O que estão dizendo sobre Use a Cabeça!

A Amazon escolheu Use a Cabeça! Java como Top Ten Editor's Choice for Computer Books of 2003 (primeira edição)

A Software Development Magazine indicou Use a Cabeça! Java para finalista do 14th Annual Jolt Cola/ Product Excellence Awards

"O livro 'Use a Cabeça! Java', de Kathy e Bert, transformará a página impressa na coisa mais próxima de uma GUI que você jamais viu. De uma maneira divertida e moderna, os autores tornam o aprendizado de Java uma experiência envolvente do tipo 'o que eles vão inventar agora?'."

– Warren Keuffel, Software Development Magazine

"... a única maneira de saber o valor de um tutorial é comprovar se ele é eficiente em ensinar. Use a Cabeça! Java sobressai-se ao ensinar. Certo, achei infantil, porém percebi que estava entendendo completamente os tópicos enquanto percorria o livro."

"O estilo de Use a Cabeça! Java tornou o aprendizado, digamos, mais fácil."

– slashdot (resenha de um alternativo sério)

"Além do estilo atraente que o conduzirá de leigo ao status de defensor exaltado da Java, Use a Cabeça! Java aborda várias questões práticas que outros livros deixam de lado, como o temível 'exercício para o leitor...'. É inteligente, ousado, moderno e prático - não existem muitos livros que conseguem alegar isso e sustentar a alegação enquanto ensinam a serialização de objetos e protocolos de inicialização de rede."

– Dr. Dan Russell, Diretor do User Sciences and Experience Research IBM Almaden Research Center (e que ensina Inteligência Artificial na Universidade de Stanford)

"É rápido, irreverente, divertido e interessante. Tome cuidado - você pode realmente aprender algo!"

– Ken Arnold, ex-engenheiro sênior da Sun Microsystems
Co-autor de "A Linguagem de Programação Java" (com James Gosling, criador do Java)

"A tecnologia Java está em todos os lugares - se você for desenvolvedor de softwares e não tiver aprendido Java, definitivamente chegou a hora de mergulhar - de cabeça."

– Scott McNealy, Presidente, conselheiro e CEO da Sun Microsystems

"Use a Cabeça! Java é como o Monty Python encontrando a gangue dos quatro… O texto é tão bem dividido por quebra-cabeças e histórias, testes e exemplos, que você abordará terreno como em nenhum outro livro de computação."

– Douglas Rowe, Grupo de Usuários Java de Columbia

Elogios a Use a Cabeça! Java

"Leia Use a Cabeça! Java e você passará a experimentar novamente a diversão ao aprender... Para pessoas que gostam de aprender novas linguagens, e não têm experiência em ciência da computação e programação, este livro é uma jóia... É um livro que torna divertido o aprendizado de uma linguagem de computador complexa. Espero que haja mais autores querendo deixar o velho molde dos estilos de escrita 'tradicionais'. Aprender linguagens de computação deve ser divertido e não difícil."

– Judith Taylor, Southeast Ohio Macromedia User Group

"Se você quer aprender Java, não procure mais: bem-vindo ao primeiro livro técnico baseado em GUIs! Este formato inovador e bem-elaborado fornece benefícios que outros textos sobre Java simplesmente não conseguem... Prepare-se para uma jornada realmente notável pelo universo do Java."

– Neil R. Bauman, Capitão & CEO, Geek Cruises (www.GeekCruises.com)

"Se você for relativamente iniciante em programação e estiver interessado em Java, aqui está seu livro... Abordando tudo, dos

objetos à criação de interfaces gráficas de usuário (GUI, graphical user interface), da manipulação de exceções (erros) às redes (soquetes) e segmentação múltipla, e até mesmo o empacotamento de sua pilha de classes em um arquivo de instalação, este livro é bem completo... Se você aprecia esse estilo, estou certo de que amará o livro e, como eu, desejará que a série Use a Cabeça! se estenda a muitos outros assuntos!"

– LinuxQuestions.org

"Fiquei viciado nos contos, códigos comentados, entrevistas engraçadas e exercícios mentais."

– Michael Yuan, autor, Enterprise J2ME

" 'Use a Cabeça! Java'...dá um novo sentido à frase de marketing 'Há sempre um O'Reilly para isso'. Adquiri este livro porque várias pessoas que respeito o descreveram com termos como 'revolucionário', dizendo que era uma abordagem totalmente diferente para um livro. O resultado é engraçado, irreverente, atual, interativo e brilhante... Ler este livro é como sentar na sala de espera de uma conferência, aprendendo – e rindo – com colegas... Se você quiser ENTENDER Java, compre-o."

– Andrew Pollack, www.thenorth.com

"Se há alguém no mundo familiarizado com o conceito de 'Use a Cabeça!', provavelmente sou eu. Este livro é tão bom, que me casaria com ele na TV!"

– Rick Rockwell, comediante
O noivo original do programa de televisão da Fox "Who wants to marry a millionaire"

"Esse negócio é tão estranhamente bom que me faz querer CHORAR! Estou perplexo."

– Floyd Jones, autor sênior de textos técnicos/Poolboy, BEA

"Alguns dias atrás recebi minha cópia de Use a Cabeça! Java de Kathy Sierra e Bert Bates. Li apenas parte do livro, mas o que me surpreendeu é que, mesmo não tendo conseguido dormir naquela primeira noite, me vi pensando: 'Certo, só mais uma página, então irei para a cama.'"

– Joe Litton

Elogios a outros livros da série Use a Cabeça! de co-autoria de Kathy e Bert

A Amazon escolheu Use a Cabeça! Servlets como Top Ten Editor's Choice for Computer Books of 2004 (primeira edição)

A Software Development Magazine indicou Use a Cabeça! Servlets e Use a Cabeça! Design Patterns como finalistas do 15th Annual Product Excellence Awards

"Sinto-me como se milhares de livros tivessem sido tirados de cima de minha cabeça."

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

"Ri, chorei, fiquei comovido."

– Dan Steinberg, editor-chefe, java.net

"Minha primeira reação foi rolar no chão de tanto rir. Depois de me refazer, percebi que este livro não é apenas altamente preciso, e sim que se trata da melhor obra de introdução já publicada sobre padrões de projeto."

– Dr. Timothy A. Budd, professor associado de ciência da computação na Universidade do Estado do Oregon e autor de vários livros, inclusive C++ for Java programmers

"O tom preciso para o codificador genial e casual guru que existe em todos nós. A obra de referência certa para estratégias práticas de desenvolvimento – este livro me fez acompanhar o assunto sem a necessidade de agüentar a ultrapassada e cansativa ladainha acadêmica."

– Travis Kalanick, fundador do Scour and Red Swoosh e membro do MIT TR100

"FINALMENTE – um livro sobre Java escrito da maneira que eu escolheria se eu fosse eu mesmo. Falando sério – este livro definitivamente deixa para trás qualquer outro livro sobre software que já li... Um bom livro é muito difícil de escrever; é preciso muito tempo para deixar as coisas se desdobrarem em uma seqüência natural, "orientada ao leitor". É muito trabalhoso. A maioria dos autores claramente não está à altura do desafio. Parabéns à equipe do Use a Cabeça! EJB por um trabalho de primeira classe!

– Wally Flint

"Não poderia imaginar uma pessoa sorrindo ao estudar um livro de TI! Usando os materiais do Use a Cabeça! EJB, acertei bastante (91%) e consegui um recorde mundial como o mais jovem SCBSD, 14 anos."

– Afsah Shafquat (SCBCD mais jovem do mundo)

"O livro Use a Cabeça! Servlets é tão bom quanto o Use a Cabeça! EJB, que me fez rir E acertar 97% do exame!"

– Jef Cumps, consultor de J2EE, Cronos

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 Padrões de Projeto
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 (2009)
Use a Cabeça Ruby on Rails (2009)

Use a Cabeça! Java™

Tradução da segunda edição

Kathy Sierra
Bert Bates

ALTA BOOKS
E D I T O R A
Rio de Janeiro • 2010

À nossa mente, por estar sempre presente
(apesar de qualquer prova em contrário)

Criadores da série Use a Cabeça!

Kathy tem interesse no ensino de teoria desde quando era projetista de jogos (criou jogos para a Virgin, MGM e Amblin'). Ela desenvolveu grande parte do formato Use a Cabeça! enquanto ensinava Criação em Nova Mídia no programa de extensão em Estudos de Entretenimento da UCLA. Recentemente foi instrutora mestre na Sun Microsystems, preparando os professores da Sun para ensinar as tecnologias Java mais novas, e foi a principal criadora de vários exames de certificação da Sun para programadores e desenvolvedores Java. Junto com Bert Bates, tem usado ativamente os conceitos do *Use a Cabeça! Java* para instruir centenas de professores, desenvolvedores e até não-programadores. Também foi a fundadora de um dos maiores sites Web de comunidade Java do mundo, o javaranch.com, e do blog Creating Passionate Users.

Além deste livro, Kathy foi co-autora de *Use a Cabeça! Servlets*, *Use a Cabeça! EJB* e *Use a Cabeça! Design Patterns*.

Em seu tempo livre ela aprecia seu novo cavalo islandês, gosta de esquiar, correr e da velocidade da luz.

kathy@wickedlysmart.com

Bert é desenvolvedor e projetista de softwares, mas a experiência de uma década em inteligência artificial direcionou seu interesse para o ensino de teoria e para treinamentos baseados em tecnologia. Desde então tem ensinado programação para clientes. Recentemente, foi membro da equipe de desenvolvimento de vários exames de certificação em Java da Sun.

Ele passou a primeira década de sua carreira em softwares viajando pelo mundo para ajudar clientes de radiodifusão como a Radio New Zealand, o Weather Channel e a Arts & Entertainment Network (A & E). Um de seus projetos favoritos foi construir a simulação completa de um sistema de ferrovias para a Union Pacific Railroad.

Bert é um adepto inveterado do player GO e há muito tempo trabalha em um programa Go. Ele é um guitarrista razoável que agora passou para o banjo e gosta de se divertir esquiando, correndo e tentando adestrar (ou ser adestrado por) seu cavalo islandês Andi.

Bert foi co-autor dos mesmos livros que Kathy e está trabalhando muito na próxima remessa (consulte o blog para ver as atualizações).

Você pode encontrá-lo no servidor Go IGS (sob o login *jackStraw*).

terrapin@wickedlysmart.com

Embora Kathy e Bert tentem responder o máximo possível de mensagens de correio eletrônico, o volume de correspondência e sua agenda de viagens torna isso difícil. A melhor (mais rápida) maneira de obter ajuda técnica com relação ao livro é no *bastante* ativo fórum de iniciantes Java em javaranch.com.

Conteúdo (Sumário)

	Introdução	xx
1	Aprofundando-se	1
2	Uma Viagem até Objetópolis	21
3	Conheça suas variáveis	37
4	Como os objetos se comportam	53
5	Métodos extra fortes	71
6	Usando a Biblioteca Java	95
7	Melhor viver em Objetópolis	125
8	Polimorfismo Real	147
9	Vida e morte de um objeto	173
10	Os números são importantes	199
11	Comportamento arriscado	227
12	Uma história muito gráfica	253
13	Trabalhe em seu swing	283
14	Salvando objetos	303
15	Crie uma conexão	333
16	Estrutura da dados	373
17	Lance seu código	407
18	Computação distribuída	423
A	Apêndice A: Receita de código final	455
B	Apêndice B: Os dez principais tópicos que quase entraram no livro	463
	Índice remissivo	475

Sumário

Introdução

i

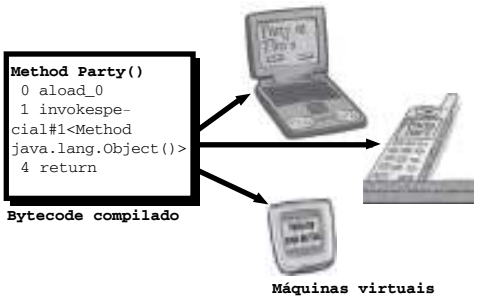
Seu cérebro e o Java. Aqui está você tentando aprender algo, enquanto o seu cérebro está lhe fazendo o favor de garantir que o aprendizado não vingue. Seu cérebro está pensando "É melhor deixar espaço para coisas mais importantes, como que animais selvagens evitar e se praticar snowboard pelado é uma má idéia". Portanto, como você fará o seu cérebro pensar que sua vida depende do que você conhecer a respeito do Java?

Para quem é este livro?	xx
Sabemos o que o seu cérebro está pensando.	xx
Metacognição	xxii
Veja o que fazer para que o seu cérebro se curve em sinal de submissão	xxiii
Requisitos deste livro	xxiv
Editores técnicos	xxvi
Agradecimentos	xxviii

Aprofundando-se

1

O Java o levará a novas fronteiras. No humilde lançamento para o público como a (suposta) versão 1.02, o Java seduziu os programadores com sua sintaxe amigável, recursos orientados a objetos, gerenciamento de memória e, o melhor de tudo – a promessa de portabilidade. Examinaremos isso rapidamente e escreveremos, compilaremos e executaremos alguns códigos. Falaremos sobre a sintaxe, loops, ramificações e o que torna o Java tão interessante. Mergulhe.

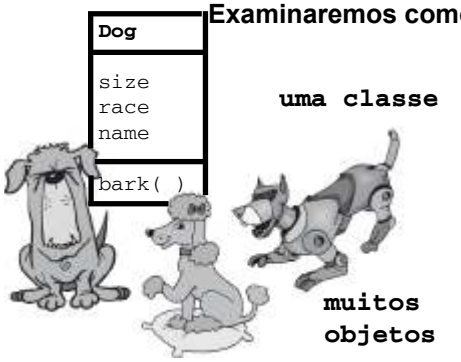


Como a Java funciona	2
Estrutura do código em Java	5
Anatomia de uma classe	7
O método main( )	8
Loops	10
Ramificação condicional (testes if)	10
Codificando o aplicativo "99 garrafas de cerveja"	11
Parafraseando	14
Conversa Informal: compilador vs. JVM	14
Exercícios e quebra-cabeças	16

Uma viagem até Objetópolis

2

Ouvi dizer que haveria objetos. No Capítulo 1, colocamos todo o código no método main( ). Essa não é exatamente uma abordagem orientada a objetos. Portanto, agora temos que deixar esse universo procedimental para trás e começar a criar alguns objetos por nossa própria conta. Examinaremos o que torna o desenvolvimento orientado a objetos (OO, object-oriented) em Java tão divertido. Discutiremos a diferença entre uma classe e um objeto. Examinaremos como os objetos podem melhorar sua vida.



Guerra nas Cadeiras (Brad O Adepto de OO vs. Larry O Usuário de Procedimentos)	22
Herança (uma introdução)	24
Sobrepondo métodos (uma introdução)	25
O que existe em uma classe (métodos, variáveis de instância)?	27
Criando seu primeiro objeto	28
Usando main( )	29
Código do Jogo de Adivinhação	30
Exercícios e quebra-cabeças	33

Conheça suas variáveis

3

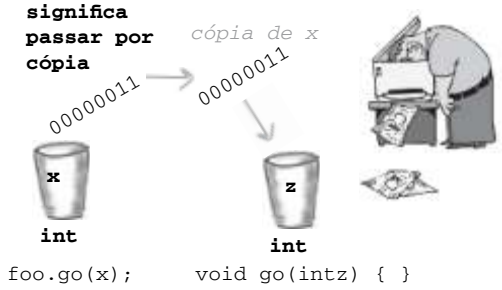
Existem duas versões de variáveis: primitivas e de referência. Deve haver mais coisas na vida além de inteiros, strings e matrizes. E se você tiver um objeto DonodeAnimal com uma variável de instância Cão? Ou um Carro com um Motor? Neste capítulo desvelaremos os mistérios dos tipos usados no Java e examinaremos o que você pode declarar como uma variável, o que pode inserir em uma variável e o que pode fazer com ela. E para concluir discutiremos o que acontece realmente na pilha de lixo coletável.

Declarando uma variável (no Java há a preocupação com o tipo) 38
Tipos primitivos ("Quero um duplo com espuma, por favor") 38
Palavras-chave no Java 40
Variáveis de referência (controle remoto de um objeto) 41
Declaração atribuição de objeto 43
Objetos na pilha de lixo coletável 44
Matrizes (uma introdução) 45
Exercícios e quebra-cabeças 49

Como os objetos se comportam

4

O estado afeta o comportamento, o comportamento afeta o estado. Sabemos que os objetos têm estado e comportamento, representados pelas variáveis de instância e métodos. Agora examinaremos como o estado e o comportamento estão relacionados. O comportamento de um objeto usa um estado exclusivo dele. Em outras palavras, os métodos usam os valores das variáveis de instância. Por exemplo "Se o cão pesar menos de 27 quilos, grite de alegria, caso contrário...". Alteremos alguns estados!

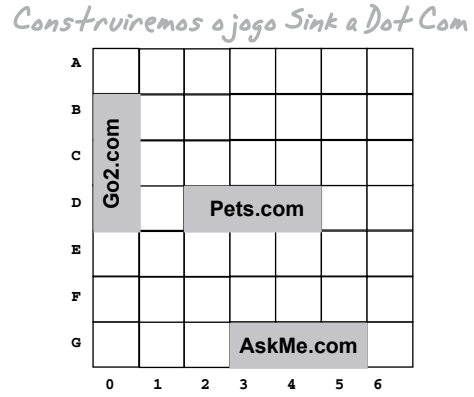


Os métodos usam o estado do objeto (latir diferente) 54
Os argumentos e tipos de retorno do método 55
Passar por valor (a variável é sempre copiada) 57
Métodos de captura e configuração 58
Encapsulamento (use-o ou arrisque-se a ser humilhado) 59
Usando referências em uma matriz 62
Exercícios e quebra-cabeças 65

Métodos extra fortes

5

Aumentemos a força de nossos métodos. Você apredeu sobre as variáveis, brincou com alguns objetos e escreveu um pequeno código. Mas precisa de mais ferramentas. Como os operadores. E os loops. Pode ser útil gerar números aleatórios. E converter uma string em um inteiro, sim, isso seria avançado. E por que não aprender tudo através da criação de algo real, para vermos como é escrever (e testar) um programa a partir do zero. Talvez um jogo, como o Sink a Dot Com (semelhante à Batalha Naval).



Construindo o jogo Sink a Dot Com 72
Começando com o jogo Sink a Dot Com simples (uma versão mais simples) 73
Escrevendo o código preparatório (pseudocódigo do jogo) 76
Código de teste do Dot Com simples 78
Codificando o jogo Dot Com simples 79
Código final do Dot Com simples 81
Gerando números aleatórios com Math.random( ) 85
Código predefinido para obtenção de entradas do usuário a partir da linha... 86
Iterando com loops for 87
Convertendo tipos primitivos extensos para um tamanho menor 90
Exercícios e quebra-cabeças 90

Usando a biblioteca Java

6

O Java vem com centenas de classes predefinidas. **Você não terá que reinventar a roda se souber como encontrar o que precisa na biblioteca Java, normalmente conhecida como API Java. Há coisas melhores a fazer. Se você pretende escrever códigos, pode escrever somente as partes que forem exclusivas de seu aplicativo. A principal biblioteca Java consiste em uma pilha gigante de classes apenas esperando para serem usadas como blocos de construção.**

"Bom saber que há uma ArrayList no pacote java. util. Mas como poderia descobrir isso sozinha?"

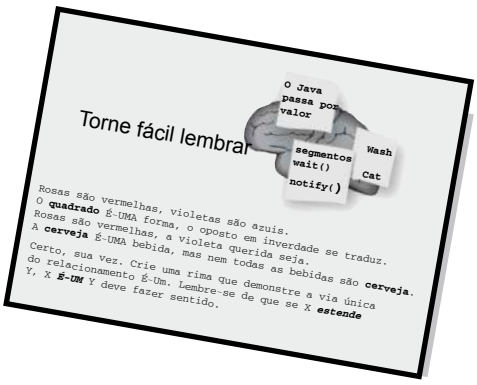
-Julia, 31, modelo de trabalho manual

Analisando o erro do jogo Dot Com simples 96
ArrayList (beneficiando-se da API Java) 99
Corrigindo o código da classe DotCom 104
Construindo o jogo real (Sink a Dot Com) 105
Código preparatório do jogo real 109
Código do jogo real 110
Expressões booleanas 114
Usando a biblioteca (API Java) 116
Usando pacotes (instruções importantes, nomes totalmente qualificados) 116
Usando os documentos e livros de referência do API HTML 119
Exercícios e quebra-cabeças 122

Melhor viver em Objetópolis

7

Planeje seus programas com o futuro em mente. **E se você pudesse escrever códigos que outra pessoa conseguisse estender, facilmente? E se pudesse escrever códigos que fossem flexíveis, para aquelas irritantes alterações de último minuto nas especificações? Quando chegar ao Plano de Polimorfismo, você aprenderá as 5 etapas para a obtenção de um projeto de classes mais adequado, os 3 truques do polimorfismo, as 8 maneiras de criar um código flexível e, se agir agora – uma lição bônus sobre as 4 dicas para a exploração da herança.**

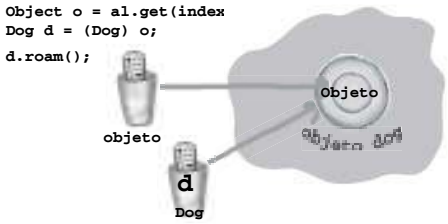


Entendendo a herança (relacionamentos da superclasse e subclasse) 127
Projetando uma árvore de herança (a simulação da classe Animal) 129
Evitando a duplicação de código (usando a herança) 129
Sobrepondo métodos 130
É-UM e TEM-UM (garota na banheira) 132
O que você herdará de sua superclasse? 135
O que a herança lhe fornecerá realmente? 136
Polimorfismo (usando a referência de um supertipo a um objeto da subclasse) 137
Regras da sobreposição (não mexa nesses argumentos e tipos de retorno!) 140
Sobrecarga do método (nada mais do que a reutilização do nome do método) 141
Exercícios e quebra-cabeças 142

Polimorfismo real

8

A herança é apenas o começo. **Para explorar o polimorfismo, precisamos de interfaces. Temos que ir além da simples herança e alcançar a flexibilidade que você só conseguirá projetando e codificando em interfaces. O que é uma interface? Uma classe 100% abstrata. O que é uma classe abstrata? Uma classe que não pode ser instanciada, Em que isso é útil? Leia o capítulo...**



Converta a classe Object em uma classe Cão que você sabe que existe.

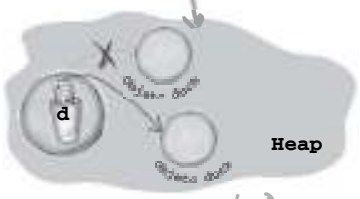
Algumas classes simplesmente não devem ser instanciadas 149
Classes abstratas (não podem ser instanciadas) 150
Métodos abstratos (devem ser implementados) 151
O polimorfismo em ação 153
Classe Object (a superclasse final de tudo) 154
Extraindo objetos de uma ArrayList (eles são capturados com o tipo Object) 155
O compilador verifica o tipo de referência (antes de permitir que você chame um método) 157
Entrando em contato com seu objeto interno 158
Referências polimórficas 159

Convertendo uma referência de objeto (passando mais para baixo na árvore de herança) 160
Losango Fatal (problema de herança múltipla) 164
Usando interfaces (a melhor solução!) 164
Exercícios e quebra-cabeças 169

9 Vida e morte de um objeto

Objetos nascem e objetos morrem. Você é quem manda. Você decide quando e como construí-los. Decide quando abandoná-los. O Coletor de Lixo (gc, garbage collector) solicita memória. Examinaremos como os objetos são criados, onde residem e como manter ou abandoná-los eficientemente. Isso significa que falaremos sobre o heap, a pilha, o escopo, construtores, superconstrutores, referências nulas e qualificação para o gc.

Quando alguém chamar o método go(), esse objeto Duck será abandonado. Sua única referência foi reprogramada para um objeto Duck diferente.



d recebeu um novo objeto Duck, deixando o objeto Duck original (o primeiro) abandonado. Agora esse primeiro objeto pode ser considerado eliminado.

A pilha e o heap onde os objetos e as variáveis residem 174
Métodos da pilha 174
Onde as variáveis locais residem 175
Onde as variáveis de instância residem 176
O milagre da criação de objetos 177
Construtores (o código que será executado quando você usar new) 177
Inicializando o estado de um novo objeto Duck (Pato) 179
Construtores sobrecarregados 181
Construtores de superclasse (cadeia de construtores) 184
Chamando construtores sobrecarregados usando this( ) 186
A vida de um objeto 189
Coleta de Lixo (e como tornar os objetos qualificados) 191
Exercícios e quebra-cabeças 196

10 Os números são importantes

Faça o cálculo. O API Java tem métodos para valor absoluto, arredondamento, min/max, etc. Mas e quanto à formatação? Você pode querer que os números sejam exibidos apenas com duas casas decimais ou com pontos em todos os locais corretos. E pode querer exibir e manipular datas, também. E quanto à conversão de uma string em um número? Ou a conversão de um número em uma string? Começaremos aprendendo o que significa para uma variável ou método ser estático.

Variáveis estáticas são compartilhadas por todas as instâncias de uma classe.

primeira instância de kid (criança)

segunda instância de kid

variável estática: iceCream (sorvete)

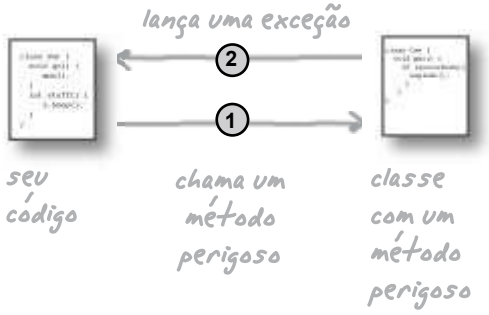
Classe Math (você precisa realmente de uma instância dela?) 200
Métodos estáticos 201
Variáveis estáticas 204
Constantes (variáveis estáticas finais) 206
Métodos de Math (random( ), round( ), abs( ), etc.) 209
Classes encapsuladoras (Integer, Boolean, Character, etc.) 209
Auto-inserção 209
Formatação de números 214
Formatação e manipulação de datas 219
Importações estáticas 222
Exercícios e quebra-cabeças 224

Variáveis de instância: uma por instância

Variáveis estáticas: uma por classe

11 Comportamento arriscado

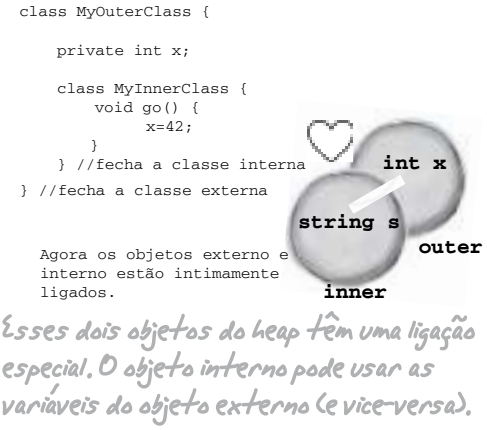
Problemas acontecem. O arquivo não está no local. O servidor está travado. Independentemente de quanto você é bom em programação, não é possível controlar tudo. Quando criar um método perigoso, precisará de um código para manipular o que acontecer de errado. Mas como saber quando um método é perigoso? Onde inserir o código que manipulará a situação excepcional? Neste capítulo, construiremos um MIDI Music Player, que usará o perigoso API JavaSound, portanto, é melhor descobrirmos.



Criando um aparelho de som (como o BeatBox) 228
E se você tiver que chamar o código perigoso? 230
Exceções dizem "algo inadequado pode ter ocorrido..." 230
O compilador garantirá (ele verificará) que você fique ciente dos riscos 231
Capturando exceções usando uma instrução try/catch (skatista) 232
Controle do fluxo em blocos try/catch 235
O bloco finally (não importa o que aconteça, desligue o forno!) 235
Capturando várias exceções (a ordem é importante) 237
Declarando uma exceção (apenas desvie) 241
Manipule ou declare como lei 242
Receita de código (emitindo sons) 243
Exercícios e quebra-cabeças 249

12 Uma história muito gráfica

Encare a realidade, você precisa criar GUIs. Mesmo se acredita que durante o resto de sua vida escreverá somente código no lado do servidor, cedo ou tarde terá que criar ferramentas e vai querer uma interface gráfica. Dedicaremos dois capítulos às GUIs e aprenderemos mais sobre determinados recursos de linguagem, inclusive a Manipulação de Eventos e as Classes Internas. Inseriremos um botão na tela, pintaremos a tela, exibiremos uma figura jpeg e trabalharemos ate mesmo com um pouco de animação.



Sua primeira GUI 254
Capturando um evento de usuário 256
Implemente uma interface ouvinte 256
Capturando o evento ActionEvent de um botão 258
Inserindo figuras em uma GUI 260
Diversão com paintComponent( ) 261
O objeto Graphics2D 262
Inserindo mais de um botão em uma tela 265
Classes internas ao resgate (torne seu ouvinte uma classe interna) 268
Animação (mova-a, pinte-a, mova-a, pinte-a, mova-a, pinte-a...) 272
Receita de código (pintando figuras ao ritmo de música) 275
Exercícios e quebra-cabeças 280

13 Trabalhe em seu Swing

O Swing é fácil. A menos que você se importe realmente com o local de cada elemento. O código Swing parece fácil, mas depois de compilar, executar e examiná-lo nos damos conta disto: "ei, isso não deveria estar aí." O que torna fácil a codificação é o que torna difícil o controle – o Gerenciador de Layout. Mas com um pouco de esforço, você pode fazer os gerenciadores de layout se curvarem a sua vontade. Neste capítulo, trabalharemos em nosso Swing e aprenderemos mais sobre os elementos gráficos.

Componentes do Swing 284
Gerenciadores de Layout (eles controlam o tamanho e o local) 284
Três Gerenciadores de Layout (borda, fluxo, caixa) 286

Os elementos das partes superior e inferior ficaram com a altura selecionada. Os componentes da esquerda e da direita ficaram com a largura selecionada.

BorderLayout (controla cinco regiões) 286
FlowLayout (controla a ordem e o tamanho preferido) 288
BoxLayout (como no fluxo, mas pode empilhar componentes verticalmente) 290
JTextField (para entrada de usuário de uma linha) 292
JTextArea (para texto de rolagem de várias linhas) 292
JCheckBox (foi feita a seleção?) 293
JList (uma lista de seleção rolável) 294
Receita de código (O código completo – construindo o cliente de bate-papo BeatBox) 295
Exercícios e quebra-cabeças 299

14 Salvando objetos

Os objetos podem ser achatados e reconstituídos. Os objetos possuem estado e comportamento. O comportamento reside na classe, mas o estado reside dentro de cada objeto. Se o seu programa tiver que salvar o estado, você poderá fazê-lo da maneira mais difícil, examinando cada objeto e gravando meticulosamente o valor de cada variável de instância. Ou, da maneira mais fácil, orientada a objetos – simplesmente congele o objeto (serialize-o) e reconstitua-a (desserialize), para que volte ao que era.

Alguma pergunta?

Salvando o estado do objeto 304
Gravando um objeto serializado em um arquivo 305
Fluxos Java de entrada e saída (conexões e encadeamentos) 306
Serialização de objeto 307
Implementando a interface Serializable 308
Usando variáveis transientes 310
Desserializando um objeto 311
Gravando em um arquivo de texto 314
java.io.File 318
Lendo em um arquivo de texto 319
Dividindo uma string em fichas com split( ) 322
Receita de código 326
Exercícios e quebra-cabeças 328

15 Crie uma conexão

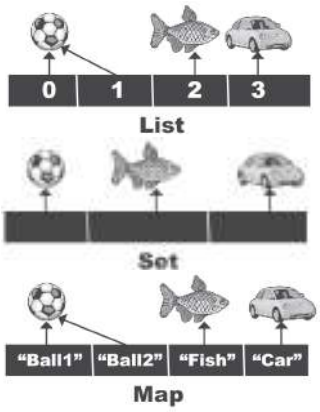
Conecte-se com o ambiente externo. É fácil. Todos os detalhes de nível inferior de rede são definidos pelas classes na biblioteca java.net. Um dos melhores recursos do Java é que enviar e receber dados através de uma rede é realmente apenas uma atividade de E/S com um fluxo de conexão um pouco diferente na extremidade da cadeia. Neste capítulo criaremos soquetes de cliente. Criaremos soquetes de servidor. Criaremos clientes e servidores. Antes do fim do capítulo, você terá um cliente de bate-papo totalmente funcional com vários segmentos. Dissemos com vários segmentos?

Visão geral do programa de bate-papo 334
Conectando, enviando e recebendo 335
Soquetes de rede 336
Portas TCP 336
Lendo dados em um soquete (usando BufferedReader) 338
Gravando dados em um soquete (usando PrintWriter) 339
Escrevendo o programa Daily Advice Client 340
Criando um servidor simples 341
Código do Daily Advice Server 341
Criando um cliente de bate-papo 343
Várias pilhas de chamada 347
Iniciando um novo segmento (crie-o, inicie-o) 347

A interface Runnable (a tarefa do segmento) 348
Três estados de um novo objeto Thread (novo, executável, em execução) 349
O loop executável em execução 350
Agendador de segmentos (é ele quem decide e não você) 350
Colocando um segmento em suspensão 353
Criando e iniciando dois segmentos 354
Problemas de concorrência: esse casal pode ser salvo? 355
O problema de concorrência de Ryan e Mônica, em código 356
Bloqueando para gerar atomicidade 359
Todo objeto tem um bloqueio 360
O temível problema da "Atualização Perdida" 360
Métodos sincronizados (usando um bloqueio) 362
Impasse! 363
Código de ChatClient com vários segmentos 365
Código predefinido para SimpleChatServer 366
Exercícios e quebra-cabeças 369

16 Estruturas de dados

A classificação é instantânea em Java. Você tem todas as ferramentas para coletar e manipular dados sem ter que escrever seus próprios algoritmos de classificação. O Java Collections Framework tem uma estrutura de dados que deve funcionar para praticamente qualquer coisa que você precisar fazer. Quer manter uma lista que você possa aumentar facilmente? Encontrar algo pelo nome? Criar uma lista que exclua automaticamente todos os dados repetidos? Classificar seus colaboradores por quantas vezes lhe traíram?

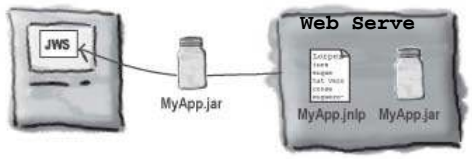


Conjuntos 374
Classificando como ArrayList com Collections.sort( ) 376
Dados genéricos e garantia de tipo 380
Classificando itens que implementam a interface Comparable 385
Classificando itens com um comparador personalizado 388
O API de conjuntos – listas, conjuntos e mapas 392
Evitando dados repetidos com HashSet 392
Sobrepondo hashCode( ) e equals( ) 394
HashMap 398
Usando curingas para gerar polimorfismo 403
Exercícios e quebra-cabeças 404

17 Lance seu código

É hora de pôr em prática. Você escreveu seu código. Testou o código. Aprimorou-o. Você contou para todo mundo que conhece que, se nunca se deparar com uma linha de código novamente, não haverá problema. Mas, no fim das contas, terá criado uma obra de arte. O negócio funciona mesmo! Porém, fazer o que agora? Nesses dois últimos capítulos, estudaremos como organizar, empacotar e implantar seu código Java. Examinaremos opções de implantação local, semilocal e remota, incluindo arquivos jar executáveis, o Java Web Start, RMI e Servlets. Calma. Alguns dos recursos mais interessantes em Java são mais fáceis de usar do que você imagina.

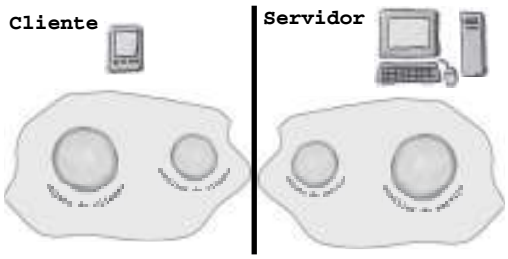
Opções de implantação 408
Mantenha os arquivos de seu código-fonte e de suas classes separados 409
Criando um arquivo JAR (Java ARchives) executável 410
Processando um arquivo JAR executável 411
Insira suas classes em um pacote! 411
Os pacotes devem ter uma estrutura de diretório adequada 412
Compilando e executando com pacotes 413
Compilando com -d



Criando um arquivo JAR executável (com pacotes) 414
O Java Web Start (JWS) para a implantação na Web 415
Como criar e implantar um aplicativo JWS 419
Exercícios e quebra-cabeças 421
421

18 Computação distribuída

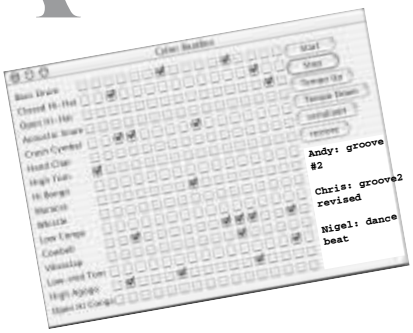
Trabalhar remotamente não precisa ser ruim. Certo, as coisas são mais fáceis quando todas as partes do aplicativo estão em um local, em um heap, com um JVM para regular tudo. Mas nem sempre isso é possível. Ou desejável. E se seu aplicativo manipular cálculos poderosos? E se ele precisar de dados de um banco de dados seguro? Neste capítulo, aprenderemos a usar o surpreendentemente simples Remote Method Invocation (RMI) do Java. Também examinaremos rapidamente os Servlets, os Enterprise Java Beans (EJB) e o Jini.



O Remote Method Invocation (RMI) do Java, na prática, bem detalhado 426
Servlets (uma visão rápida) 437
Enterprise Java Beans (EJB), uma visão muito rápida 442
Jini, o melhor entre todos os truques 443
Construindo o navegador universal de serviços realmente avançado 445
O Fim 454

A Apêndice A

O projeto final da receita de código. O código completo do beat box de bate-papo cliente-servidor. Sua chance de ser uma estrela do rock.



BeatBoxFinal (código cliente) 456
MusicServer (código servidor) 460

B Apêndice B

Os dez principais itens que não apareceram no livro. Ainda não podemos soltá-lo no mundo. Temos mais algumas coisas para você, mas é aqui que acaba o livro. E dessa vez é verdade.

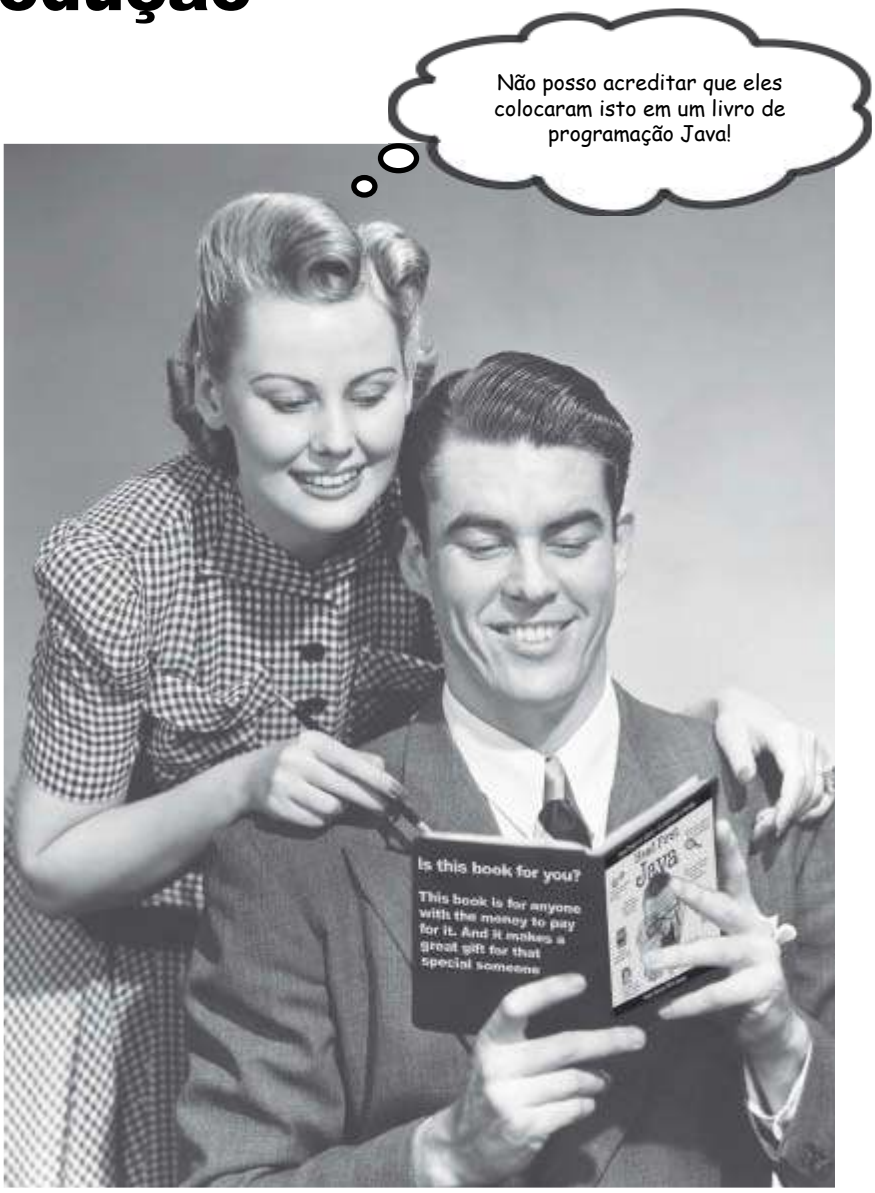
Lista dos dez mais 464

I Índice Remissivo

475

Como usar este livro

Introdução



Nesta seção, respondemos a pergunta intrigante:
Mas por que eles colocaram isto em um livro de programação em Java?

A quem se destina este livro?

Se você puder responder "sim" a ***todas*** estas perguntas:

1. **Você já programou?**
2. **Quer aprender Java?**
3. **Prefere conversas estimulantes na hora do jantar a palestras secas, chatas e técnicas?**

Então este livro é destinado a você.

Este não é uma obra de referência. Use a Cabeça! Java é um livro projetado para aprendizado, não é uma enciclopédia de fatos sobre a Java.

Quem provavelmente deve ficar longe deste livro?

Se você puder responder "sim" a **qualquer** das perguntas a seguir:

1. **Sua experiência em programação se limita somente à HTML, sem nenhum contato com linguagens de script?**
   (Se você já fez algo com loops ou lógica if/then, conseguirá se virar com este livro, mas somente tags HTML podem não ser o suficiente.)
2. **Você é um bom programador de C++ procurando um livro de *referência*?**
3. **Você tem medo de tentar algo diferente? Prefere fazer um tratamento de canal a misturar listras e xadrez? Acredita que um livro técnico não pode ser sério se houver a figura de um pato na seção de gerenciamento da memória?**

Então este livro não é para você.

[Nota do pessoal de marketing: quem retirou a parte sobre como este livro serve para qualquer pessoa com um cartão de crédito válido? E quanto à promoção de férias Dê uma Java de presente que discutimos... – Fred.]

Sabemos em que você está pensando.

"Como *isto* pode ser um livro sério de programação Java?"

"Para que todas as figuras?"

"Conseguirei aprender realmente dessa forma?"

"Estou sentindo cheiro de pizza?"

Seu cérebro acha que ISSO é importante.

Sabemos o que o seu cérebro está pensando.

Seu cérebro procura novidade. Está sempre procurando, pesquisando, *esperando* por algo incomum. Ele foi gerado assim e o ajuda a permanecer vivo.

Atualmente, há menos probabilidades de você se tornar o almoço de um tigre. Mas seu cérebro ainda está procurando. Só que você não sabe.

Mas o que seu cérebro faz com toda as coisas rotineiras, comuns e cotidianas que você encontra? O *possível* para evitar que interfiram em sua *verdadeira* tarefa – gravar as coisas que *interessam*. Não interessa gravar as coisas irrelevantes; elas nunca passam pelo filtro "é claro que isso não é importante".

Como seu cérebro *sabe* o que é importante? Suponhamos que você saia para a sua caminhada diária e um tigre salte em sua frente; o que aconteceria dentro de sua cabeça?

Acionamento dos neurônios. Ativação das emoções. *Explosão química*.

E é assim que seu cérebro fica sabendo...

Isso deve ser importante! Não esqueça!

Mas imagine se você estivesse em casa, ou em uma biblioteca. É um local seguro, aconchegante, sem tigres. Você está estudando. Preparando-se para um exame. Ou tentando aprender algum assunto técnico difícil, algo que o seu chefe acha que vai levar uma semana, dez dias no máximo.

Só há um problema. Seu cérebro está tentando lhe fazer um grande favor. Está tentando se certificar de que esse conteúdo *obviamente* irrelevante não use recursos escassos. Recursos que seriam melhor utilizados no armazenamento

das coisas realmente *importantes*. Como os tigres. Como o perigo de incêndio. Como você nunca tentar praticar novamente snowboard de short.

E não há uma maneira simples de dizer a seu cérebro: "Ei, cérebro, muito obrigado, mas, independentemente de o quanto esse livro seja chato, e de como eu estou registrando esse fato em um nível baixo na escala Richter emocional nesse momento, quero *realmente* que você guarde isso."

Pensamos no leitor de "Use a Cabeça! Java" como um aprendiz.

Mas o que é necessário para *aprender* algo? Primeiro, você tem que *captar*, depois se certificar de que não vai deixar isso *escapar*. Não é apenas empurrar os fatos para dentro de sua cabeça. Com base nas pesquisas mais recentes da ciência cognitiva, da neurobiologia e da psicologia educacional, é preciso muito mais para se *aprender* do que apenas texto em uma página. Sabemos o que interessa ao seu cérebro.

Alguns dos princípios de aprendizagem da série Use a Cabeça!:

Destaque o aspecto visual. As figuras são muito mais fáceis de memorizar do que palavras isoladas e tornam o aprendizado mais eficaz (até 89% de melhoria em estudos de lembrança e transferência de conhecimento). Também tornam as coisas mais inteligíveis. **Coloque as palavras dentro ou perto da figura** às quais estão relacionadas, em vez de embaixo ou em outra página, e os aprendizes ficarão *duas vezes* mais aptos do que o normal a resolver problemas referentes ao conteúdo.

Use um estilo coloquial e personalizado. Em estudos recentes, alunos se saíram até 40% melhor em testes pós-aprendizado quando o conteúdo se comunicava diretamente com o leitor, usando um estilo coloquial em primeira pessoa em vez de adotar um tom formal. Contar histórias em vez de proferir palestras. Usar linguagem casual. Não se leve tão a sério. Em quem *você* prestaria mais atenção: em uma companhia estimulante no jantar ou em uma palestra?

Faça o aprendiz pensar com mais afinco. Em outras palavras, a menos que você flexione seus neurônios ativamente, não acontecerá muita coisa em sua cabeça. Um leitor tem que estar motivado, engajado, curioso e inspirado a resolver problemas, tirar conclusões e gerar novos conhecimentos. E, para que isso ocorra, você precisa de desafios, exercícios e perguntas que instiguem o pensamento, além de atividades que envolvam os dois lados do cérebro e vários sentidos.

Faz sentido dizer que a banheira É-UM banheiro? Que o banheiro É-UMA banheira? Ou trata-se de um relacionamento TEM-UM?

Prenda – e mantenha presa – a atenção do leitor. Todos nós já tivemos uma experiência do tipo "quero realmente aprender isso, mas não consigo ficar acordado após a página um". Seu cérebro presta atenção em coisas que são fora do comum, interessantes, estranhas, atraentes, inesperadas. Aprender sobre um assunto técnico novo e difícil não precisa ser chato. O cérebro aprenderá muito mais rapidamente se não o for.

Toque as emoções deles. Agora sabemos que sua habilidade de se lembrar de algo depende muito do conteúdo emocional. Você se lembrará do que lhe preocupar. Lembrará quando sentir algo. Não estamos falando de histórias dramáticas sobre um menino e seu cachorro. Estamos falando de surpresa, curiosidade, diversão, de pensamentos do tipo "mas o que é isso?" e do sentimento "sou mais eu!" que surge quando você resolve um enigma, aprende algo que as outras pessoas acham difícil ou percebe que sabe algo que Bob "sou mais técnico que você" da engenharia *não sabe.*

Metacognição: entendendo o pensamento.

Se você quiser realmente aprender, e quiser fazê-lo mais rápida e eficientemente, preste atenção em como sua atenção é atraída. Pense em como você pensa. Entenda como aprende.

Quase ninguém fez cursos de metacognição ou teoria do aprendizado quando estava crescendo. *Esperavam* que aprendêssemos, mas raramente nos ensinavam a aprender.

Mas presumimos que, por você estar segurando este livro, deseja aprender Java. E é provável que não queira demorar muito.

Para aproveitar este livro ao máximo, ou *qualquer* livro ou experiência de aprendizagem, tome as rédeas de seu cérebro. Dedique-se a *esse* conteúdo.

O truque é fazer seu cérebro ver o novo material que você está aprendendo como Realmente Importante. Crucial para seu bem-estar. Tão importante quanto um tigre. Caso contrário, você estará em batalha constante, com seu cérebro fazendo o melhor para não deixar o novo conteúdo escapar.

Mas exatamente *como* fazer seu cérebro tratar a Java como se fosse um tigre faminto?

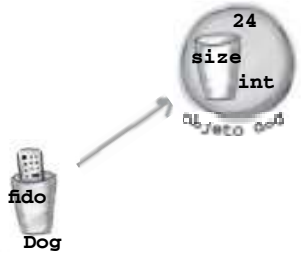
Há a maneira tediosa e lenta ou a mais rápida e eficaz. A maneira lenta é através da repetição contínua. É claro que você sabe que pode aprender e se lembrar até do mais chato dos tópicos, se continuar insistindo nisso. Com um nível suficiente de repetição, seu cérebro pensará: "Isto não *parece* importante, mas, como ele continua se dedicando à mesma coisa *repetidamente*, portanto, suponho que deva ser."

A maneira mais rápida é fazer ***qualquer coisa que aumente a atividade cerebral***, principalmente *tipos* diferentes de atividade cerebral. Os itens da página anterior são grande parte da solução e todos comprovadamente ajudarão seu cérebro a trabalhar a seu favor. Por exemplo, estudos mostram que inserir palavras *dentro* das figuras que elas descrevem (e não em algum outro local da página, como em uma legenda ou no corpo do texto) fará com que seu cérebro tente descobrir como as palavras e a figura estão relacionadas, e isso ocasionará o acionamento de mais neurônios. Maior acionamento de neurônios = mais chances de seu cérebro *perceber* que isso é algo em que vale a pena prestar atenção e possivelmente memorizar.

O estilo coloquial ajuda, porque as pessoas tendem a prestar mais atenção quando percebem que estão em uma conversa, já que se espera que elas acompanhem o assunto e exponham sua opinião. O interessante é que seu cérebro não está necessariamente *preocupado* com o fato de a "conversa" ser entre você e um livro! Por outro lado, se o estilo da redação for formal e seco, ele a perceberá como se você estivesse assistindo a uma palestra enquanto senta em uma sala cheia de espectadores passivos. Não é preciso ficar acordado.

Mas figuras e um estilo coloquial são apenas o começo.

Aqui está o que NÓS fizemos:



Usamos ***figuras***, porque seu cérebro capta estímulos visuais e não texto. No que diz respeito ao cérebro, uma figura realmente *vale* por 1.024 palavras. E quando usamos texto e figuras em conjunto, embutimos o texto *nas* figuras, porque o cérebro funciona mais eficientemente quando o texto está *dentro* daquilo a que ele se refere e não em uma legenda ou oculto em algum local da redação.

Usamos a ***repetição***, dizendo a mesma coisa de diferentes maneiras e por meios distintos, e *vários sentidos*, para aumentar a chance de que o conteúdo seja codificado em mais de uma área de seu cérebro.

Usamos conceitos e figuras de maneiras ***inesperadas***, porque seu cérebro capta novidades, e empregamos figuras e idéias com pelo menos *algum conteúdo* ***emocional***, porque o cérebro foi programado para prestar atenção à bioquímica das emoções. Qualquer coisa que fizer você *sentir* algo terá mais probabilidade de ser lembrada, mesmo se esse sentimento não passar de uma pequena **animação, surpresa** ou **interesse.**

Usamos um **estilo coloquial** personalizado, porque seu cérebro foi programado para prestar mais atenção quando acredita que está ocorrendo uma conversa do que quando acha que você está passivamente assistindo a uma apresentação. Ele fará isso até mesmo quando você estiver *lendo.*

Incluímos mais de 50 ***exercícios***, porque seu cérebro foi programado para aprender e lembrar melhor quando você ***faz*** coisas e não quando *lê*. E criamos exercícios desafiadores porém viáveis, porque é isso que a maioria das *pessoas* prefere.

PONTOS DE BALA

Usamos ***vários estilos de aprendizagem***, porque *você* pode preferir procedimentos passo a passo, enquanto outra pessoa pode querer ter uma visão geral primeiro e outra deseje apenas ver um exemplo de código. Mas independentemente de sua preferência de aprendizado, *todos* se beneficiarão em ver o mesmo conteúdo representado de várias maneiras.

Tudo sobre o Java

Incluímos conteúdo para os ***dois lados de seu cérebro***, porque, quanto mais ele estiver comprometido, maior probabilidade você terá de aprender e lembrar e mais tempo conseguirá se concentrar. Já que trabalhar um lado do cérebro geralmente significa dar ao outro lado a chance de descansar, você pode ser mais produtivo no aprendizado durante um período maior.

E incluímos ***histórias*** e exercícios que apresentam ***mais de um ponto de vista***, porque seu cérebro foi programado para aprender mais intensamente quando é forçado a fazer avaliações e julgamentos.

Exercitando o cérebro

Incluímos ***desafios***, com exercícios, e ***perguntas*** que nem sempre têm uma resposta direta, porque seu cérebro foi programado para aprender e lembrar quando tem que *trabalhar* em algo (da mesma forma que você não consegue colocar seu corpo em forma apenas observando pessoas fazendo ginástica). Mas fizemos o melhor que pudemos para assegurar que, quando você estiver se esforçando muito, isso ocorra envolvendo as coisas *certas*. Que ***você não gaste nem mesmo um dendrite extra*** processando um exemplo difícil de entender ou analisando jargões carregados e complexos ou texto extremamente conciso.

Usamos uma abordagem ***80/20***. Presumimos que, se você estiver tentando obter um PhD em Java, esse não será seu único livro. Portanto, não abordamos *tudo*. Apenas o que você realmente *usará*.

Veja o que fazer para que o seu cérebro se curve em sinal de submissão

Fizemos nossa parte. O resto é com você. Essas dicas são um ponto de partida; escute seu cérebro e descubra o que funciona com você e o que não funciona. Tente coisas novas.

Recorte isso e cole em sua geladeira.

1 Tenha calma. Quando mais você entender, menos terá que memorizar.
Não *leia* apenas. Pare e pense. Quando o livro lhe fizer uma pergunta, não passe apenas para a resposta. Imagine que alguém *está* realmente fazendo a pergunta. Quando você forçar seu cérebro a pensar, mais chances terá de aprender e lembrar.

2 Faça os exercícios. Escreva suas próprias anotações.
Nós os inserimos, mas se os resolvermos, isso seria como ter outra pessoa fazendo uma prova para você. E não *olhe* apenas para os exercícios. **Use um lápis**. Há muitas evidências de que a atividade física *durante* o estudo pode aumentar o aprendizado.

3 Leia a parte "Não existem perguntas idiotas"
Quero dizer todas. Não são apenas notas laterais – elas fazem parte do conteúdo principal! Às vezes as perguntas são mais úteis do que as respostas.

4 Não leia tudo no mesmo local.
Levante-se, estique o corpo, mova-se, mude de cadeira, vá até outra sala. Isso ajudará seu cérebro a *sentir* algo e não deixará que o aprendizado fique muito ligado a um local específico.

5 Deixe essa ser a última coisa a ser feita antes de você ir para a cama. Ou pelo menos a última coisa *desafiadora*.
Parte do aprendizado (principalmente a transferência para a memória de longo prazo) ocorrerá *depois* que você fechar o livro. Seu cérebro precisa de um tempo próprio, para continuar processando. Se você captar algo novo durante esse tempo de processamento, parte do que acabou de aprender será perdida.

6 Beba água. Muita água.
Seu cérebro funcionará melhor com um bom banho. A desidratação (que pode ocorrer antes mesmo de você sentir sede) diminui a função cognitiva.

7 Fale sobre o assunto. Em voz alta.
Falar ativa uma parte diferente do cérebro. Se você estiver tentando entender algo, ou aumentar suas chances de se lembrar de algo posteriormente, fale em voz alta. Melhor ainda, tente explicar em voz alta para outra pessoa. Você aprenderá mais rapidamente e pode descobrir particularidades que não tinha percebido ao ler sobre o assunto.

8 Escute seu cérebro.
Preste atenção se seu cérebro está ficando sobrecarregado. Se perceber que começou a ler superficialmente ou a esquecer o que acabou de ler, é hora de fazer um intervalo. Uma vez que tiver passado de um certo ponto, você não aprenderá com maior rapidez tentando assimilar mais e pode até prejudicar o processo.

9 Sinta algo!
Seu cérebro precisa saber que isso é *importante*. Envolva-se com as histórias. Crie suas próprias legendas para as fotos. Reclamar de uma piada ruim é melhor do que não sentir absolutamente nada.

10 Digite e execute o código.
Digite e execute os exemplos de código. Em seguida, você poderá fazer testes alterando e aperfeiçoando o código (ou interrompendo-o, o que às vezes é a melhor maneira de descobrir o que está realmente acontecendo). Em exemplos longos de códigos predefinidos, você pode fazer o download dos arquivos-fonte a partir de jeadjavafirst.com

Requisitos deste livro:

Você *não* precisa de nenhuma outra ferramenta de desenvolvimento, como um ambiente de desenvolvimento integrado (IDE, Integrated Development Environment). Recomendamos que *não* use nada a não ser um editor de texto básico até concluir a leitura (e *principalmente* não antes do Capítulo 16). Um IDE pode lhe proteger de alguns dos detalhes que são muito importantes, portanto, você se sairá muito melhor aprendendo na linha de comando e, em seguida, após ter compreendido realmente o que está acontecendo, passe para uma ferramenta que automatize parte do processo.

Configurando o Java

- Se você já não tiver um **SDK** (Software Development Kit) **Java 2 Standard Edition 1.5** ou superior, precisará dele. Se estiver trabalhando no Linux, Windows ou Solaris, poderá adquiri-lo gratuitamente em java.sun.com (site Web para desenvolvedores Java). Geralmente não são necessários mais do que dois cliques na página principal para que a página de downloads do J2SE seja acessada. Capture a última versão *não-beta* publicada. O SDK inclui tudo que você precisará para compilar e executar o Java.

Se você estiver executando o Mac OS X 10.4, o SDK Java já estará instalado. Ele faz parte do OS X e você não terá que fazer mais *nada*. Se estiver com uma versão anterior do OS X, terá uma versão desatualizada do Java que servirá para 95% dos códigos deste livro.

Nota: este livro foi baseado no Java 1.5, mas por razões desconhecidas de marketing, logo após o lançamento, a Sun a renomeou como Java 5, embora tenha mantido "1.5" como o número da versão no kit do desenvolvedor. Portanto, se você se deparar com Java 1.5, Java 5, Java 5.0 ou "Tiger" (pseudônimo original da versão 5), *são todas a mesma coisa*. Nunca houve um Java 3.0 ou 4.0 – ele saltou da versão 1.4 para a 5.0, mas você ainda encontrará locais onde é chamado de 1.5 em vez de 5. Não pergunte por quê. (Ah, e apenas para deixar a situação mais divertida, tanto o Java 5 quanto o Mac OS X 10.4 receberam o mesmo pseudônimo "Tiger", e já que o OS X 10.4 é a versão do Mac OS necessária à execução do Java 5, você ouvirá pessoas falando sobre "Tiger em Tiger". Isso significa apenas Java 5 no OS X 10.4.)

- O SDK *não* inclui a **documentação do API** e você precisa dela! Acesse novamente java.sun.com e capture a documentação do API J2SE. Você também pode acessar os documentos do API on-line, sem fazer o download, mas isso é complicado. Acredite. Melhor fazer o download.

- Você precisará de um **editor de texto.** Praticamente qualquer editor de texto pode ser usado (vi, emacs, pico), inclusive os de GUI que vêm com a maioria dos sistemas operacionais. O Bloco de Notas, WordPad, TextEdit, etc., todos servirão, contanto que você se certifique de que eles não acrescentem um ".txt" ao final de seu código-fonte.

- Quando você tiver feito o download e **descompactado, compactado** ou seja lá o que for preciso (depende de que versão e para qual sistema operacional), terá que adicionar uma entrada para sua variável de ambiente **PATH**, que apontará para o diretório bin dentro do diretório Java principal. Por exemplo, se o J2SDK inserir um diretório em sua unidade de disco chamado "j2sdk1.5.0", olhe dentro desse diretório e você encontrará o diretório "bin" onde os arquivos binários (as ferramentas) do Java residem. O diretório bin é aquele para o qual você precisará de uma variável PATH, para que, quando digitar:

```
% javac
```

na linha de comando, seu terminal saiba como encontrar o compilador *javac*.

Nota: se você tiver problemas com sua instalação, recomendamos que acesse javaranch.com e se associe ao fórum Java-Beggining! Na verdade, você deve fazer isso, tendo ou não problemas.

Nota: grande parte dos códigos deste livro estão disponíveis em wickedlysmart.com

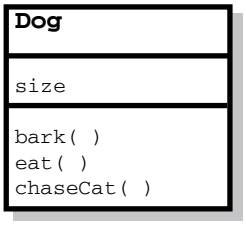
Coisas de última hora que você precisa saber:

Esta é uma experiência de aprendizado e não uma obra de referência. Eliminamos deliberadamente tudo que pudesse atrapalhar o *aprendizado*, independentemente do que estivéssemos abordando em um certo ponto do livro. E, na primeira leitura, é preciso estudar desde o início, porque o livro faz suposições sobre o que você já viu e aprendeu.

Usamos diagramas simples semelhantes à UML.

Se tivéssemos usado UML *pura*, você veria algo *parecido* com Java, mas com sintaxe totalmente *errada*. Portanto, usamos uma versão simplificada de UML que não entra em conflito com a sintaxe do Java. Se você ainda não conhece UML, não terá que se preocupar em aprender Java *e* UML ao mesmo tempo.

Usamos uma UML fictícia modificada e mais simples



Não nos preocupamos com a organização e o empacotamento de seu código até o fim do livro.

Neste livro, você pode dar prosseguimento à tarefa de aprender Java, sem se preocupar com alguns dos detalhes organizacionais e administrativos do desenvolvimento de programas em Java. No dia-a-dia, você *terá* que conhecer – e usar – esses detalhes, portanto, eles foram abordados cuidadosamente. Mas deixamos para o fim do livro (Capítulo 17). Relaxe enquanto aprecia o Java, tranqüilamente.

Você deve executar TODAS as atividades Aponte seu lápis

Os exercícios de fim de capítulo são obrigatórios; os quebra-cabeças são opcionais. As respostas dos dois estão no fim de cada capítulo.

Uma coisa que você precisa saber sobre os quebra-cabeças – eles são enigmas. Como nos enigmas lógicos, nos estimuladores cerebrais, nas palavras cruzadas, etc. Os *exercícios* estão aqui para ajudar você a praticar o que aprendeu, e é recomendável fazê-los. Os quebra-cabeças são diferentes e alguns deles são bem desafiadores de uma maneira *enigmática*. Esses quebra-cabeças foram projetados para funcionar como *decifradores de enigmas* e, provavelmente, você saberá quando encontrar um. Se não tiver certeza, sugerimos que tente fazer alguns deles, mas, independentemente do que acontecer, não desanime se não *conseguir* resolver um quebra-cabeça ou se simplesmente não tiver tempo para isso.

As atividades marcadas com o logotipo Exercício (tênis de corrida) são obrigatórias! Não deixe de executá-las, se quiser mesmo aprender Java.

Os exercícios "Aponte Seu Lápis" não têm respostas.

Pelo menos não impressas neste livro. Para alguns deles, não *há* resposta correta e, em outros, parte da experiência de aprendizado da atividade será *você* decidir se e quando suas repostas estão corretas. (Algumas de nossa respostas *sugeridas* estão disponíveis em wickedlysmart.com.)

Se você se deparar com o logotipo Quebra-Cabeças, a atividade é opcional e, se não for um apreciador de lógica traiçoeira ou palavras cruzadas, também não gostará dessas atividades.

Os exemplos de código são tão simples quanto possível

É frustrante percorrer 200 linhas de código procurando pelas duas linhas que você precisa entender. A maioria dos exemplos deste livro é mostrada no contexto mais simples possível, para que a parte que você estiver tentando aprender fique clara e fácil. Portanto, não espere que o código seja robusto ou mesmo completo. Isso vai depender de *você* depois que terminar o livro. Os exemplos do livro foram escritos especificamente para o *aprendizado* e nem sempre funcionam perfeitamente.

Editores técnicos

"Todos merecem crédito, mas os erros são responsabilidade apenas do autor..." Alguém acredita nisso? Estão vendo as duas pessoas que se encontram nesta página? Se você encontrar problemas técnicos, provavelmente será culpa *delas*. :)

O MINI de Jessica

Jessica Sant

Valentin Crettaz

A gravata de Valentin

Jess trabalha na Hewlett-Packard na Equipe de Serviços de Auto-Reparo. Ela se formou em Engenharia da Computação na Villanova University, tem os certificados SCPJ 1.4 e SCWCD e se passaram literalmente vários meses desde que recebeu seu diploma de Mestre em Engenharia de Softwares na Drexel University (uau!).

Quando não está trabalhando, estudando ou dirigindo seu MINI Cooper S, Jess pode ser encontrada brigando com seu gato pelo novelo enquanto conclui seu último projeto de tricô ou crochê (alguém aí quer um gorro?). Ela é originalmente de Salt Lake City, Utah (não, ela não é mórmon... É claro, você deve ter pensado nisso) e atualmente vive perto da Filadélfia com seu marido, Mendra, e dois gatos: Chai e Sake.

Você pode encontrá-la moderando fóruns técnicos em javaranch.com.

Valentin Valentin Crettaz tem o diploma de Mestre em Ciência da Informação e Computação do Instituto Federal Suíço de Tecnologia em Lausanne. (EPFL). Ele trabalhou como engenheiro de softwares com a SRI International (Menlo Park, CA) e como engenheiro chefe no Laboratório de Engenharia de Softwares do EPFL.

Valentin é co-fundador e CTO da Condris Technologies, uma empresa especializada no desenvolvimento de soluções de arquitetura de softwares.

Seus interesses em pesquisa e desenvolvimento incluem tecnologias orientadas à apresentação, padrões de projeto e arquitetura, serviços Web e arquitetura de software. Além de cuidar de sua esposa, praticar jardinagem, ler e fazer algum esporte, Valentin é moderador dos fóruns SCBCD e SCDJWS no Javaranch.com. Ele tem os certificados SCJP, SCJD, SCBSC, SCWCD e SCDJWS. Também teve a oportunidade de ser o co-autor do Simulador de Exames SCBCD do Whizlabs.

(Ainda estamos chocados em vê-lo de *gravata*).

Outras pessoas que merecem (culpa) crédito:

Na O'Reilly:

Muito obrigado a **Mike Loukides** da O'Reilly, por participar deste projeto e ajudar a formar o conceito *Use a Cabeça!* em um livro (e *série*). Quando esta segunda edição foi para a gráfica, já havia cinco livros *Use a Cabeça!* e ele esteve conosco em todos. A **Tim O'Reilly**, por se dispor a entrar em algo *completamente* novo e diferente. Obrigado ao inteligente **Kyle Hart**, por descobrir como a série *Use a Cabeça!* poderia servir aos outros e por lançá-la. Para concluir, a **Eddie Freedman,** por projetar a capa da série "com ênfase na cabeça".

Nossos intrépidos testadores beta e equipe de revisores:

Agradecemos muito ao diretor de nossa equipe técnica no javaranch, **Johannes de Jong**. Esta é sua quinta vez trabalhando conosco em um livro *Use a Cabeça!* e estamos felizes por você ainda estar em contato. **Jeff Cumps** já está em seu terceiro livro conosco e continua implacável em achar áreas onde teríamos que ser mais claros ou corretos.

Corey McGlone,você é ótimo. E achamos que deu as explicações mais claras sobre o javaranch. Você deve perceber que roubamos uma ou duas delas. **Jason Menard** nos salvou tecnicamente em mais do que apenas alguns detalhes, e **Thomas Paul**, como sempre, nos forneceu feedback especializado e encontrou os sutis problemas do Java que o resto de nós deixou passar. **Jane Griscti** conhece Java (e sabe alguma coisa de *redação*) e foi ótimo tê-la ajudando na nova edição junto com o associado de longa data do javaranch **Barry Gaunt**.

Mailyn de Queiroz nos deu excelente suporte nas *duas* edições do livro. **Chris Jones, John Nyquist, James Cubeta, Terri Cubeta** e **Ira Becker** nos foram de grande ajuda na primeira edição.

Agradecimentos especiais a alguns dos envolvidos na série *Use a Cabeça!* que nos têm ajudado desde o início: **Angelo Celeste, Mikalai Zaikin** e **Thomas Duff** (twduff.com). E obrigado a nosso fantástico agente, David Rogelberg do StudioB (mas, falando sério, e quanto aos direitos do *filme*?)

Alguns de nossos revisores especialistas em Java...

Jef Cumps

Johannes de Jong

Jason Menard

Thomas Paul

Marilyn de Queiroz

Rodney J. Woodruff

Terri Cubeta

James Cubeta

Ira Becker

John Nyquist

Chris Jones

Quando você pensou que não haveria mais agradecimentos*.

Mais especialistas técnicos em Java que ajudaram na primeira edição (em ordem semialeatória):

Emiko Hori, Michael Taupitz, Mike Gallihugh, Manish Hatwalne, James Chegwidden, Shweta Mathur, Mohamed Mazahim, John Paverd, Joseph Bih, Skulrat Patanavanich, Sunil Palicha, Suddhasatwa Ghosh, Ramki Srinivasan, Alfred Raouf, Angelo Celeste, Mikalai Zaikin, John Zoetebier, Jim Pleger, Barry Gaunt e Mark Dielen.

A equipe dos quebra-cabeças da primeira edição:

Dirk Schrekmann. Mary "Campeã de Cruzadas em Java" Leners, Rodney J. Woodruff, Gavin Bong e Jason Menard. O Javaranch tem sorte por contar com seu apoio.

Outros conspiradores a agradecer:

Paul Wheaton, o principal orientador do javaranch por dar suporte a milhares de aprendizes de Java.

Solveig Haugland, instrutora de J2EE e autora de *Dating design patterns* (Encontrando-se com os padrões de projeto).

Os autores **Dori Smith** e **Tom Negrino (backupbrain.com)**, por nos ajudarem a conhecer o mundo dos livros técnicos.

Nossos parceiros no crime *Head First*, **Eric Freeman e Beth Freeman** (autores de *Use a Cabeça! Design Patterns*), por disponibilizarem a Bawls™ para que terminássemos a tempo.

Sherry Dorris, por tudo que realmente importa.

Os bravos pioneiros que adotaram a série Use a Cabeça!:

Joe Litton, Ross P. Goldberg, Dominic Da Silva, *honestpuck*, Danny Bromberg, Stephen Lepp, Elton Hughes, Eric Christensen, Vulinh Nguyen, Mark Rau, Abdulhaf, Nathan Oliphant, Michael Bradly, Alex Darrow, Michael Fischer, Sarah Nottingham, Tim Allen, Bob Thomas e Mike Bibby (o primeiro).

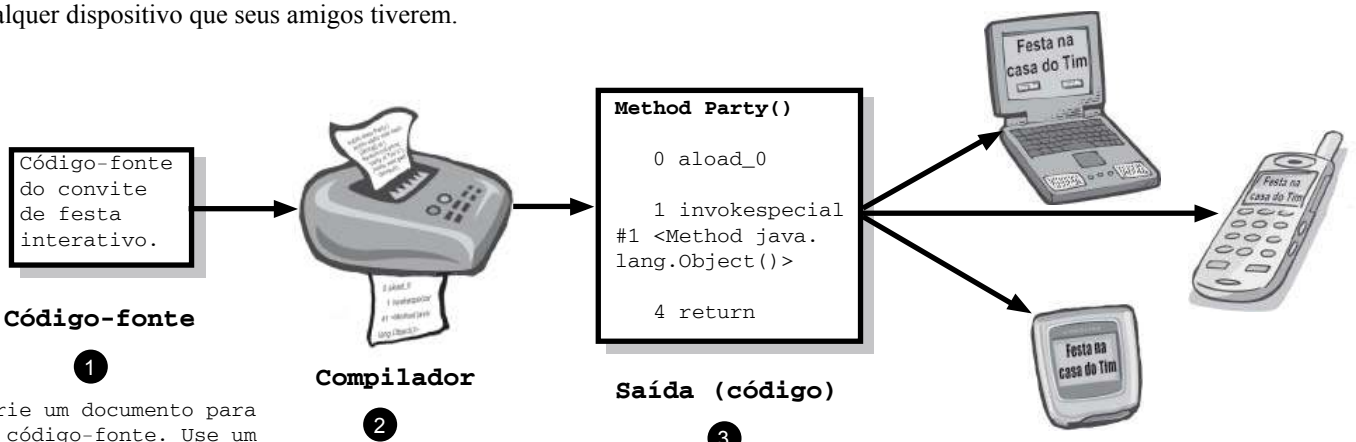
* O grande número de agradecimentos se deve ao fato de estarmos testando a teoria de que todas as pessoas mencionadas nos agradecimentos de um livro comprarão pelo menos uma cópia, provavelmente mais, pensando nos parentes e conhecidos. Se você quiser estar nos agradecimentos de nosso *próximo* livro e tiver uma família grande, nos escreva.

Aprofundando-se

O Java o levará a novas fronteiras. No humilde lançamento para o público como a (suposta) versão 1.02, o Java seduziu os programadores com sua sintaxe amigável, recursos orientados a objetos, gerenciamento de memória e, o melhor de tudo — a promessa de portabilidade. A possibilidade de **escrever uma vez/ executar em qualquer local** exerce uma atração muito forte. Seguidores devotados surgiram, enquanto os programadores combatiam os erros, limitações e, ah sim, o fato de ela ser muito lenta. Mas isso foi há muito tempo. Se você for iniciante em Java, **tem sorte**. Alguns de nós tiveram que passar por algo como andar quase dez quilômetros na neve e subir montanhas pelos dois lados (descalços), para fazer até mesmo o applet mais simples funcionar. Mas você pode manipular o **mais fácil, rápido e muito mais poderoso Java atual.**

Como o Java funciona

O objetivo é escrever um aplicativo (neste exemplo, um convite de festa interativo) e fazê-lo funcionar em qualquer dispositivo que seus amigos tiverem.



Código-fonte

1

Crie um documento para o código-fonte. Use um protocolo estabelecido (nesse caso, a linguagem Java).

Compilador

2

Execute seu documento em um compilador de código-fonte. O compilador procurará erros e não deixará você compilar até ter certeza de que tudo será executado corretamente.

Saída (código)

3

O compilador criará um novo documento, codificado em ***bytecode*** Java. Qualquer dispositivo capaz de executar Java conseguirá interpretar/converter esse arquivo em algo que possa processar. O bytecode compilado é independente da plataforma.

Máquinas virtuais

4

Seus amigos não têm uma máquina Java física, mas todos têm uma máquina Java ***virtual*** (implementada em software) sendo executada dentro de seus aparelhos eletrônicos. A máquina virtual lerá e *executará* o bytecode.

O que você fará em Java

Você criará um arquivo de código-fonte, compilará usando o compilador javac e, em seguida, executará o bytecode compilado em uma máquina virtual Java.

Código-fonte

1

Digite seu código-fonte. Salve como: ***Party.java***

Compilador

2

Compile o arquivo ***Party.java*** executando o javac (o aplicativo do compilador). Se não houver erros, você terá um segundo documento chamado ***Party.class***
O arquivo Party. class gerado pelo compilador é composto de *bytecodes*.

Saída (código)

3

Código compilado: ***Party. class***

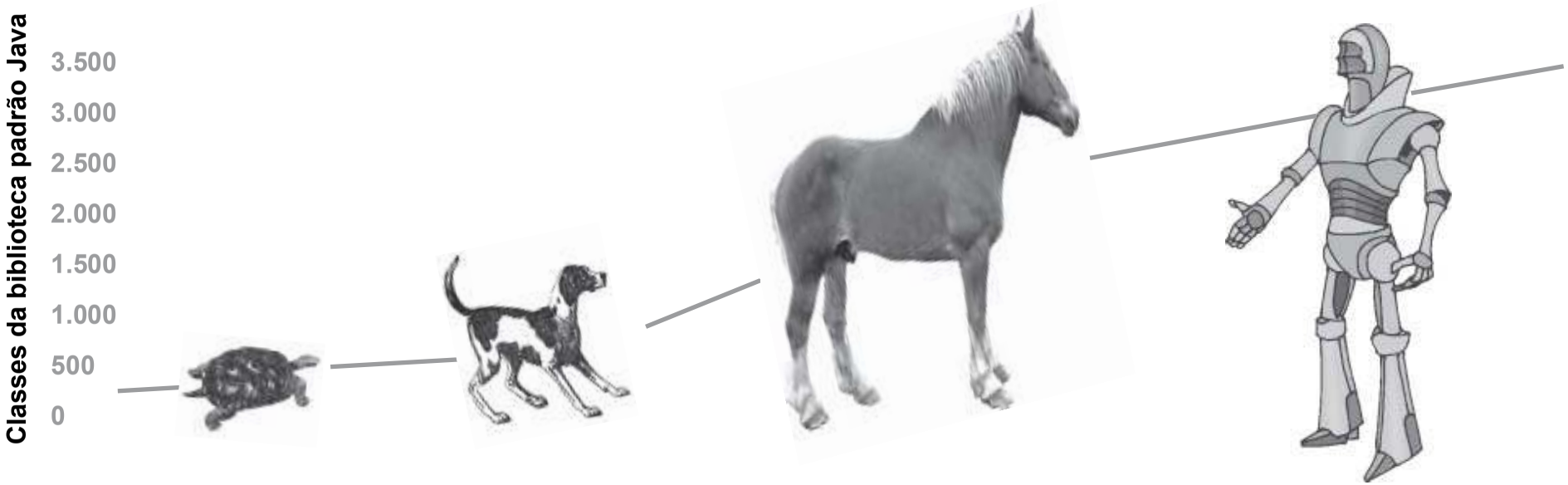
Máquinas virtuais

4

Execute o programa iniciando a Java Virtual Machine (JVM) com o arquivo ***Party.class***. A JVM converterá o *bytecode* em algo que a plataforma subjacente entenda e executará seu programa.

(Nota: não pretendemos que essas instruções sejam um tutorial... Você vai escrever código real em breve, mas, por enquanto, queremos apenas que tenha uma idéia de como tudo se encaixa.)

Um histórico bem resumido do Java



Java 1.02

250 classes

Lenta.

Nome e logotipo interessantes. Divertida de usar. Muitos erros. Os ***applets*** são o destaque.

Java 1.1

500 classes

Um ***pouco*** mais rápida.

Mais recursos, ***mais amigável***. Começando a se tornar muito ***popular***. Código de GUI mais adequado.

Java 2 (versões 1.2 — 1.4)

2.300 classes

Muito mais rápida.

Pode (em algumas situações) ser executada em velocidades condizentes. Profissional, ***poderosa***. Vem em três versões: Micro Edition (J2ME), Standard Edition (J2SE) e Enterprise Edition (J2EE). Torna-se a ***linguagem preferida*** para novos aplicativos empresariais (principalmente os baseados na Web) e móveis.

Java 5.0 (versões 1.5 e posteriores)

3.500 classes

Mais recursos, mais fácil de desenvolver.

Além de adicionar mais de mil classes complementares, a Java 5.0 (conhecida como "Tiger") acrescentou alterações significativas à própria linguagem, tornando-a mais fácil (pelo menos em teoria) para os programadores e fornecendo novos recursos que eram populares em outras linguagens.

Aponte seu lápis

Tente adivinhar o que cada linha de código está fazendo...
(As respostas estão na próxima página.)

Veja como é fácil escrever código Java.

```
int size = 27;'
String name = "Fido";
Dog myDog = new Dog(name, size);
x = size - 5;
if (x < 15) myDog.bark(8);

while (x > 3) {
   myDog.play();
}

int[] numList = {2,4,6,8};
System.out.print("Hello");
System.out.print("Dog: " + name);
String num = "8";
int z = Integer.parseInt(num);

try {
   readTheFile("myFile.txt");
}
catch(FileNotFoundException ex) {
System.out.print("File not found.");
}
```

declara uma variável de tipo inteiro chamada size e lhe atribui o valor 27

P: Sei que existem o Java 2 e o Java 5.0, mas existiram o Java 3 e 4? E por que Java 5.0 e não Java 2.0?

R: As brincadeiras do marketing... Quando a versão do Java passou de 1.1 para 1.2, as alterações foram tão significativas, que os anunciantes decidiram que precisavam de um "nome" totalmente novo, portanto .começaram a chamá-la de Java 2, ainda que a versão fosse realmente a 1.2. Porém, as versões 1.3 e 1.4 continuaram a ser consideradas como Java 2. Nunca houve o Java 3 ou 4. Começando pelo Java versão 1.5, os anunciantes decidiram novamente que as alterações eram tão significativas, que um novo nome era necessário (e a maioria dos desenvolvedores concordou), logo, eles avaliaram as opções. O próximo número na seqüência do nome seria "3", mas chamar o Java 1.5 de Java 3 parecia mais confuso, portanto, decidiram nomeá-lo Java 5.0 para usar o "5" da versão "1.5".

Logo, o Java original compreendeu as versões que iam da 1.02 (o primeiro lançamento oficial) às conhecidas simplesmente como "Java". As versões 1.2, 1.3 e 1.4 consistiram no "Java 2". E começando na versão 1.5, ele passou a se chamar "Java 5.0". Mas você também o verá sendo chamado de "Java 5" (sem o ".0") e "Tiger" (seu codinome original). Não temos idéia do que acontecerá com a próxima versão...

Ainda não é preciso se preocupar em entender tudo isso!

Tudo que se encontra aqui é explicado com maiores detalhes no livro, grande parte nas primeiras 40 páginas. Se o Java lembrar uma linguagem que você usou no passado, alguns desses itens parecerão simples. Caso contrário, não se preocupe com isso. *Chegaremos lá...*

Veja como é fácil escrever código Java.

`int size = 27;`	*declara uma variável de tipo inteiro chamada 'size' e lhe atribui o valor 27*
`String name = "Fido";`	*declara uma variável de string de caracteres chamada 'name' e lhe atribui o valor "Fido"*
`Dog myDog = new Dog(name, size);`	*declara a nova variável de tipo Dog chamada 'myDog' e cria o novo objeto Dog usando 'name' e 'size'*
`x = size - 5;`	*subtrai 5 de 27 (valor de 'size') e atribui o valor a uma variável chamada 'x'*
`if (x < 15) myDog.bark(8);`	*se x (valor = 22) for menor do que 15, informa ao cão (dog) para latir (bark) 8 vezes*
`while (x > 3) {`	*mantém o loop até x ser maior que 3...*
`  myDog.play();`	*pede ao cão que brinque (independentemente do que ISSO signifique para um cão...)*
`}`	*aqui parece ser o fim do loop — tudo que estiver entre { } será feito no loop*
`int[] numList = {2,4,6,8};`	*declara a lista de variáveis de tipo inteiro 'numList' e insere 2, 4, 6, 8 nela*
`System.out.print("Hello");`	*exibe "Hello"... provavelmente na linha de comando*
`System.out.print("Dog: " + name);`	*exibe "Hello Fido" (o valor de 'name' é "Fido") na linha de comando*
`String num = "8";`	*declara a variável de string de caracteres 'num' e lhe atribui o valor "8"*
`int z = Integer.parseInt(num);`	*converte a string de caracteres "8" no valor numérico real 8*
`try {`	*tenta fazer algo... Pode ser que o que estamos tentando não funcione...*
`  readTheFile("myFile.txt");`	*lê um arquivo de texto chamado "myFile.txt" (ou pelo menos TENTA ler o arquivo...)*
`}`	*deve ser o fim das "tentativas", portanto, acho que é possível tentar fazer muitas coisas...*
`catch(FileNotFoundException ex) {`	*aqui deve ser onde você saberá se o que tentou não funcionou...*
`System.out.print("File not found.");`	*se o que tentamos não deu certo, exibiremos "File not found" na linha de comando*
`}`	*parece que tudo que se encontra entre { } é o que deve ser feito se a 'tentativa' não funcionar...*

Estrutura do código em Java

O que existe em um arquivo-FONTE?

Um arquivo de código-fonte (com a extensão *.java*) contém uma definição de ***classe***. A classe representa uma *parte* de seu programa, embora um aplicativo muito pequeno possa precisar apenas de uma classe. A classe deve ficar dentro de uma par de chaves.

```
public class Dog{

}
```

classe

O que existe em uma CLASSE?

Uma classe tem um ou mais ***métodos***. Na classe Dog, o método ***bark*** conterá instruções de como o cão deve latir. Seus métodos devem ser declarados *dentro* de uma classe (em outras palavras, dentro das chaves da classe).

```
public class Dog {
   void bark( ) {

}
}
```

método