Elogios a Use a Cabeça! Padrões de Projetos (Design Patterns)

"Recebi o livro ontem e comecei a ler no caminho para casa... e não consegui parar. Levei-o para a academia e imagino que as pessoas tenham me visto sorrindo bastante enquanto eu me exercitava e lia. O livro é muito legal. É divertido e ao mesmo tempo eles cobrem vários tópicos e vão direto ao assunto. Estou realmente impressionado."

- Erich Gamma, Engenheiro Laureado da IBM, e co-autor de Padrões de Projetos

"Use a Cabeça! Padrões de Projetos" consegue misturar diversão, grandes risadas, profundidade técnica e ótimos conselhos práticos em uma leitura divertida e inspiradora. Independentemente de ser um novato nos padrões de projetos, ou os ter usado por anos, você certamente sairá ganhando ao visitar a Objetolândia"

- Richard Helm, co-autor de "Padrões de Projetos" com o resto da Gang of Four
– Erich Gamma, Ralph Johnson e John Vlissides

"Me sinto como se uma pilha de mil livros tivesse acabado de ser tirada de cima da minha cabeça"

- Ward Cunningham, inventor do Wiki e fundador do Hillside Group

"Este livro chega quase à perfeição, por causa da maneira como ele combina o conhecimento e a facilidade de leitura. Ele fala com autoridade e é ótimo de ler. É um dos muito poucos livros sobre software que já li e considero indispensáveis. (Eu colocaria talvez 10 livros nessa categoria, no máximo.)"

- David Gelerneter, Professor de Ciência da Computação, Universidade de Yale e autor de " Mirror Worlds" e "Machine Beauty".

"Um mergulho de cabeça no reino dos padrões, um lugar onde coisas complexas se tornam simples, mas onde coisas simples podem também se tornar complexas. Não consigo pensar em guias melhores do que os Freemans.

- Miko Matsummura, Analista da Indústria, The Middleware Company; Ex-Pregador Chefe do Java, Sun Microsystems

"Eu ri, eu chorei, o livro me emocionou"

- Daniel Steinberg, Editor-em-Chefe, java.net

"Minha primeira reação foi rolar de rir. Depois de me recompor, percebi que o livro não somente é tecnicamente preciso, como também é a introdução aos padrões de projetos mais fácil de entender que eu já vi."

-Dr. Timothy A. Budd, Professor Associado de Ciência da Computação na Universidade Estadual de Oregon e autor de mais de uma dúzia de livro, incluindo "C++ para Programadores Java"

"Jerry Rice executa padrões melhor do que qualquer jogador da NFL, mas os Freemans acabaram de superá-lo. Mas falando sério... este é um dos livros mais engraçados e inteligentes sobre projetos de software que eu já li."

- Aaron LaBerge , VP Technology, ESPN.com

Mais elogios a Use a Cabeça! Padrões de Projetos

"Um ótimo projeto de código é, antes de mais nada, um ótimo projeto de informação. Um projetista de código está ensinando um computador como fazer algo, e não é surpresa que a editora seja uma grande professora de programadores. A clareza, o humor e as substanciais doses de inteligência deste livro o tornam o tipo que ajuda até mesmo não-programadores a pensar bem sobre como solucionar problemas."

- Cory Doctorow, co-editor de Boing Boing e autor de "Down and Out in the Magic Kingdom" e "Someone Comes to Town, Someone leaves Town"

"Há um velho ditado na indústria de computadores e videogames – bem, não pode ser tão velho porque a própria indústria não é velha assim – que diz mais ou menos o seguinte: O Projeto é Vida. O mais curioso sobre essa frase é que mesmo hoje em dia quase ninguém que trabalha com a arte de criar jogos eletrônicos consegue chegar a um acordo sobre o que significa "projetar" um jogo. O projetista é um engenheiro de software? Um diretor de arte? Um contador de histórias? Um arquiteto ou um pedreiro? Uma pessoa com os pés no chão ou um visionário? É possível um mesmo indivíduo ter algo de todos esses papéis? E, mais importante, quem liga para essa %$#&*?

Já foi dito que o crédito "projetado por" em entretenimento interativo é similar ao "dirigido por" na indústria dos filmes, o que de fato permite ao nosso crédito compartilhar DNA com aquele que é talvez o crédito mais controverso, exagerado, e freqüentemente sem nenhuma humildade já propagado na arte comercial. Boa companhia, hein? E mesmo assim, o Projeto é Vida, então talvez seja hora de despendermos algum tempo pensando sobre o que ele é. Eric e Elisabeth Freeman intrepidamente se ofereceram para explorar os mistérios do código para nós em *Use a Cabeça! Padrões de Projetos*. Não estou certo de que qualquer um deles se importe tanto assim com o PlayStation ou o X-Box, e nem eles deveriam. Mesmo assim, eles abordam a noção de projeto em um nível significativamente honesto, de forma tal que qualquer um procurando por elogios aos seus próprios talentos como escritor deve ficar avisado de que terá uma boa lição de humildade ao ler este livro".

- Ken Goldstein, Vice-Presidente Executivo & Diretor Gerente, Disney Online

"O livro tem o tom exato para o guru nerd e casualmente legal em todos nós. A referência correta para estratégias práticas de desenvolvimento – faz o meu cérebro funcionar sem ter de empurrar um monte de conversa velha e mofada de professor."

- Travis Kalanick, Fundador do Scour and Red Swoosh, membro do MIT TR100

"Este livro combina bom humor, ótimos exemplos e conhecimento aprofundado dos Padrões de Projetos de forma tal que torna o aprendizado divertido. Trabalhando na indústria de tecnologia de entretenimento, eu tenho dúvidas sobre o Princípio Hollywood e o Padrão Facade dos home theaters, entre outros. O entendimento dos Padrões de Projetos não apenas nos ajudam a criar bom software reutilizável e fácil de manter, mas também ajuda a melhorar as nossas habilidades de resolução de problemas em várias áreas. Este livro é indispensável para todos os profissionais e estudantes de computação."

**- Newton Lee, Fundador e Editor-Chefe, Association for Computing Machinery´s (ACM)
Computers int Entertainment (acmcie.org.)**

Elogios à abordagem Use a Cabeça!

"A tecnologia Java está em toda parte – em telefones celulares, câmeras, impressoras, jogos, PDAs, caixas eletrônicos, smart cards, bombas de gasolina, estádios esportivos, dispositivos médicos, web cams, servidores, entre outros. Se você desenvolve software e ainda não aprendeu Java, definitivamente está na hora de mergulhar – Use A Cabeça!."

– Scott McNealy, Presidente e CEO da Sun Microsystems

"É rápido, irreverente, divertido e envolvente. Cuidado – você pode acabar realmente aprendendo algo!"

– Ken Arnold, ex-Engenheiro Sênior da Sun Microsystems Co-autor (com James Gosling, criador do Java) de *The Java Programming Language*

"Use A Cabeça! Java é mais ou menos o que acontece quando o Monty Python encontra a gang of four... o texto é tão bem dividido entre quebra-cabeças e estórias, testes e exemplos, que você aprende com uma profundidade que nenhum outro livro sobre computação oferece."

– Douglas Rowe, Grupo de Usuários Java de Columbia

" 'Use A Cabeça! Java'... dá um novo significado à frase de marketing deles 'Há um O'Reilly para isso'. Eu escolhi este aqui porque várias pessoas que respeito o tinham descrito em termos como 'revolucionário' e ressaltando a abordagem radicalmente diferente do livro-texto. Elas estavam (estão) certas... De modo tipicamente O'Reilly, eles usaram uma abordagem científica e bem pensada. O resultado é engraçado, irreverente, dividido em tópicos, interativo e brilhante... Ler este livro é como estar no saguão de uma conferência técnica, aprendendo – e rindo – com os seus pares... Se você quiser ENTENDER Java, vá comprar este livro."

– Andrew Pollack, www.thenorth.corn

"Se você quer *aprender* Java, achou o que procurava: bem-vindo ao primeiro livro técnico com uma GUI! Este formato perfeitamente executado e inovador oferece benefícios que outros textos sobre Java simplesmente não conseguem... Prepare-se para uma viagem realmente notável pela Javalândia."

– Neil R. Bauman, Capitão e CEO, Geek Cruises (www.GeekCruises.corn)

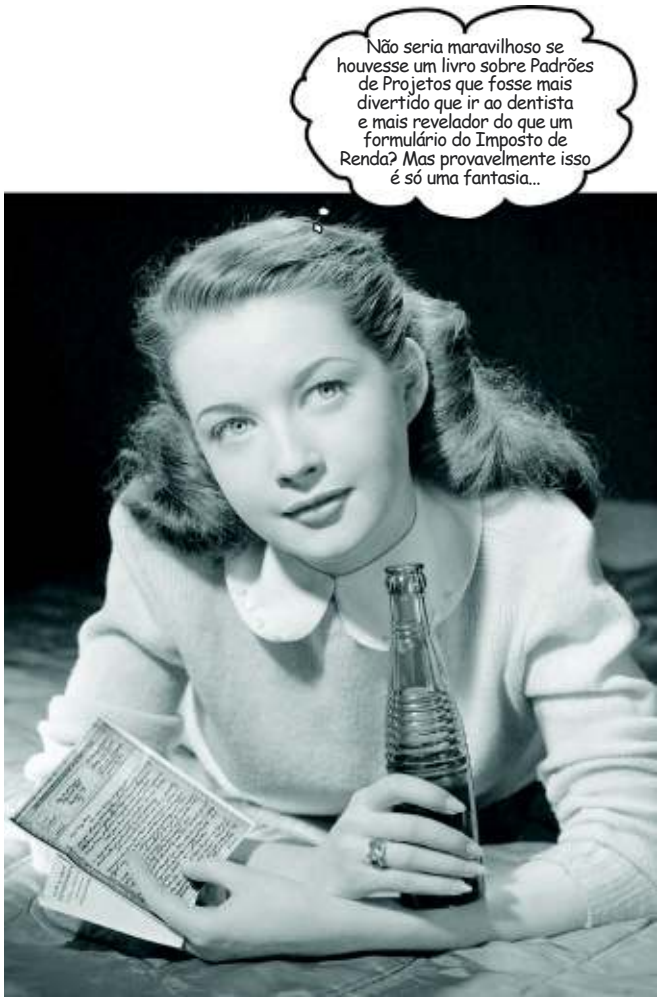
"Que fantástica maneira de aprender!!! NÃO CONSIGO FECHAR ESTE LIVRO!!! Meu filho de 3 anos acordou à 1h40 da madrugada e eu o coloquei de volta na cama com o livro e uma lanterna na mão, para poder continuar lendo por cerca de mais uma hora.

– Ross Goldberg

Use a Cabeça

Padrões e Projetos

2ª Edição Revisada



Eric Freeman
Elisabeth Freeman

com
Kathy Sierra
Bert Bates

ALTA BOOKS
EDITORA
Rio de Janeiro • 2009

Para a Gangue dos Quatro, cuja visão e conhecimentos em matéria de captura e comunicação sobre Padrões de Projetos mudou a face da concepção do software para sempre, e melhorado a vida dos desenvolvedores em todo o mundo.

Autores/Desenvolvedores de Use a Cabeça Padrões de Projetos

Eric Freeman é um cientista da computação apaixonado por arquiteturas de mídia e de software. Ele acabou de concluir quatro anos em um emprego dos sonhos – dirigindo trabalhos de Internet banda larga e wireless na Disney – e agora está voltado a programar, criando softwares legais e tirando o máximo do Java e Macs.

Eric passou boa parte dos anos 90 trabalhando em alternativas à metáfora do desktop com David Gelernter (e os dois *ainda* estão fazendo a pergunta "por que eu tenho de dar um nome ao meu arquivo?"). Com base nesse trabalho, Eric obteve um Ph.D. na Yale University em 97. Foi também co-fundador da Mirror Worlds Technologies (agora adquirida para criar uma versão comercial do trabalho da sua tese, o Lifestreams).

Em uma vida anterior, Eric criava software para redes e supercomputadores. Talvez você o conheça de livros como *JavaSpaces Principles Patterns and Practice*. Eric tem boas lembranças de implementar sistemas tuple-space em Thinking Machine CM-5s e de criar alguns dos primeiros sistemas de informação Internet para a NASA, no final dos anos 80.

Eric atualmente mora no deserto perto de Santa Fe. Quando não está escrevendo texto nem código, você o encontrará passando mais tempo ajustando seus programas do que assistindo TV no seu home theater ou tentando consertar um videogame do Dragon's Lair dos anos 80. Ele também não se importaria de passar a noite como um DJ de música eletrônica.Escreva para ele (em inglês) em eric@wickedlysmart.com ou visite o seu blog em http://www.ericfreeman.com.

Elisabeth Freeman é escritora, desenvolvedora de software e artista digital. Ela esteve envolvida com a Internet desde o princípio, sendo co-fundadora do The Ada Project (TAP), um premiado site voltado às mulheres que trabalham com computação, adotado agora pela ACM. Mais recentemente, Elisabeth liderou trabalhos de pesquisa e desenvolvimento em mídia digital na Walt Disney Company, onde co-inventou o Motion, um sistema de conteúdo que distribui terabytes de vídeo todos os dias aos usuários da Disney, da ESPN e de Movies.com.

Elisabeth é cientista da computação por formação e tem diplomas universitários em Ciência da Computação pela Yale University e pela Indiana University. Tem trabalhado em uma série de áreas, incluindo linguagens visuais, transmissões RSS e sistemas Internet. Tem sido também uma ativista pelas mulheres na computação, desenvolvendo programas que encorajam as mulheres a se envolverem na área. Hoje em dia, você a encontra bebendo café ou achocolatado ao lado do seu Mac, embora ela sonhe com o dia em que todo o mundo usará o Scheme.

Elisabeth sempre adorou caminhadas ao ar livre desde a sua infância na Escócia. Quando está ao ar livre, sua câmera está sempre ao seu alcance. Ela é também uma ávida ciclista, vegetariana e amante dos animais. Você pode enviar um e-mail para ela (em inglês) em beth@wickedlysmart.com.

Criadores da série Use a Cabeça! (e co-conspiradores neste livro)

Kathy se interessa por teoria da aprendizagem desde os seus dias como designer de jogos (ela escreveu jogos para a Virgin, a MGM e a Amblin). Ela desenvolveu a maior parte do formato Use a Cabeça! enquanto ensinava Autoração de Novas Mídias no programa de extensão Estudos de Entretenimento da UCLA. Mais recentemente, ela tem trabalhado como instrutora máster para a Sun Microsystems, ensinando aos instrutores Java da Sun sobre como ensinar as tecnologias Java mais recentes e desenvolvendo vários dos exames de certificação da Sun. Junto com Bert Bates, ela tem usado ativamente os conceitos Use a Cabeça! para ensinar a milhares de desenvolvedores. Kathy é a fundadora do javaranch.com, que recebeu em 2003 e 2004 o prêmio Jolt Cola Productivity Award, da revista Software Development. Também é possível encontrá-la dando aulas de certificação Java no site Java Jam Geek Cruise (geekcruises.com).

Recentemente se mudou da Califórnia para Boulder, Colorado, onde teve que aprender novas palavras, como "limpador de gelo" e "pele", mas pelo menos lá é possível esquiar. Gosta de: correr, esquiar, andar de skate, brincar com seu cavalo islandês e de ciência esquisita. Não gosta de: entropia. Um artigo de uma revista uma vez a descreveu como "persuasiva". Ela ainda está tentando descobrir se eles disseram isso no *bom* sentido. Se você souber, ou se tiver qualquer outro comentário, escreva para ela (em inglês) em kathy@wickedlysmart.com.

Bert é desenvolvedor e arquiteto de software, mas uma passagem de dez anos no campo de inteligência artificial levou-o a se interessar por teoria da apren-dizagem e pelo trei-namento baseado em tecnologia. Ele tem ensinado programação a clientes desde então. Recentemente, tornou-se membro da equipe de desenvolvimento de vários dos exames de Certificação Java da Sun, incluindo o novo SCBCD.

Ele passou a primeira década da sua carreira em software viajando pelo mundo para ajudar clientes broadcast como a Rádio Nova Zelândia, o Weather Channel e a rede Arts & Entertainment (A & E). Um dos seus projetos favoritos foi a criação de um sistema completo de simulação de malha ferroviária para a Union Pacific Railroad.

Bert é um jogador viciado de *go* de longa data e tem trabalhado em um programa de *go* por tempo demais. Ele é um guitarrista razoavelmente bom e agora está tentando aprender banjo. Escreva para ele (em inglês) em terrapin@wickedlysmart.com.

Tabela de Conteúdo (sumário)

Tabela de Conteúdo (de verdade)

Introdução

Sua mente concentrada nos Padrões de Projetos. Aqui *você* está tentando *aprender* algo, enquanto o seu *cérebro* está lhe fazendo um favor certificando-se de que o aprendizado não se *manterá*. Seu cérebro está pensando: "é melhor deixar espaço para coisas mais importantes, como quais animais selvagens evitar e fazer esqui na neve pelado é uma má idéia". Então, *como* você engana o seu cérebro para fazê-lo pensar que a sua vida depende de aprender os Padrões de Projetos?

Introdução aos Padrões de Projetos

1 Bem-vindo aos Padrões de Projetos

Alguém já resolveu os seus problemas. Neste capítulo, você aprenderá por que (e como) pode se aproveitar da sabedoria e das lições aprendidas por outros desenvolvedores que tenham passado pelo mesmo campo minado dos problemas de projetos e tenham sobrevivido à viagem. Antes de terminarmos, veremos o uso e os benefícios dos padrões de projetos, veremos alguns princípios chaves dos projetos OO, e o guiaremos através de um exemplo de como um padrão funciona. A melhor maneira de usar os padrões é *carregando-os no seu cérebro* e depois *reconhecendo lugares* de seu projeto e das suas aplicações existentes onde pode *aplicá-los*. Em vez de reutilização de código, com padrões você tem uma reutilização de *experiência*.

Lembre-se: conhecer conceitos como abstração, herança e polimorfismo não o torna um bom projetista orientado a objetos. Um guru dos projetos pensa sobre como criar projetos flexíveis que sejam fáceis de manter e modificar.

O Padrão Observer

2 Mantendo os seus Objetos informados

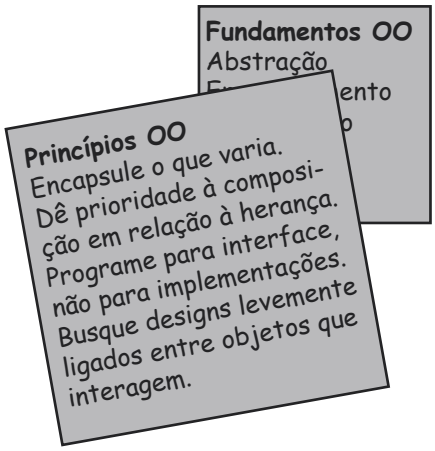
Não durma no ponto quando algo interessante acontecer! Temos um padrão que mantém os seus objetos informados quando algo que possa lhes interessar acontece. Os objetos podem até mesmo decidir, no momento da execução, se querem ser mantidos informados ou não. O Padrão Observer é um dos mais usados do JDK e é incrivelmente útil. Antes de terminarmos, veremos também as relações de um para muitos e o acoplamento solto (sim, isso mesmo, nós dissemos acoplamento). Com o Observer, você será a alma da Festa dos Padrões.

O Padrão Decorator

3 Decorando Objetos

Chame este capítulo de "Olho de design para o garoto da herança". Examinaremos novamente o típico uso excessivo da herança e você aprenderá a decorar suas classes no tempo de execução usando uma forma de composição de objeto. Por quê? Depois que você descobrir as técnicas de decoração, poderá dar novas responsabilidades a seus objetos (ou de outra pessoa) *sem fazer nenhuma mudança de código nas classes básicas*.

Eu achava que homens de verdade subclassificavam tudo. Isso até eu aprender o poder da extensão no tempo de execução, em vez de no tempo de compilação. Olhe só para mim agora!



O Padrão Factory

4 Cozinhando com a polpa da programação OO

Prepare-se para cozinhar alguns projetos OO com acoplamento solto. Para se criar objetos, é preciso mais do que apenas usar o operador **new**. Você aprenderá que a chamada é uma atividade que não deve ser feita sempre em público e pode freqüentemente levar a *problemas de acoplamento*. E você não quer que *isso* aconteça, quer? Descubra como os Padrões Factory podem ajudar a salvá-lo de dependências embaraçosas.

O Padrão Singleton

5 Objetos Únicos

O Padrão Singleton: o seu ingresso para criar objetos únicos, para os quais há apenas uma instância. Talvez você fique contente de saber que, de todos os padrões, o Singleton é o mais simples em termos do diagrama das suas classes; na verdade, o diagrama possui apenas uma classe! Mas não se sinta à vontade demais; apesar da sua simplicidade de um ponto de vista do design das classes, encontraremos algumas ondulações e alguns buracos na pista da sua implementação. Então aperte o cinto – este aqui não é tão simples quanto parece...

O Padrão Command

6 Encapsulando a Chamada de Métodos

Neste capítulo, levaremos o encapsulamento a um novo nível: vamos encapsular a *chamada de métodos*. É isso mesmo – quando encapsulamos a chamada de métodos, podemos cristalizar partes da computação para que o objeto que está invocando esse processo não precise saber como as coisas são feitas. Ele só precisa usar o método cristalizado para executar a tarefa. O encapsulamento das chamadas de métodos também nos permite fazer algumas coisas muito engenhosas, como salvar os métodos encapsulados para fins de registro ou reutilizá-los para implementar recursos de "desfazer" no nosso código.

Singleton – Verifique se uma classe tem apenas uma instância e forneça um ponto global de acesso a ela.

Os Padrões Adapter e Facade

7 Como ser Adaptável

Neste capítulo, iremos tentar feitos tão impossíveis quanto colocar um objeto quadrado em um buraco redondo. Parece impossível? Não quando temos os Padrões de Projetos. Lembra-se do Padrão Decorator? Nós **embrulhamos os objetos** para dar a eles novas responsabilidades. Agora iremos embrulhar alguns objetos com um propósito diferente: fazer com que as suas interfaces aparentem ser algo que não são. Por que faríamos algo assim? Para podermos adaptar um projeto que espera uma determinada interface a uma classe que implementa uma interface diferente. E não é só isso: enquanto estamos com a mão na massa, veremos também um outro padrão que embrulha objetos para simplificar as suas interfaces.

O Padrão Template

8 Encapsulando Algoritmos

Já encapsulamos a criação de objetos, métodos, interfaces complexas, patos, pizzas... o que virá depois? Vamos encapsular *algoritmos* para que as subclasses possam se conectar diretamente a uma computação a qualquer momento que quiserem. Iremos aprender até mesmo um princípio de projetos inspirado em Hollywood.

Os Padrões Iterator e Composite

9 Coleções bem gerenciadas

Existem várias maneiras de inserir objetos em uma coleção. Colocá-los em um Array, um Stack, uma Lista, um Mapa... você escolhe. Cada um tem as suas vantagens e desvantagens. Mas, quando o seu cliente quiser iterar através dos seus objetos, você vai deixá-lo ver a sua implementação? Certamente esperamos que não! Isso simplesmente *não* seria profissional. Não se preocupe – neste capítulo você verá como permitir aos seus clientes *iterar* através dos seus objetos sem jamais verem como você *armazena* os objetos. Você aprenderá também como criar *super coleções* de objetos capazes de saltar sobre algumas estruturas de dados impressionantes em uma única passada. Você aprenderá também algumas coisas sobre a responsabilidade dos objetos.

O Padrão State

10 O Estado das Coisas

Um fato pouco conhecido: os Padrões State e Strategy são irmãos gêmeos que foram separados ao nascer. Como você sabe, o Padrão Strategy acabou se dando muito bem no negócio dos algoritmos intercambiáveis. O Padrão State, por sua vez, tomou o caminho talvez mais nobre de ajudar objetos a controlarem seu comportamento através de mudanças no seu estado interno: “Repita comigo: eu sou bastante bom, eu sou bastante esperto, e isso é o que importa…”.

O Padrão Proxy

11 Controlando o Acesso de Objetos

Já brincou de tira bonzinho e tira malvado? Você é o tira bonzinho e fornece todos os seus serviços de forma agradável e amistosa, mas você não quer que *todo mundo* peça os seus serviços, então você faz o tira malvado *controlar o acesso* a você. É isso que os proxies fazem: controlam e gerenciam o acesso. Como você verá, existem *várias* maneiras pelas quais os proxies substituem os seus respectivos objetos. Os proxies têm a fama de carregar chamadas a métodos inteiros através da Internet para os objetos que substituem; eles também têm a fama de pacientemente substituir alguns objetos bem preguiçosos.

Padrões Compostos

12 Padrões de Padrões

Quem poderia adivinhar que os Padrões conseguem trabalhar juntos? Tendo presenciado as exaltadas Conversas ao Pé do Fogo (e você ainda nem viu as páginas do Combate Mortal entre Padrões, que o editor nos forçou a remover do livro), você certamente não imaginaria que ocasionalmente os padrões podem até conviver em paz. Bem, acredite ou não, alguns dos esquemas mais poderosos da programação orientada a objetos baseiam-se na utilização simultânea de vários padrões. Portanto, prepare-se para subir mais um degrau no seu domínio dos padrões: chegou a hora dos padrões compostos. Mas cuidado – os seus colegas podem ter que sacrificá-lo se você pegar a Febre dos Padrões

Vivendo Melhor com os Padrões

13 Padrões no Mundo Real

Ahh, agora você está pronto para um brilhante mundo novo cheio de Padrões de Projetos. Mas, antes de abrir todas essas novas portas de oportunidades, precisamos abordar alguns detalhes que você encontrará no mundo real – as coisas ficam um pouco mais complexas *lá fora* do que elas são na Objetolândia. Siga-nos, temos um bom guia para ajudá-lo na transição...

14 Apêndice: Padrões Restantes 459

Nem todos podem ser os mais populares. Muitas coisas mudaram nos últimos dez anos. Desde que saiu a primeira edição de *Padrões de Projetos: Elementos de Software Reutilizado Orientado a Objetos*, os desenvolvedores vêm aplicando esses padrões milhares de vezes. Os padrões que resumimos neste apêndice são padrões GoF (padrões fundamentais originais) oficiais completos e de carteirinha, mas não são sempre usados com a mesma freqüência que os padrões que exploramos até aqui. Mas os padrões deste apêndice são eles mesmos excelentes e, se a situação exigir, você deve ser capaz de aplicá-los com orgulho. O nosso objetivo neste apêndice é lhe dar uma idéia geral do que são estes padrões.

Outros títulos da Série Use a Cabeça!

Use a Cabeça Java
Use a Cabeça Análise & Projeto Orientado a Objetos (A&POO)
Use a Cabeça Ajax Iniciação Rápida
Use a Cabeça HTML com CSS e XHTML
Use a Cabeça Servlets e JSP
Use a Cabeça PMP
Use a Cabeça SQL
Use a Cabeça Desenvolvimento de Software
Use a Cabeça JavaScript
Use a Cabeça C#
Use a Cabeça PHP & MySQL (2009)
Use a Cabeça Física (2009)
Use a Cabeça Algebra (2009)
Use a Cabeça Ajax Profissional (2009)
Use a Cabeça Estatística
Use a Cabeça Ruby on Rails

Introdução

Nesta seção, respondemos à pergunta que não quer calar: Então, POR QUE eles colocaram isso em um livro sobre padrões de projetos?.

A quem se destina este livro?

Você provavelmente se dará bem se souber C# em vez de Java.

Se você responder "sim" a todas estas perguntas:

- Você sabe algo de Java? (Não é necessário ser um especialista.)
- Você deseja aprender, entender, se lembrar e aplicar os padrões de projetos, incluindo os princípios de design OO sobre os quais os padrões se baseiam?
- Você prefere uma conversa informal estimulante a aulas acadêmicas secas e chatas?

então este livro é para você.

Quem provavelmente deve evitar este livro?

Se você responder "sim" a qualquer uma destas perguntas:

- Você é totalmente iniciante em Java?
  (Não é preciso ser um conhecedor avançado e, mesmo se você não souber Java mas souber C#, provavelmente entenderá pelo menos 80% dos exemplos de código. Também é possível que você se dê bem tendo um background apenas em C++.)
- Você é um designer/desenvolvedor OO experiente procurando um livro de referência?
- Você é um arquiteto procurando por padrões de projetos corporativos?
- Você tem medo de experimentar algo novo? Você prefere fazer um tratamento de canal a misturar uma camisa verde com uma gravata cor-de-rosa? Você não acredita que um livro técnico no qual os componentes são antropomorfizados possa ser sério?

então este livro não é para você.

nota do departamento de marketing: este livro é adequado para qualquer um que tenha um cartão de crédito.

Sabemos o que você está pensando.

"Como isso aqui pode ser um livro de programação sério?"
"Pra que tantos desenhos?"
"É realmente possível aprender desta forma?"

E sabemos o que o seu cérebro está pensando.

seu cérebro acha que ISTO é importante.

O seu cérebro tem sede de novidades. Ele está sempre buscando, atento, esperando por algo fora do comum. Ele foi feito assim, e isso o ajuda a se manter vivo.

Hoje em dia, é menos provável que você acabe virando o lanche de algum tigre. Mas o seu cérebro continua em alerta. Nunca se sabe.

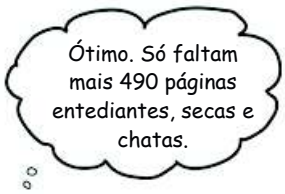
Então o que o seu cérebro faz com todas as coisas rotineiras, comuns e normais que encontra? Tudo o que puder para impedir que elas interfiram com verdadeiro trabalho do cérebro – registrar as coisas que são importantes. Ele não se dá ao trabalho de armazenar as coisas chatas; elas nunca passam pelo filtro "isto obviamente não é importante".

Como o seu cérebro sabe o que é importante? Suponha que você saia para uma caminhada e um tigre pule na sua frente, o que acontece dentro da sua cabeça e do seu corpo?

Os neurônios disparam. As emoções aumentam. Os elementos químicos borbulham.

E é assim que o seu cérebro sabe que...

Isto deve ser importante! Não se esqueça disto!



Mas imagine que você esteja em casa, ou em uma biblioteca. É uma área segura, aconchegante e sem tigres. Você está estudando. Preparando-se para um exame. Ou tentando aprender algum tópico técnico difícil que o seu chefe acha que só vai levar uma semana, dez dias no máximo.

Apenas um problema. O seu cérebro está tentando lhe fazer um grande favor. Está tentando certificar-se de que este conteúdo obviamente sem importância não gaste recursos escassos. Recursos esses que seriam melhor gastos armazenando as coisas realmente importantes. Como tigres. Como o perigo do fogo. Como você não deve nunca mais praticar esqui na neve de calção.

E não existe nenhuma maneira simples de dizer ao seu cérebro: "ei, cérebro, muito obrigado, mas não importa o quanto este livro seja chato, nem quanto eu estou marcando na escala Richter emocional neste momento, eu realmente quero armazenar isto aqui."

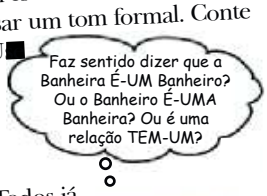
Nós pensamos no leitor de "Use a Cabeça!" como um aprendiz.

Então, o que é necessário para se aprender algo? Primeiro você tem que entender, depois certificar-se de que não vai esquecer. Não se trata de empurrar fatos para dentro da sua cabeça. Conforme as pesquisas mais recentes sobre ciência cognitiva, neurobiologia e psicologia da educação, aprender exige muito mais do que texto em uma página. Nós sabemos o que interessa ao seu cérebro.

Alguns dos princípios do aprendizado Use a Cabeça!:

Use o visual. Imagens são muito mais memoráveis do que apenas as palavras e tornam o aprendizado muito mais efetivo (até 89% de melhoria em estudos de lembrança e transferência). Também torna as coisas mais fáceis de entender. Coloque as palavras dentro ou perto dos gráficos a que elas se referem, em vez de no final ou em outra página, e a probabilidade de os aprendizes conseguirem resolver problemas relacionados ao conteúdo poderá dobrar.

Use um estilo conversacional e personalizado. Em estudos recentes, os estudantes tiveram uma melhora de até 40% no desempenho em testes pós-aprendizado quando o conteúdo falava diretamente ao leitor, usando um estilo conversacional, em primeira pessoa, em vez de se usar um tom formal. Conte histórias em vez de dar aulas. Use linguagem casual. Não se leve a sério demais. Em que você prestaria mais atenção: em uma conversa interessante durante um jantar ou em uma aula?



Consiga – e mantenha – a atenção do leitor. Todos já tivemos a experiência "eu realmente quero aprender isto mas não consigo passar da página um acordado". O seu cérebro presta atenção em coisas que saiam do comum, que sejam interessantes, estranhas, impressionantes, inesperadas. Aprender um novo e difícil assunto técnico não precisa ser enfadonho. O seu cérebro aprenderá muito mais rápido se não for.

Envolva as emoções do leitor. Nós sabemos, hoje em dia, que a sua capacidade de lembrar-se de algo depende muito do conteúdo emocional. Você se lembra das coisas com as quais se importa. Você se lembra quando sente algo. Não, não estamos falando de histórias de partir o coração sobre um garoto e o seu cachorro. Estamos falando de emoções como surpresa, curiosidade, diversão, "o que é isso?", e da sensação de "Eu sou demais!" que vem quando você resolve um quebra-cabeça, aprende algo que as demais pessoas acham difícil ou percebe que você sabe algo que o Bob "eu sou mais técnico do que vós" do departamento de engenharia não sabe.

Metacognição: pensando sobre o pensamento

Se você realmente deseja aprender, e quer aprender mais rápido e mais profundamente, preste atenção em como você presta atenção. Pense sobre como você pensa. Aprenda sobre como você aprende.

Muitos de nós não tivemos aulas sobre metacognição ou sobre teoria da aprendizagem quando éramos estudantes. Esperava-se que nós aprendêssemos, mas raramente nos ensinavam a aprender.

Mas nós presumimos que, se você está lendo este livro, é porque realmente deseja aprender os padrões de projetos. E você provavelmente não deseja gastar muito tempo. E quer se lembrar do que lê e ser capaz de aplicá-lo. E, para isso, você tem que entender o que leu. Para tirar o máximo proveito deste livro, ou de qualquer livro ou experiência de aprendizado, tome as rédeas do seu cérebro. Faça-o se concentrar no conteúdo.

O segredo é fazer o seu cérebro ver o novo material que você está aprendendo como Realmente Importante. Crucial para o seu bem-estar. Tão importante quanto um tigre. Caso contrário, você acabará em uma batalha constante com o seu cérebro fazendo o melhor que puder para impedir que o novo conteúdo seja registrado.

Então COMO você faz o seu cérebro pensar que os Padrões de Projetos são tão importantes quanto um tigre?

Existe a maneira lenta e tediosa e a maneira mais rápida e mais efetiva. A maneira lenta refere-se à repetição exaustiva. Você obviamente sabe que é capaz de aprender e memorizar até mesmo o assunto mais enfadonho, se se mantiver batendo na mesma tecla. Repetindo o suficiente, o seu cérebro diz: "Isto não parece importante, mas ele fica olhando a mesma coisa repetidas e repetidas vezes, então eu suponho que deve ser."

A maneira mais rápida é fazer qualquer coisa que aumente a atividade cerebral, especialmente aumentando diferentes tipos de atividades cerebrais. As coisas que você viu na página anterior são uma grande parte da solução, e elas são todas coisas que comprovadamente ajudam o seu cérebro a trabalhar em seu favor. Por exemplo, estudos mostram que colocar palavras dentro das figuras que elas descrevem (em vez de em algum outro lugar da página, como em uma legenda ou no corpo do texto) faz o seu cérebro tentar perceber em que sentido as palavras e a figura se relacionam, e isso faz disparar mais neurônios. Mais neurônios disparados = maior a chance de o seu cérebro perceber que isto é algo que vale a pena prestar atenção e possivelmente armazenar.

Um estilo conversacional ajuda porque as pessoas tendem a prestar mais atenção quando percebem que estão em uma conversa, porque nessa situação espera-se que elas sigam o que está sendo dito e mantenham o seu lado da conexão ativo. O incrível é que o seu cérebro não necessariamente se importa se a "conversa" é entre você e um livro! Por outro lado, se o estilo de escrita for formal e seco, o seu cérebro o percebe da mesma forma como percebe quando você está ouvindo uma aula, sentado em uma sala cheia de alunos passivos. Não é preciso ficar acordado.

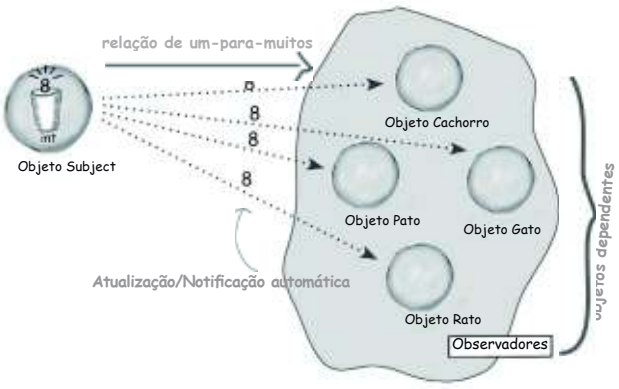
Mas figuras e estilo conversacional são apenas o início.

Eis o que NÓS fizemos:

Usamos figuras, porque o seu cérebro está ajustado para imagens, e não texto. No que diz respeito ao seu cérebro, uma imagem realmente vale por 1.024 palavras. E, nos casos em que texto e imagens trabalham juntos, nós inserimos o texto nas imagens, porque o seu cérebro trabalha de modo mais eficaz quando o texto se encontra dentro da coisa a que se refere, em vez de em uma legenda ou enterrado em alguma parte do texto.

Usamos a redundância, dizendo a mesma coisa de diferentes formas e com diferentes tipos de mídia, e com múltiplos sentidos, para aumentar a chance de o conteúdo ser codificado em mais de uma área do seu cérebro.

Usamos conceitos e imagens de formas inesperadas, porque o seu cérebro está ajustado para captar novidades, e usamos imagens e idéias com pelo menos algum conteúdo emocional, porque o seu cérebro está ajustado para prestar atenção na bioquímica das emoções. Aquilo que o faz sentir algo tem maior probabilidade de ser lembrado, mesmo se o sentimento não passar de um pouco de humor, surpresa ou interesse.



Usamos um estilo personalizado, conversacional, porque o seu cérebro está ajustado para prestar mais atenção quando acredita que você está em uma conversação do que quando pensa que você está ouvindo passivamente uma apresentação. Seu cérebro faz isso mesmo quando você está lendo.

Incluímos mais de 40 atividades, porque o seu cérebro está ajustado para aprender e memorizar mais quando você faz coisas do que quando você lê sobre as coisas. E fizemos os exercícios desafiadores, porém solucionáveis, porque é isso que a maioria das pessoas prefere.

O Guru dos Padrões

Usamos múltiplos estilos de aprendizado, porque você poderia preferir procedimentos passo-a-passo, enquanto outra pessoa pode desejar entender o geral primeiro, e uma outra poderia apenas querer ver um exemplo de código. Porém, independentemente da sua preferência em termos de aprendizado, todos se beneficiarão de ver o mesmo conteúdo representado em múltiplas maneiras.

Incluímos conteúdo para ambos os hemisférios do cérebro, porque, quanto mais do seu cérebro você ativa, maior a probabilidade de você aprender e memorizar, e maior o tempo durante o qual você pode se manter concentrado. Uma vez que trabalhar apenas um lado do cérebro freqüentemente significa dar ao outro lado uma chance para descansar, você pode ter maior produtividade no aprendizado por um período de tempo maior.

Pontos Importantes

E incluímos histórias e exercícios que apresentam mais de um ponto de vista, porque o seu cérebro está ajustado para aprender com mais profundidade quando é forçado a fazer avaliações e julgamentos.

Incluímos desafios, com exercícios e fazendo perguntas que nem sempre têm uma resposta direta, porque o seu cérebro está ajustado para aprender e memorizar quando precisa trabalhar em algo. Pense no seguinte: não é possível ficar em forma apenas olhando as pessoas na academia de ginástica. Mas fizemos o nosso melhor para nos certificarmos de que você estará trabalhando duro, e nas coisas certas. Que você não está gastando uma dendrite extra para processar um exemplo difícil de entender, ou "traduzindo" um texto difícil, cheio de jargão ou exageradamente conciso.

Quebra-cabeças

Usamos pessoas em histórias, exemplos, figuras etc., porque, bem, porque você é uma pessoa. E o seu cérebro presta mais atenção em pessoas do que em coisas.

Usamos uma abordagem 80/20. Partimos do princípio de que, caso você vá tentar um PhD em projetos de software, este não vai ser o seu único livro. Assim, não falamos sobre tudo. Apenas sobre as coisas que você de fato irá usar.

Recorte isto e cole na porta da sua geladeira

Eis o que VOCÊ pode fazer para obrigar seu cérebro a obedecê-lo

OK, fizemos a nossa parte. O resto depende de você. Estas dicas são um ponto de partida; escute o seu cérebro e descubra o que funciona para você e o que não funciona. Experimente coisas novas.

Vá com calma. Quanto mais você entende, menos você precisa memorizar.
Não leia, apenas. Pare e pense. Quando o livro lhe fizer uma pergunta, não pule direto para a resposta. Imagine que alguém está realmente fazendo lhe a pergunta. Quanto mais profundamente você força seu cérebro a pensar, maior a sua chance de aprender e memorizar.

Faça os exercícios. Escreva as suas próprias notas.
Nós os incluímos, mas, se os resolvêssemos para você, seria como ter alguém fazendo os seus abdominais para você. E não olhe os exercícios simplesmente. Use um lápis. Há muita evidência de que atividade física durante o aprendizado pode melhorá-lo.

Leia as seções "Perguntar não ofende"/"Não existem perguntas idiotas".
Isso significa todas elas. Elas não são enfeites opcionais – fazem parte do conteúdo central! Não as pule.

Faça deste livro a sua última leitura antes de dormir. Ou pelo menos a última leitura desafiadora.
Parte do aprendizado (especialmente a transferência para a memória de longo prazo) acontece depois que você fecha o livro. O seu cérebro precisa tirar um tempo para fazer mais processamentos. Se você acrescentar algo durante esse tempo de processamento, parte do que você acabou de aprender se perderá.

Beba água. Muita água.
O seu cérebro funciona melhor envolvido por bastante fluido. A desidratação (que pode ocorrer antes de você sequer sentir sede) diminui a capacidade cognitiva.

Fale sobre o assunto. Em voz alta.
Falar ativa uma parte diferente do cérebro. Se estiver tentando entender algo, ou aumentar a sua chance de se lembrar do assunto depois, pronuncie-o em voz alta. Melhor ainda, tente explicá-lo em voz alta para alguém. Você aprenderá mais rapidamente e poderá descobrir idéias que você nem suspeitava que estavam lá antes de começar a ler.

Ouça o seu cérebro.
Preste atenção se o seu cérebro está ficando sobrecarregado. Se você se pegar começando a ler superficialmente ou esquecendo o que acabou de ler, é hora de fazer uma pausa. Depois de passar de um certo ponto, você não vai aprender mais rápido tentando inserir mais informações e poderá até prejudicar o processo.

Sinta algo!
O seu cérebro precisa saber que isto é importante. Envolva-se com as estórias. Crie as suas próprias legendas para as fotos. Resmungar sobre uma piada ruim ainda é melhor do que não sentir nada.

Projete algo!
Aplique o conteúdo a algo que você esteja projetando ou refatore um projeto antigo. Apenas faça algo para obter alguma experiência além dos exercícios e atividades deste livro. Tudo de que você precisa é um lápis e um problema para resolver... um problema que possa se beneficiar de um ou mais padrões de projetos.

Leia-me

Isto é uma experiência de aprendizagem, não um livro de referência. Nós deliberadamente retiramos tudo o que poderia atrapalhar o aprendizado do que quer que estivéssemos apresentando na parte do livro em questão. E, na primeira leitura, você precisa começar do começo, porque o livro faz assunções sobre o que você já viu e aprendeu.

Usamos diagramas simples semelhantes a UML.

Embora seja bastante provável que você conheça a UML, não a abordamos no livro, nem é um pré-requisito para a sua leitura. Caso você nunca tenha visto UML antes, não se preocupe, daremos a você algumas dicas no caminho. Assim, em outras palavras, você não terá que se preocupar com Padrões de Projetos e UML ao mesmo tempo. Os nossos diagramas são "pseudo-UML" – embora tentemos ser fiéis à UML, em alguns momentos iremos distorcer as regras um pouco, normalmente por causa das nossas próprias razões artísticas egoístas.

Não abordamos todos os Padrões de Projetos já criados.

Existem muitos Padrões de Projetos: os padrões fundamentais originais (conhecidos como padrões GoF), os padrões J2EE da Sun, os padrões JSP, padrões arquiteturais, padrões de projetos de jogos e muitos mais. Mas o nosso objetivo foi nos certificarmos de que o livro pesaria menos do que a pessoa que o esteja lendo, então não abordamos todos eles aqui. O nosso foco é nos padrões centrais importantes dentre os padrões GoF originais e em nos certificarmos de que você

realmente, sinceramente, profundamente entenda como e quando usá-los. Você encontrará uma breve menção a alguns dos outros padrões (os muito menos prováveis que você vá usar) no apêndice. Em todo caso, depois de terminar o Use a Cabeça! Padrões de Projetos, você será capaz de pegar qualquer catálogo de padrões e entrar em ação rapidamente.

As atividades NÃO são opcionais.

Os exercícios e as atividades não são adicionais; eles fazem parte do conteúdo central do livro. Alguns deles servem para ajudar a memorizar, outros a entender, e alguns para ajudá-lo a aplicar o que você aprendeu. Não pule os exercícios. As palavras cruzadas são as únicas coisas que você não precisa fazer, mas são boas para dar ao seu cérebro uma chance de pensar sobre as palavras a partir de um contexto diferente.

Usamos a palavra "composição" no sentido OO geral, que é mais flexível do que o uso estrito de "composição" em UML.

Quando dizemos que "um objeto é composto com outro objeto", queremos dizer que eles se relacionam de forma TEM-UM. O nosso uso reflete o uso tradicional do termo e é o mesmo empregado no texto GoF (você aprenderá o que é isso mais tarde). Mais recentemente, a UML refinou esse termo em vários tipos de composição. Se for um expert em UML, ainda assim você poderá ler o livro e deverá ser capaz de mapear facilmente o uso de composição para os termos mais refinados, à medida que for lendo.

A redundância é intencional e importante.

Uma diferença distinta em um livro Use a Cabeça! é que queremos que você realmente entenda. E queremos que você termine o livro lembrando-se do que aprendeu. A maioria dos livros de referência não têm a retenção e a lembrança como objetivo, mas este livro é sobre aprendizado, então você verá alguns conceitos aparecerem mais de uma vez.

Os exemplos de códigos são os mais enxutos possíveis.

Os nossos leitores nos dizem que é frustrante percorrer 200 linhas de código procurando as duas linhas que eles precisam entender. A maioria dos exemplos deste livro é mostrada dentro do menor contexto possível, para que a parte que você está tentando aprender esteja clara e simples. Não espere que todo o código seja robusto, ou mesmo completo – os exemplos são escritos especificamente para o aprendizado, e não são sempre totalmente funcionais.

Em alguns casos, não incluímos todos os comandos import necessários, mas presumimos que, se você é um programador de Java, você saberá que ArrayList se encontra em java.util, por exemplo. Quando o conteúdo importado não fizer parte da API J2SE normal, nós o mencionaremos. Também colocamos todo o código-fonte na web, para que você possa baixá-lo. Você o encontrará em http://www.altabooks.com.br

Além disso, para nos concentramos na questão do aprendizado do código, não colocamos as nossas classes em pacotes (em outras palavras, elas estão todas no pacote padrão Java). Não recomendamos isso no mundo real, e, quando você baixar os exemplos de códigos deste livro, verá que todas as classes estão em pacotes.

Os exercícios "Poder da Mente" não têm respostas.

Para alguns deles, não há uma resposta certa; para outros, parte da experiência de aprendizado das atividades Poder da Mente requer que você decida se e quando as suas respostas estão corretas. Em alguns dos exercícios Poder da Mente, você encontrará dicas para lhe apontar a direção correta.

Revisores Técnicos

Em memória de Philippe Maquet
1960-2004

O seu impressionante conhecimento técnico, seu entusiasmo inacabável e a sua profunda preocupação com o leitor sempre irão nos inspirar.

Nunca te esqueceremos.

Philippe Maquet

Agradecimentos

Na O'Reilly:

O nosso muito obrigado a Mike Loukides da O'Reilly, por ter começado tudo e ajudado a dar forma ao conceito Use a Cabeça!, transformando-o em uma série. E um grande obrigado ao motor por trás da série Use a Cabeça!, Tim O'Reilly. Obrigado à inteligente "mamãe da série" Use a Cabeça! Kyle Hart, à estrela do rock and roll Ellie Volkhausen pelo seu inspirado desenho da capa e também a Colleen Gorman pela sua infalível preparação dos originais. Finalmente, obrigado a Mike Hendrickson por "vestir a camisa" deste livro de Padrões de Projetos e por ter criado a equipe.

Os nossos intrépidos revisores:

Somos extremamente gratos ao nosso diretor de revisão técnica Johannes deJong. Você é o nosso herói, Johannes. E apreciamos profundamente as contribuições do co-diretor da equipe de revisão Javaranch, o saudoso Philippe Maquet. Você sozinho iluminou a vida de milhares de desenvolvedores, e o impacto que teve nas suas (e nas nossas) vidas será eterno.
Jef Cumps é assustadoramente bom em encontrar problemas nos esboços dos nossos capítulos e, mais uma vez, fez uma enorme diferença para o livro. Obrigado, Jef! Valentin Cretazz (da AOP), que esteve conosco desde o primeiro livro Use a Cabeça!, provou (como sempre) o quanto nós realmente precisamos do seu conhecimento técnico e da sua inspiração. Você é o cara, Valentin (mas jogue fora a gravata).
Dois recém-chegados à equipe de revisão UC, Barney Marispini e Ike Van Atta, fizeram um trabalho primoroso no livro – vocês nos deram um feedback realmente crucial. Obrigado por se juntarem à equipe.
Também obtivemos excelente ajuda técnica dos moderadores/gurus do Javaranch Mark Spritzler, Jason Menard, Dirk Schreckmann, Thomas Paul e Margarita Isaeva. E, como sempre, obrigado especialmente ao chefão do javaranch.com, Paul Wheaton.
Obrigado aos finalistas do concurso "Escolha a Capa do Use a Cabeça! Padrões de Design". O vencedor, Si Brewster, enviou o ensaio vencedor que nos convenceu a escolher a mulher que você vê na nossa capa. Outros finalistas foram Andrew Esse, Gian Franco Casula, Helen Crosbie, Pho Tek, Helen Thomas, Sateesh Kommineni e Jeff Fisher.

*Ainda mais gente**

de Eric e Elisabeth

Escrever um livro da série Use a Cabeça! é um passeio emocionante com dois guias incríveis: Kathy Sierra e Bert Bates. Com Kathy e Bert você joga fora todas as convenções de escrita de livros e adentra um mundo cheio de histórias para contar, teoria de aprendizagem, ciência cognitiva e cultura pop, onde o leitor é sempre quem manda. Obrigado a ambos por nos deixar entrar no seu mundo maravilhoso; esperamos ter feito justiça à série Use a Cabeça!. Falando sério, foi uma experiência incrível. Obrigado por toda a sua cuidadosa orientação, nos levando sempre adiante e, acima de tudo, por confiar a nós o bebê de vocês. Vocês dois são certamente "incrivelmente espertos" e também são as pessoas de 29 anos mais legais que conhecemos. Então... quem mais?
Um muito obrigado a Mike Loukides e a Mike Hendrickson. Mike L. esteve conosco a cada passo do caminho. Mike, o seu inspirado feedback nos ajudou a dar forma ao livro e o seu encorajamento nos manteve seguindo em frente. Mike H., obrigado pela sua persistência ao longo de cinco anos tentando nos fazer escrever um livro sobre padrões; finalmente o fizemos e estamos felizes de termos esperado pela série Use a Cabeça!.
Um obrigado muito especial a Erich Gamma, que fez muito mais do que o seu dever ao revisar este livro (ele até levou um esboço consigo nas suas próprias férias). Erich, o seu interesse neste livro nos inspirou e a sua revisão técnica completa o melhorou imensamente. Obrigado também a toda a Gang of Four pelo seu suporte e interesse, e por fazer uma aparição especial na Objetolândia. Estamos em débito também com Ward Cunningham e com a comunidade dos padrões, que criaram o Portland Pattern Repository (Repositório de Padrões de Portland) – um recurso indispensável para nós ao escrevermos este livro.
É preciso toda uma cidade para escrever um livro técnico: Bill Pugh e Ken Arnold nos forneceram aconselhamento especializado sobre o Singleton. Joshua Marinacci forneceu dicas e aconselhamentos inestimáveis. O trabalho "Why a Duck?" ("Por que um Pato?") de John Brewer inspirou o SimUDuck (e ficamos felizes por ele gostar de patos também). Dan Friedman inspirou o exemplo Little Singleton. Daniel Steinberg agiu como a nossa "ligação técnica" e a nossa rede de suporte emocional. E obrigado a James Dempsey da Apple por nos permitir usar a sua canção sobre o MVC.
Finalmente, um obrigado pessoal à equipe de revisão do Javaranch pelas suas excelentes revisões e o seu suporte caloroso. Há mais de vocês neste livro do que vocês pensam.

De Kathy e Bert

Gostaríamos de agradecer a Mike Hendrickson por encontrar Eric e Elisabeth... mas não podemos. Por causa desses dois, nós descobrimos (para o nosso horror) que não somos os únicos capazes de escrever um livro Use a Cabeça!.;) Porém, se os leitores quiserem acreditar que foram realmente a Kathy e o Bert que fizeram as coisas legais do livro, bem, quem somos nós para corrigir os leitores?

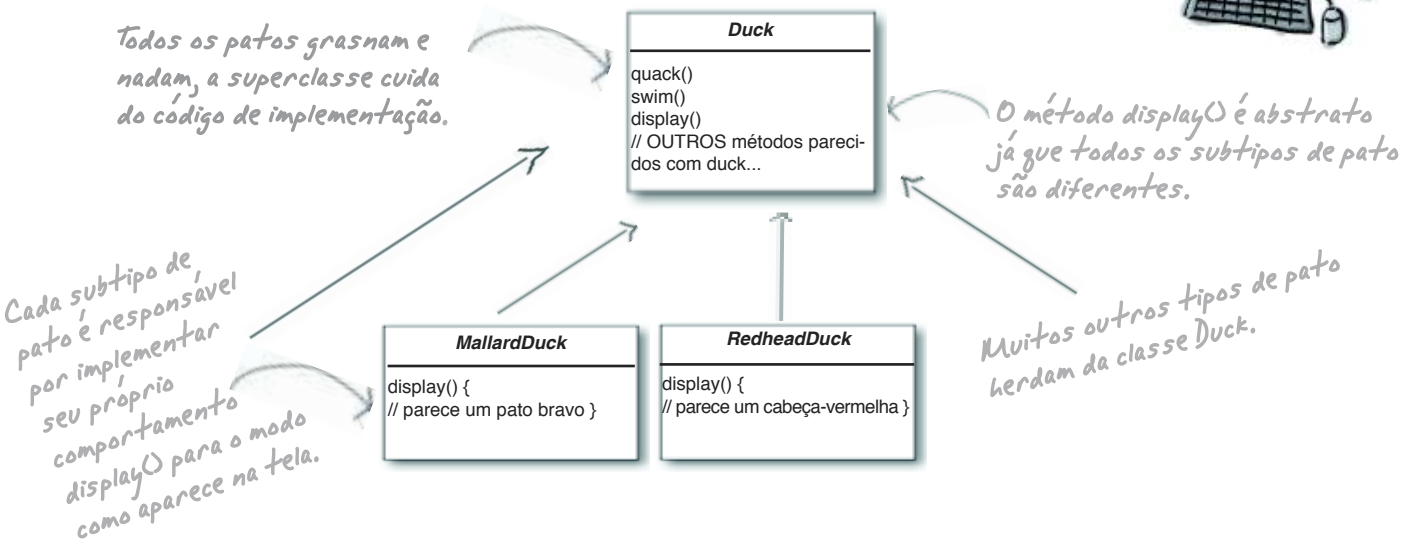
*O grande número de agradecimentos se deve ao fato de estarmos testando a teoria de que todas as pessoas mencionadas em uma lista de agradecimentos de um livro comprarão pelo menos uma cópia, e provavelmente mais, com os parentes e tudo o mais. Se você quiser estar na lista de agradecimentos do nosso próximo livro e tiver uma grande família, escreva para nós.

Bem-vindo aos Padrões de Projetos

Alguém resolveu seus problemas. Neste capítulo, você saberá por que (e como) pode explorar o conhecimento e as lições aprendidas por outros desenvolvedores que tiveram o mesmo problema de design e sobreviveram. Antes de terminarmos, vamos analisar o uso e os benefícios dos padrões de projetos, ver alguns princípios importantes de design OO e dar um exemplo de como um padrão funciona. A melhor maneira de usar os padrões é carregar seu cérebro com eles e depois reconhecer locais em seus designs e aplicativos existentes onde eles possam ser aplicados. Em vez da reutilização de código, com os padrões você obtém a reutilização de experiência.

Começou com um simples aplicativo SimUDuck

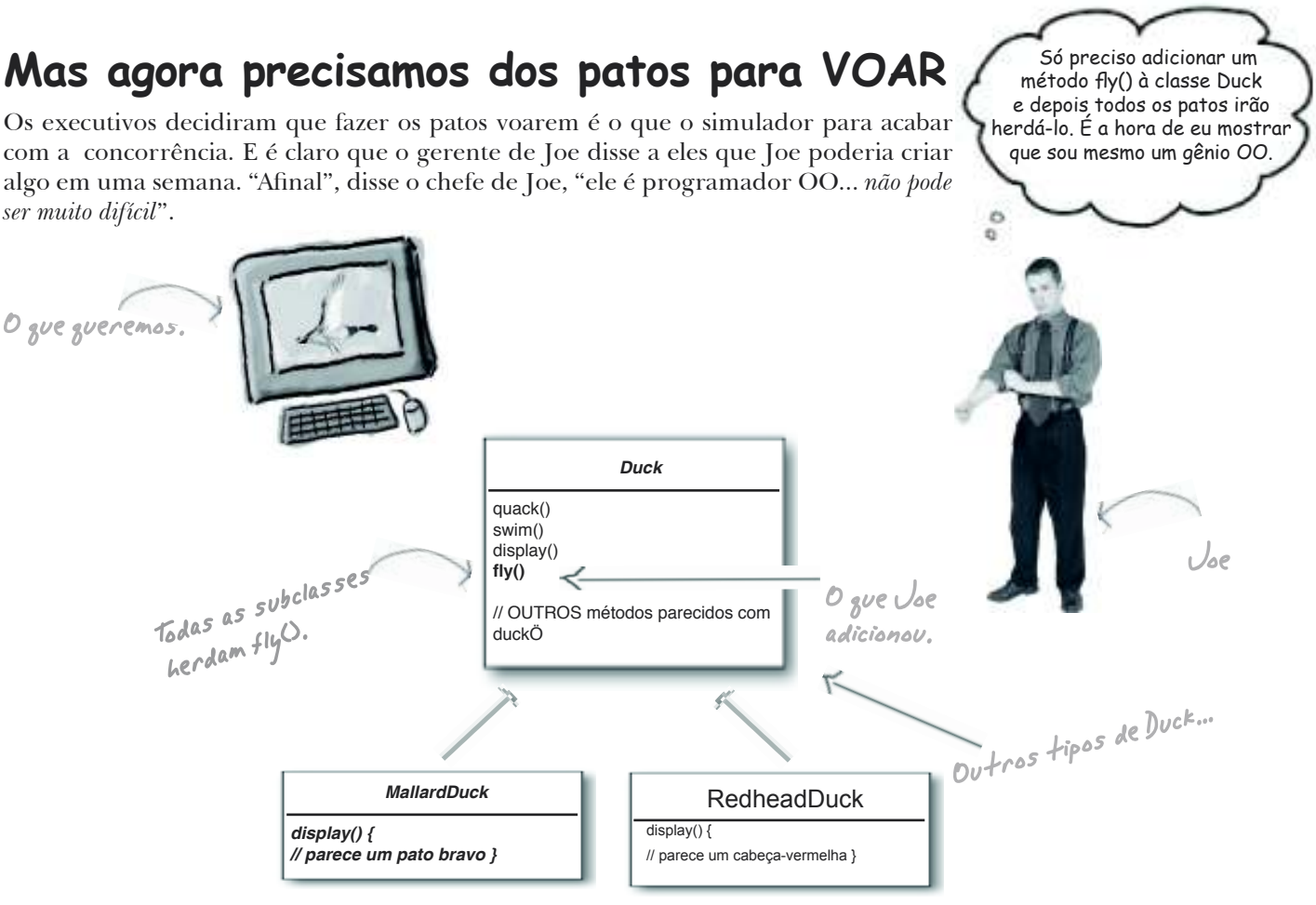
Joe trabalha para uma empresa que cria um jogo de simulação de lago com patos de grande sucesso, o *SimUDuck*. O jogo pode mostrar uma grande variedade de espécies de pato nadando e produzindo sons. Os designers iniciais do sistema usaram técnicas OO padrão e criaram uma superclasse Duck (Pato) herdada por todos os outros tipos de pato.



Ano passado, a empresa esteve sob pressão crescente dos concorrentes. Após uma sessão de brainstorming de uma semana jogando golfe, os executivos da empresa acham que está na hora de fazer uma grande inovação. Eles precisam de algo *realmente* impressionante para mostrar na reunião de acionistas em Maui *na semana seguinte*.

Mas agora precisamos dos patos para VOAR

Os executivos decidiram que fazer os patos voarem é o que o simulador para acabar com a concorrência. E é claro que o gerente de Joe disse a eles que Joe poderia criar algo em uma semana. "Afinal", disse o chefe de Joe, "ele é programador OO... *não pode ser muito difícil*".



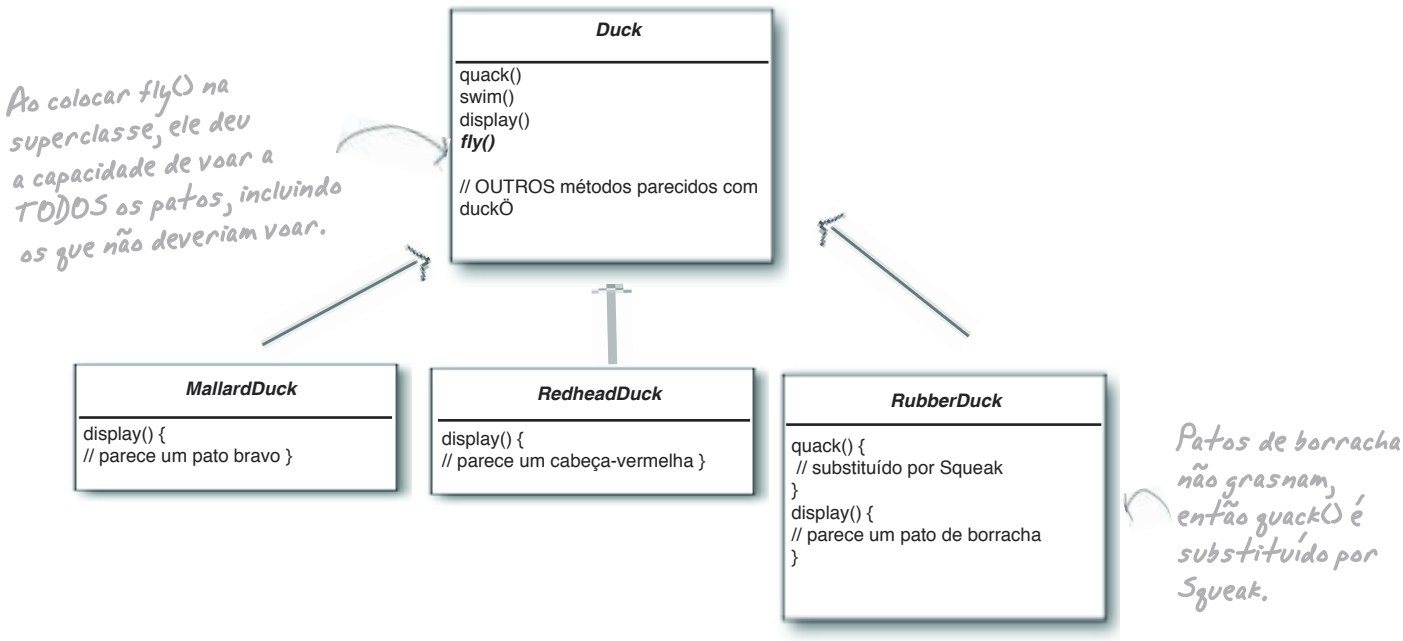
Mas alguma coisa deu muito errado...

O que aconteceu?

Joe não percebeu que nem *todas* as subclasses de Duck deveriam *voar*. Quando Joe adicionou um novo comportamento à superclasse Duck, também estava adicionando um comportamento que *não* era apropriado para algumas subclasses Duck. Agora, objetos inanimados estavam voando no programa SimUPato.

Uma atualização localizada no código causou um efeito colateral não-local (patos de borracha voadores)!

O que ele pensou que fosse um excelente uso da herança para fins de <u>reutilização</u> não dá tão certo quando se trata de <u>manutenção</u>.



Joe pensa sobre herança...

Eu poderia substituir sempre o método fly() em pato de borracha, como faço com o método quack()...

RubberDuck

```
quack() { // squeak}
display() { .// rubber duck }
fly() {
 // substituir para fazer nada}
```

Mas então o que acontece quando adicionamos patos de madeira como isca ao programa? Eles não devem voar nem grasnar...

Esta é outra classe na hierarquia; observe que, assim como RubberDuck, ela não voa, mas também não grasna.

DecoyDuck

```
quack() {
 // substituir para fazer nada
}
display() { // decoy duck}
fly() {
 // substituir para fazer nada
}
```

Aponte o seu lápis

Quais opções a seguir são desvantagens do uso de *herança* para produzir o comportamento Duck? (Selecione todas as opções possíveis.)

A. O código é duplicado entre as subclasses.

B. As alterações no comportamento de tempo de execução são difíceis.

C. Não podemos fazer patos dançar.

D. É difícil conhecer o comportamento de todos os patos.

E. Os patos não conseguem voar e grasnar ao mesmo tempo.

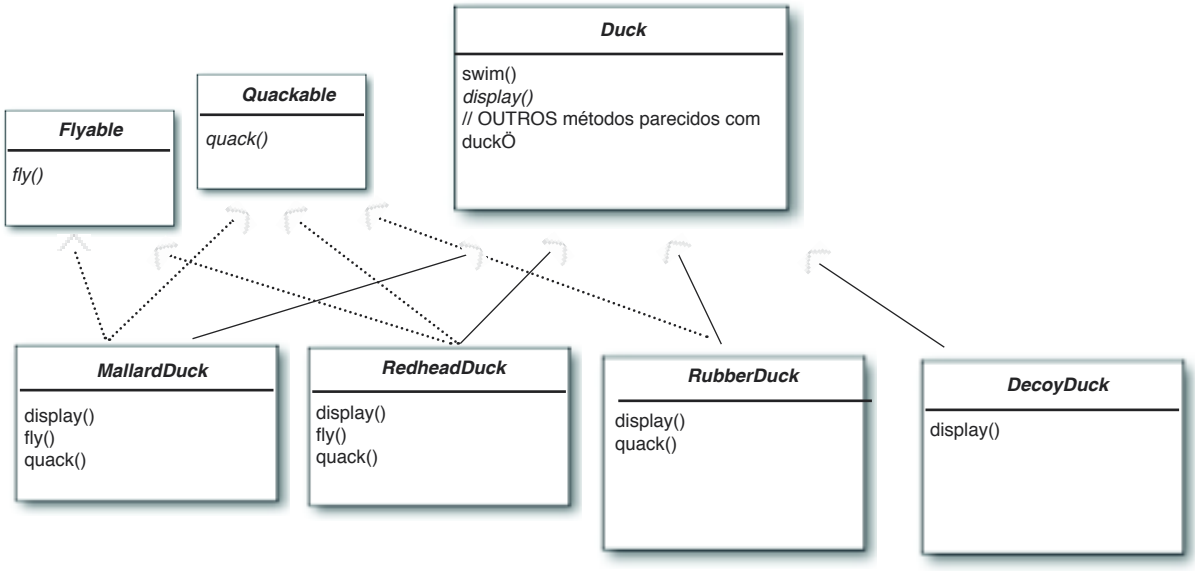
F. As alterações podem afetar sem querer outros patos.

Que tal uma interface?

Joe percebeu que a herança provavelmente não era a resposta por que ele recebeu o aviso de que os executivos agora querem atualizar o produto a cada seis meses (eles ainda não decidiram como). Joe sabe que as especificações vão continuar mudando e que ele será forçado a analisar e possivelmente substituir fly() e quack() para cada nova subclasse Duck adicionada ao programa...*sempre*.

Por isso, ele precisa de uma maneira mais simples para que apenas *alguns* (mas não *todos*) tipos de pato voem ou grasnem.

Eu poderia tirar fly() da superclasse Duck e criar uma **interface Flyable()** com um método fly(). Assim, somente os patos que devem voar irão implementar essa interface e ter um método fly()... e também posso criar uma Quackable, já que nem todos os patos podem grasnar.



O que VOCÊ acha deste projeto?

O que você faria se fosse Joe?

Sabemos que nem *todas* as subclasses devem ter o comportamento de voar ou grasnar, então a herança não é a resposta certa. Mas, embora fazer as subclasses implementar Flyable e/ou Quackable resolva *parte* do problema (sem fazer patos de borracha voar inadequadamente), destrói completamente a reutilização de código para esses comportamentos, criando apenas um pesadelo de manutenção *diferente*. E é claro que pode haver mais de um tipo de comportamento de vôo até entre os patos que *voam*...

Neste ponto, você pode estar esperando que um Padrão de Projetos apareça em um cavalo branco e salve o dia. Mas qual seria a graça? Não, vamos encontrar uma solução da maneira antiga – *aplicando princípios de design de software OO adequados*.