn't quite fit together in your head will all snap into place and you'll have The Ajax Power! You'll know the secrets behind some of the most popular web applications on the Internet. You'll impress your friends and co-workers with you knowledge of how those interactive maps and web forms really work."

— Elisabeth Freeman, Director, Technology, The Walt Disney Internet Group
Co-Author, *Head First Design Patterns* **and** *Head First HTML with CSS & XHTML*

"If you thought Ajax was rocket science, this book is for you. *Head Rush Ajax* puts dynamic, compelling experiences within reach for every web developer."

— Jesse James Garrett, Adaptive Path

"This stuff is brain candy; I can't get enough of it."

— Pauline McNamara, Center for New Technologies and Education
Fribourg University, Switzerland

Praise for other *Head First* Books

"I *heart* *Head First HTML with CSS & XHTML* – it teaches you everything you need to learn in a 'fun coated' format!"

— Sally Applin, UI Designer and Fine Artist, http://sally.com.

"My wife stole the book. She's never done any web design, so she needed a book like *Head First HTML with CSS & XHTML* to take her from beginning to end. She now has a list of web sites she wants to build – for our son's class, our family, ... If I'm lucky, I'll get the book back when she's done."

— David Kaminsky, Master Inventor, IBM

"Freeman's *Head First HTML with CSS & XHTML* is a most entertaining book for learning how to build a great web page. It not only covers everything you need to know about HTML, CSS, and XHTML, it also excels in explaining everything in layman's terms with a lot of great examples. I found the book truly enjoyable to read, and I learned something new!"

— Newton Lee, Editor-in-Chief, ACM Computers in Entertainment
http://www.acmcie.org

From the awesome *Head First Java* folks, this book uses every conceivable trick to help you understand and remember. Not just loads of pictures: pictures of humans, which tend to interest other humans. Surprises everywhere. Stories, because humans love narrative. (Stories about things like pizza and chocolate. Need we say more?) Plus, it's darned funny.

— Bill Camarda, READ ONLY

"This book's admirable clarity, humor and substantial doses of clever make it the sort of book that helps even non-programmers think well about problem-solving."

— Cory Doctorow, co-editor of Boing Boing
Author, "Down and Out in the Magic Kingdom"
and "Someone Comes to Town, Someone Leaves Town"

"I feel like a thousand pounds of books have just been lifted off of my head."

— Ward Cunningham, inventor of the Wiki
and founder of the Hillside Group

"I literally love this book. In fact, I kissed this book in front of my wife."

— Satish Kumar

Other related books from O'Reilly

Practical Development Environments

Process Improvement Essentials

Prefactoring

Ajax Design Patterns

Learning UML

Applied Software Project Management

The Art of Project Management

UML 2.0 in a Nutshell

Unit Test Frameworks

Other books in O'Reilly's *Head First* Series

Head First Design Patterns

Head First Java

Head First Servlets and JSP

Head First EJB

Head First HTML with CSS & XHTML

Head Rush Ajax

Head First OOA&D

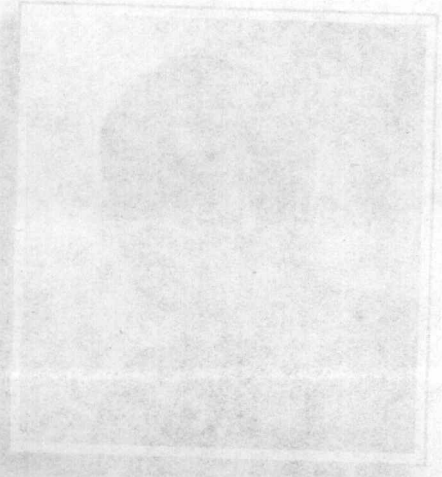
Head First PMP (2007)

Head First Algebra (2007)

Head First Software Development (2007)

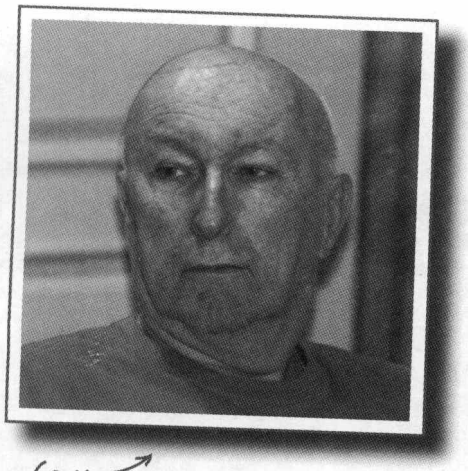
To all the brilliant people who came up with various ways to gather requirements, analyze software, and design code...

...thanks for coming up with something good enough to produce great software, but hard enough that we needed this book to explain it all.



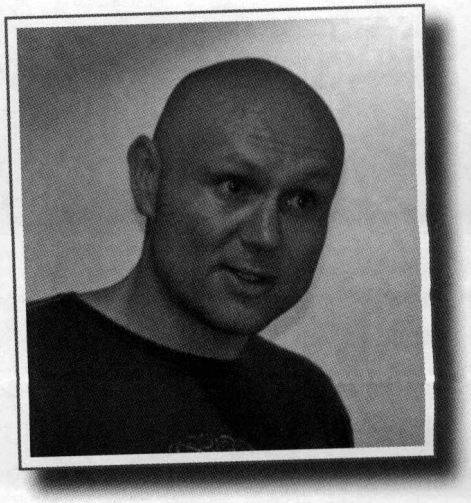
Brett McLaughlin is a guitar player who is still struggling with the realization that you can't pay the bills if you're into acoustic fingerstyle blues and jazz. He's just recently discovered, to his delight, that writing books that help people become better programmers does pay the bills. He's very happy about this, as are his wife Leigh, and his kids, Dean and Robbie.

Before Brett wandered into Head First land, he developed enterprise Java applications for Nextel Communications and Allegiance Telecom. When that became fairly mundane, Brett took on application servers, working on the internals of the Lutris Enhydra servlet engine and EJB container. Along the way, Brett got hooked on open source software, and helped found several cool programming tools, like Jakarta Turbine and JDOM. Write to him at brett@oreilly.com.



Gary Pollice is a self-labeled curmudgeon (that's a crusty, ill-tempered, usually old man) who spent over 35 years in industry trying to figure out what he wanted to be when he grew up. Even though he hasn't grown up yet, he did make the move in 2003 to the hallowed halls of academia where he has been corrupting the minds of the next generation of software developers with radical ideas like, "develop software for your customer, learn how to work as part of a team, design and code quality and elegance and correctness counts, and it's okay to be a nerd as long as you are a great one."

Gary is a Professor of Practice (meaning he had a real job before becoming a professor) at Worcester Polytechnic Institute. He lives in central Massachusetts with his wife, Vikki, and their two dogs, Aloysius and Ignatius. You can visit his WPI home page at `http://web.cs.wpi.edu/~gpollice/`. Feel free to drop him a note and complain or cheer about the book.

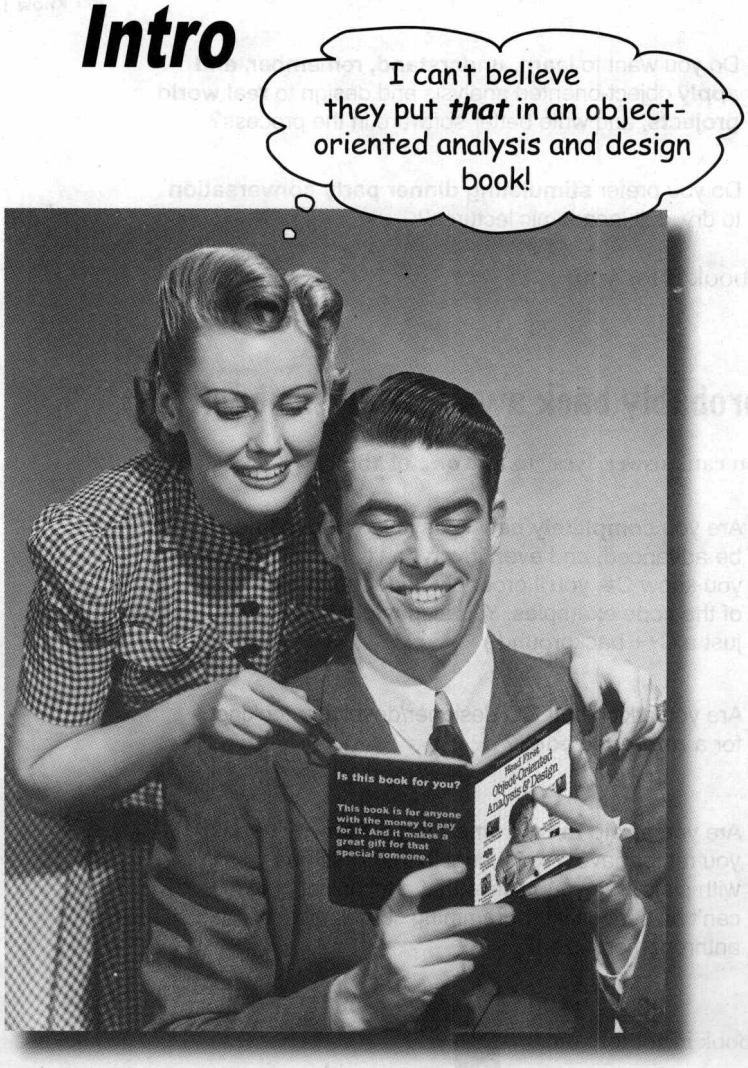


Dave West would like to describe himself as sheik geek. Unfortunately no one else would describe him in that way. They would say he is a professional Englishman who likes to talk about software development best practices with the passion and energy of an evangelical preacher. Recently Dave has moved to Ivar Jacobson Consulting, where he runs the Americas and can combine his desire to talk about software development and spread the word on rugby and football, and argue that cricket is more exciting than baseball.

Before running the Americas for Ivar Jacobson Consulting, Dave worked for a number of years at Rational Software (now a part of IBM). Dave held many positions at Rational and then IBM, including Product Manager for RUP where he introduced the idea of process plug-ins and agility to RUP. Dave can be contacted at dwest@ivarjacobson.com.

how to use this book

Intro



In this section, we answer the burning question:
"So why DID they put that in an OOA&D book?"

Who is this book for?

If you can answer "yes" to all of these:

1. Do you know **Java**? (You don't need to be a guru.) ← You'll probably be okay if you know C# instead.
2. Do you want to **learn, understand, remember, and *apply*** object-oriented analysis and design to **real world projects**, and write better software in the process?
3. Do you prefer **stimulating dinner party conversation** to dry, dull, academic lectures?

this book is for you.

Who should probably back away from this book?

If you can answer "yes" to any ***one*** of these:

1. Are you **completely new** to Java? (You don't need to be advanced, and even if you don't know Java, but you know C#, you'll probably understand almost all of the code examples. You also might be okay with just a C++ background.)
2. Are you a kick-butt OO designer/developer looking for a ***reference* book?**
3. Are you **afraid to try something different**? Would you rather have a root canal than mix stripes with plaid? Do you believe that a technical book can't be serious if programming concepts are anthropomorphized?

this book is not for you.

[note from marketing: this book is for anyone with a credit card.]

We know what you're thinking.

"How can *this* be a serious programming book?"

"What's with all the graphics?"

"Can I actually *learn* it this way?"

And we know what your *brain* is thinking.

Your brain craves novelty. It's always searching, scanning, *waiting* for something unusual. It was built that way, and it helps you stay alive.

So what does your brain do with all the routine, ordinary, normal things you encounter? Everything it *can* to stop them from interfering with the brain's *real* job—recording things that *matter*. It doesn't bother saving the boring things; they never make it past the "this is obviously not important" filter.

How does your brain *know* what's important? Suppose you're out for a day hike and a tiger jumps in front of you, what happens inside your head and body?

Neurons fire. Emotions crank up. *Chemicals surge.*

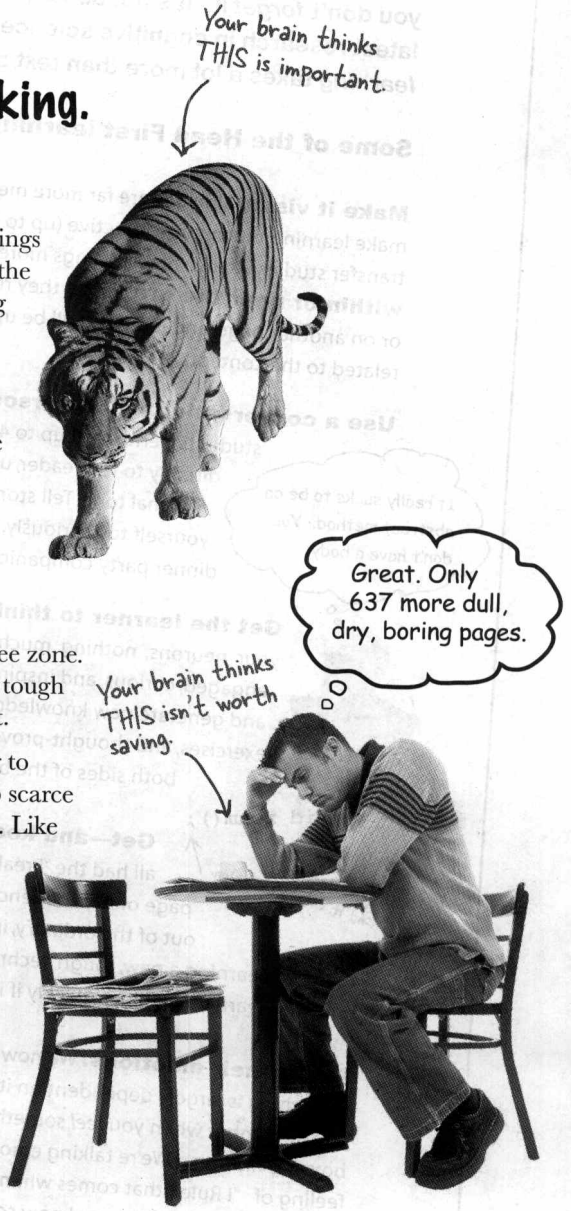
And that's how your brain knows...

This must be important! Don't forget it!

But imagine you're at home, or in a library. It's a safe, warm, tiger-free zone. You're studying. Getting ready for an exam. Or trying to learn some tough technical topic your boss thinks will take a week, ten days at the most.

Just one problem. Your brain's trying to do you a big favor. It's trying to make sure that this *obviously* non-important content doesn't clutter up scarce resources. Resources that are better spent storing the really *big* things. Like tigers. Like the danger of fire. Like how you should never again snowboard in shorts.

And there's no simple way to tell your brain, "Hey brain, thank you very much, but no matter how dull this book is, and how little I'm registering on the emotional Richter scale right now, I really *do* want you to keep this stuff around."

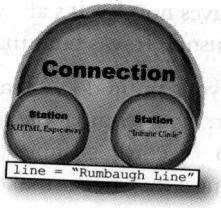


We think of a "Head First" reader as a learner.

So what does it take to *learn* something? First, you have to *get* it, then make sure you don't *forget* it. It's not about pushing facts into your head. Based on the latest research in cognitive science, neurobiology, and educational psychology, *learning* takes a lot more than text on a page. We know what turns your brain on.

Some of the Head First learning principles:

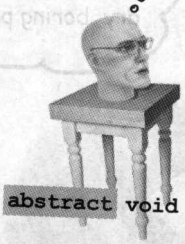
All of this is represented in a single Connection object.



Make it visual. Images are far more memorable than words alone, and make learning much more effective (up to 89% improvement in recall and transfer studies). It also makes things more understandable. **Put the words within or near the graphics** they relate to, rather than on the bottom or on another page, and learners will be up to *twice* as likely to solve problems related to the content.

Use a conversational and personalized style. In recent studies, students performed up to 40% better on post-learning tests if the content spoke directly to the reader, using a first-person, conversational style rather than taking a formal tone. Tell stories instead of lecturing. Use casual language. Don't take yourself too seriously. Which would *you* pay more attention to: a stimulating dinner party companion, or a lecture?

It really sucks to be an abstract method. You don't have a body.



```
abstract void roam();
```

No method body! End it with a semicolon.

Get the learner to think more deeply. In other words, unless you actively flex your neurons, nothing much happens in your head. A reader has to be motivated, engaged, curious, and inspired to solve problems, draw conclusions, and generate new knowledge. And for that, you need challenges, exercises, and thought-provoking questions, and activities that involve both sides of the brain, and multiple senses.

Great software *every time*? I can hardly **imagine** what that would be like!

Get—and keep—the reader's attention. We've all had the "I really want to learn this but I can't stay awake past page one" experience. Your brain pays attention to things that are out of the ordinary, interesting, strange, eye-catching, unexpected. Learning a new, tough, technical topic doesn't have to be boring. Your brain will learn much more quickly if it's not.

Touch their emotions. We now know that your ability to remember something is largely dependent on its emotional content. You remember what you *care* about. You remember when you *feel* something. No, we're not talking heart-wrenching stories about a boy and his dog. We're talking emotions like surprise, curiosity, fun, "what the...?", and the feeling of "I Rule!" that comes when you solve a puzzle, learn something everybody else thinks is hard, or realize you know something that "I'm more technical than thou" Bob from engineering *doesn't*.

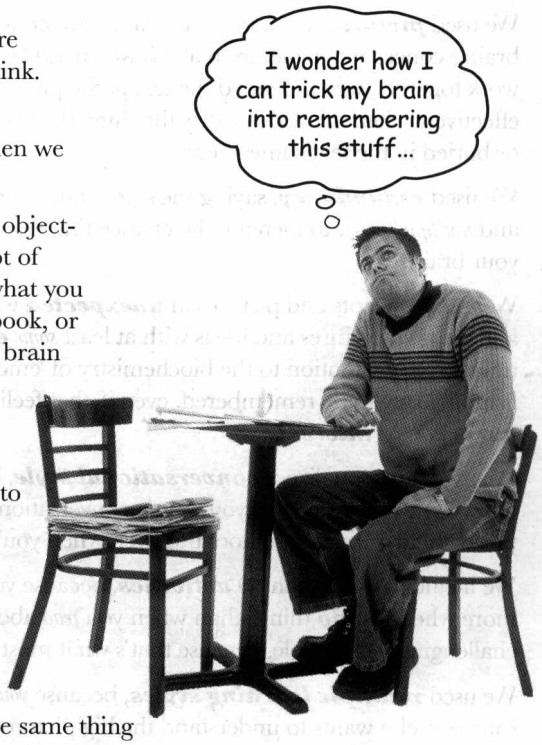
Metacognition: thinking about thinking

If you really want to learn, and you want to learn more quickly and more deeply, pay attention to how you pay attention. Think about how you think. Learn how you learn.

Most of us did not take courses on metacognition or learning theory when we were growing up. We were *expected* to learn, but rarely *taught* to learn.

But we assume that if you're holding this book, you really want to learn object-oriented analysis and design. And you probably don't want to spend a lot of time. And since you're going to develop software, you need to *remember* what you read. And for that, you've got to *understand* it. To get the most from this book, or *any* book or learning experience, take responsibility for your brain. Your brain on *this* content.

The trick is to get your brain to see the new material you're learning as Really Important. Crucial to your well-being. As important as a tiger. Otherwise, you're in for a constant battle, with your brain doing its best to keep the new content from sticking.



So just how *DO* you get your brain to think object-oriented analysis and design is a hungry tiger?

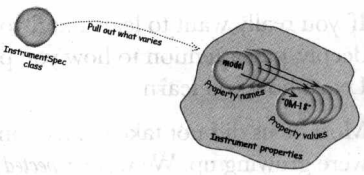
There's the slow, tedious way, or the faster, more effective way. The slow way is about sheer repetition. You obviously know that you *are* able to learn and remember even the dullest of topics if you keep pounding the same thing into your brain. With enough repetition, your brain says, "This doesn't *feel* important to him, but he keeps looking at the same thing *over* and *over* and *over*, so I suppose it must be."

The faster way is to do ***anything that increases brain activity,*** especially different *types* of brain activity. The things on the previous page are a big part of the solution, and they're all things that have been proven to help your brain work in your favor. For example, studies show that putting words *within* the pictures they describe (as opposed to somewhere else in the page, like a caption or in the body text) causes your brain to try to makes sense of how the words and picture relate, and this causes more neurons to fire. More neurons firing = more chances for your brain to *get* that this is something worth paying attention to, and possibly recording.

A conversational style helps because people tend to pay more attention when they perceive that they're in a conversation, since they're expected to follow along and hold up their end. The amazing thing is, your brain doesn't necessarily *care* that the "conversation" is between you and a book! On the other hand, if the writing style is formal and dry, your brain perceives it the same way you experience being lectured to while sitting in a roomful of passive attendees. No need to stay awake.

But pictures and conversational style are just the beginning.

Here's what WE did:



We used ***pictures***, because your brain is tuned for visuals, not text. As far as your brain's concerned, a picture really *is* worth 1,024 words. And when text and pictures work together, we embedded the text *in* the pictures because your brain works more effectively when the text is *within* the thing the text refers to, as opposed to in a caption or buried in the text somewhere.

We used ***redundancy***, saying the same thing in *different* ways and with different media types, and *multiple senses*, to increase the chance that the content gets coded into more than one area of your brain.

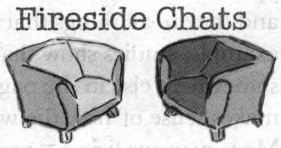
We used concepts and pictures in ***unexpected*** ways because your brain is tuned for novelty, and we used pictures and ideas with at least *some* ***emotional*** *content*, because your brain is tuned to pay attention to the biochemistry of emotions. That which causes you to *feel* something is more likely to be remembered, even if that feeling is nothing more than a little ***humor***, ***surprise***, or ***interest.***

We used a personalized, ***conversational style***, because your brain is tuned to pay more attention when it believes you're in a conversation than if it thinks you're passively listening to a presentation. Your brain does this even when you're *reading*.

We included more than 80 ***activities***, because your brain is tuned to learn and remember more when you ***do*** things than when you *read* about things. And we made the exercises challenging-yet-do-able, because that's what most people prefer.

We used ***multiple learning styles***, because *you* might prefer step-by-step procedures, while someone else wants to understand the big picture first, and someone else just wants to see a code example. But regardless of your own learning preference, *everyone* benefits from seeing the same content represented in multiple ways.

We include content for ***both sides of your brain***, because the more of your brain you engage, the more likely you are to learn and remember, and the longer you can stay focused. Since working one side of the brain often means giving the other side a chance to rest, you can be more productive at learning for a longer period of time.



And we included ***stories*** and exercises that present ***more than one point of view,*** because your brain is tuned to learn more deeply when it's forced to make evaluations and judgements.

We included ***challenges***, with exercises, and by asking ***questions*** that don't always have a straight answer, because your brain is tuned to learn and remember when it has to *work* at something. Think about it—you can't get your *body* in shape just by *watching* people at the gym. But we did our best to make sure that when you're working hard, it's on the *right* things. That ***you're not spending one extra dendrite*** processing a hard-to-understand example, or parsing difficult, jargon-laden, or overly terse text.

We used ***people***. In stories, examples, pictures, etc., because, well, because *you're* a person. And your brain pays more attention to *people* than it does to *things*.

We used an ***80/20*** approach. We assume that if you're going for a PhD in software design, this won't be your only book. So we don't talk about *everything*. Just the stuff you'll actually *need*.

Here's what YOU can do to bend your brain into submission

So, we did our part. The rest is up to you. These tips are a starting point; listen to your brain and figure out what works for you and what doesn't. Try new things.

Cut this out and stick it on your refrigerator.

1. **Slow down. The more you understand, the less you have to memorize.**

Don't just *read*. Stop and think. When the book asks you a question, don't just skip to the answer. Imagine that someone really *is* asking the question. The more deeply you force your brain to think, the better chance you have of learning and remembering.

2. **Do the exercises. Write your own notes.**

We put them in, but if we did them for you, that would be like having someone else do your workouts for you. And don't just *look* at the exercises. **Use a pencil.** There's plenty of evidence that physical activity *while* learning can increase the learning.

3. **Read the "There are No Dumb Questions"**

That means all of them. They're not optional side-bars—***they're part of the core content!*** Don't skip them.

4. **Make this the last thing you read before bed. Or at least the last *challenging* thing.**

Part of the learning (especially the transfer to long-term memory) happens *after* you put the book down. Your brain needs time on its own, to do more processing. If you put in something new during that processing time, some of what you just learned will be lost.

5. **Drink water. Lots of it.**

Your brain works best in a nice bath of fluid. Dehydration (which can happen before you ever feel thirsty) decreases cognitive function.

6. **Talk about it. Out loud.**

Speaking activates a different part of the brain. If you're trying to understand something, or increase your chance of remembering it later, say it out loud. Better still, try to explain it out loud to someone else. You'll learn more quickly, and you might uncover ideas you hadn't known were there when you were reading about it.

7. **Listen to your brain.**

Pay attention to whether your brain is getting overloaded. If you find yourself starting to skim the surface or forget what you just read, it's time for a break. Once you go past a certain point, you won't learn faster by trying to shove more in, and you might even hurt the process.

8. ***Feel* something!**

Your brain needs to know that this *matters*. Get involved with the stories. Make up your own captions for the photos. Groaning over a bad joke is *still* better than feeling nothing at all.

9. ***Design* something!**

Apply what you read to something new you're designing, or rework an older project. Just do *something* to get some experience beyond the exercises and activities in this book. All you need is a problem to solve... a problem that might benefit from one or more techniques that we talk about.

Read Me

This is a learning experience, not a reference book. We deliberately stripped out everything that might get in the way of learning whatever it is we're working on at that point in the book. And the first time through, you need to begin at the beginning, because the book makes assumptions about what you've already seen and learned.

We assume you are familiar with Java.

It would take an entire book to teach you Java (in fact, that's exactly what it took: *Head First Java*). We chose to focus this book on analysis and design, so the chapters are written with the assumption that you know the basics of Java. When intermediate or advanced concepts come up, they're taught as if they might be totally new to you, though.

If you're completely new to Java, or coming to this book from a C# or C++ background, we strongly recommend you turn to the back of the book and read Appendix II before going on. That appendix has some intro material that will help you start this book off on the right foot.

We only use Java 5 when we have to.

Java 5.0 introduces a lot of new features to the Java language, ranging from generics to parameterized types to enumerated types to the **`foreach`** looping construct. Since many professional programmers are just moving to Java 5, we didn't want you getting hung up on new syntax while you're trying to learn about OOA&D. In most cases, we stuck with pre-Java 5 syntax. The only exception is in Chapter 1, when we needed an enumerated type—and we explained enums in that section in some detail.

If you're new to Java 5, you should have no trouble with any of the code examples. If you're already comfortable with Java 5, then you will get a few compiler warnings about unchecked and unsafe operations, due to our lack of typed collections, but you should be able to update the code for Java 5 on your own quite easily.

The activities are NOT optional.

The exercises and activities are not add-ons; they're part of the core content of the book. Some of them are to help with memory, some are for understanding, and some will help you apply what you've learned. ***Don't skip the exercises.*** The crossword puzzles are the only things you don't *have* to do, but they're good for giving your brain a chance to think about the words and terms you've been learning in a different context.