Design Patterns com Java

Projeto orientado a objetos guiado por padrões

Casa do Código

EDUARDO GUERRA

Casa do Código
Livros para o programador
Rua Vergueiro, 3185 - 8º andar
04101-300 – Vila Mariana – São Paulo – SP – Brasil

Design Patterns com Java: Projeto orientado a objetos guiado por padrões / Eduardo Guerra
São Paulo: Casa do Código, 2013
ISBN 978-85-66250-11-4

Agradecimentos Pessoais

Meu primeiro agradecimento é a Deus, por Ele ter me proporcionado tudo para que trilhasse o caminho que trilhei até chegar onde estou agora. Por ter colocado as pessoas certas para me ajudar em minhas dificuldades, por ter me colocado obstáculos nas horas adequadas que me permitiram amadurecer para superar obstáculos maiores no futuro e por ter me iluminado no momento das decisões mais críticas que acabaram direcionando minha vida no futuro.

Em seguida envio meu "muito obrigado" para meus pais, José Maria e Maria da Graça, que sempre me incentivaram e me apoiaram em minhas escolhas. Mesmo estando distantes fisicamente hoje, sei que todos os dias eles rezam e torcem por mim. Imagino direitinho eles mostrando esse livro para todo mundo com o maior orgulho! Aproveito essa oportunidade para estender esse agradecimento para toda minha família, em especial para minha Tia Dorinha que sempre esteve ao meu lado quando precisei.

Agradeço também a Maria Eduarda, a Duda, minha filha mais velha, por trazer só alegria desde que chegou nesse mundo. Seu jeito amigo e carinhoso me conforta sempre que por algum motivo estou para baixo. Não sei como tanta bondade e paciência pode caber em apenas uma pessoa. Sou muito grato pelo seu companheirismo, como quando ela senta do meu lado bem agarradinha para assistir um filme ou jogar um videogame.

Seguindo para a mais nova da família, agradeço a Ana Beatriz, a Bia, por conter a maior concentração de energia e alegria de todo mundo. Apesar de ser bem bagunceira e gostar de um alvoroço, ela sempre faz alguma coisa que consegue tirar uma gargalhada de nossa boca nas horas mais inesperadas. É até difícil de descrever a alegria que sinto quando eu chego e essa baixinha vem gritando o meu nome e dá um abraço na minha perna.

Deixei para o fim os agradecimentos para minha companheira de toda vida, minha esposa Roberta, ou, como eu a chamo, Lindinha. Acho que ela não tem noção do

amor que sinto por ela e como sua presença é central em minha vida. Eu agradeço a ela por através de suas ações ter se tornado a pessoa mais importante da minha vida. Sem ela eu seria incompleto e não teria estrutura para ter trilhado o caminho que segui em minha vida até esse momento. Muito obrigado por todo seu apoio, por todo seu amor e por todo seu carinho!

Agradecimentos Profissionais

Antes de mencionar alguém de forma específica, gostaria de agradecer a todos meus professores, colegas de trabalho, amigos e companheiros que de alguma forma contribuíram para meu conhecimento e me incentivaram a seguir em frente. Nossa vida é feita de pessoas e de experiências, e é a partir disso que seguimos nossos caminhos e realizamos nossas escolhas.

Primeiramente deixo um agradecimento institucional ao ITA, um instituto que é referência nacional e internacional em engenharia, em onde me formei em Engenharia da Computação e fiz meu mestrado e doutorado. Além disso, também foi um local onde tive a oportunidade de atuar como professor por mais de 5 anos. Foram experiências que me marcaram e são uma importante parte da minha história. Eu saí do ITA, mas com certeza o ITA não saiu de mim! Agradeço em especial ao professor Clovis Fernandes, meu orientador na graduação, no mestrado e no doutorado. Um grande amigo, que sempre me incentivou e que foi grande mentor, cujos ensinamentos foram muito além de questões técnicas. Espero ainda fazermos muitas coisas juntos!

Agradeço a revista MundoJ, principalmente ao Marco Guapo, inicialmente por ter aberto espaço para que escrevesse sobre minhas ideias e meus conhecimentos. Em seguida, Guapo confiou a mim o conteúdo da revista como editor-chefe. Através da minha atuação na revista eu pude aprender muita coisa e me manter sempre atualizado durante os últimos anos. Através dos artigos que escrevi ganhei experiência e tomei gosto pela escrita de conteúdo técnico. Posso afirmar com certeza que a semente desse livro foi plantada em alguns de meus artigos durante os últimos anos. Espero que essa parceria ainda dure bastante tempo!

Agradeço também a comunidade de padrões, tanto nacional quanto internacional, por ter me influenciado com toda sua cultura e ideias. Sou grato por terem me recebido de braços abertos e por terem fornecido um forum onde foi possível debater as minhas ideias e obter feedback dos trabalhos que estou realizando. Agra-

deço em especial pelo apoio e incentivo de Fabio Kon, Fábio Silveira, Jefferson Souza, Uirá Kulezsa, Ademar Aguiar e Filipe Correia. Também agradeço pelos meus gurus Joseph Yoder e Rebecca Wirfs-Brock, com quem aprendi demais e tive discussões muito interessantes sobre modelagem e arquitetura de software.

Agradeço ao INPE, instituição onde acabei de ingressar, por ter me recebido de braços abertos e pela confiança que tem depositado em meu trabalho. Espero poder dar muitas contribuições e desenvolver diversos trabalhos interessantes durante os anos que estão por vir!

Finalmente agradeço a Casa do Código pelo convite para escrita desse livro. Apesar da vontade sempre ter existido, o convite de vocês foi o estopim para que esse projeto se tornasse algo concreto. Deixo um agradecimento em especial ao editor desse livro, Paulo Silveira, pelos questionamentos e sugestões que contribuíram de forma definitiva para o conteúdo desse livro.

Minibiografia do Autor

Eduardo Martins Guerra nasceu em Juiz de Fora em 1980 e cursou o ensino básico e médio em escola pública, o Colégio de Aplicação João XXIII. Desde essa época ele já despertou seu interesse pela programação, começando com a digitação de código Basic que vinha em revistas no seu defasado computador Exato Pro. A diversão era mais ver o efeito de pequenas modificações no código do que o jogo que saia como resultado. Nessa época, pouco depois, também fez um curso de Clipper, onde desenvolveu seu primeiro software: um gerenciador de canil!

Incentivado por seus professores, pela facilidade com disciplinas na área de exatas, prestou o vestibular para o ITA em 1998, logo após o término de seus estudos, e foi aprovado. Durante o curso, ele optou pela carreira militar e pelo curso de computação, além de participar de atividades extracurriculares como projetos para a empresa júnior e aulas no curso pré-vestibular para pessoas carentes, CASD Vestibulares. Nesses cinco anos, Eduardo se apaixonou pela área de desenvolvimento de software e pela possibilidade de usar constantemente sua criatividade para propor sempre soluções inteligentes e inovadoras. Dois dias depois de se formar, casou-se com sua mais forte paixão, sua esposa atual, Roberta.

Após formado, em 2003, foi alocado no Centro de Computação da Aeronáutica de São José dos Campos (CCA-SJ), onde ficaria pelos próximos 4 anos. Durante esse período, trabalhou com o desenvolvimento de software operacional para atender as necessidades da Força Aérea Brasileira, adquirindo uma valiosa experiência na área.

No final de 2005, com dedicação em tempo parcial, Eduardo conseguiu seu título de mestre em Engenharia da Computação e Eletrônica apresentando a dissertação "Um Estudo sobre Refatoração de Código de Teste". Nesse trabalho, Eduardo desenvolve sua paixão pelo design de software, realizando um estudo inovador sobre boas práticas ao se codificar testes automatizados. Essa dissertação talvez tenha sido um dos primeiros trabalhos acadêmicos no Brasil no contexto de desenvolvimento ágil. No ano de conclusão de seu mestrado, veio sua primeira filha, Maria Eduarda.

Devido sua sede de conhecimento, ainda nesse período, estudou por conta própria e tirou sete certificações referentes a tecnologia Java, sendo na época um dos profissionais brasileiros com maior número de certificações na área. Além disso, ocupa o cargo de editor-chefe da revista MundoJ desde 2006, na qual já publicou dezenas de artigos técnicos de reconhecida qualidade. Esses fatos lhe deram uma visão da indústria de desenvolvimento de software que foi muito importante em sua trajetória, direcionando sua pesquisa e seus trabalhos para problemas reais e relevantes.

No CCA-SJ, Eduardo atuou de forma decisiva no desenvolvimento de frameworks. Seus frameworks simplificaram a criação das aplicações e deram maior produtividade para a equipe de implementação. Um deles, o SwingBean, foi transformado em um projeto open-source e já possui mais de 5000 downloads. A grande inovação e diferencial de seus frameworks estava no fato de serem baseados em metadados. A partir desse caso de sucesso, ele decidiu estudar mais sobre como a utilização de metadados poderia ser feita em outros contextos. Foi uma surpresa descobrir que, apesar de outros frameworks líderes de mercado utilizarem essas técnicas, ainda não havia nenhum estudo sobre o assunto. Foi, então, que Eduardo começou sua jornada para estudar e tornar acessível o uso dessa técnica por outros desenvolvedores, pois ele sabia do potencial que os frameworks baseados em metadados tem para agilizar o desenvolvimento de software e o tornar mais flexível.

Então, em 2007, ele ingressou no curso de doutorado do ITA pelo Programa de Pós-Graduação em Aplicações Operacionais – PPGAO. A pesquisa sobre frameworks baseados em metadados se encaixava perfeitamente no objetivo do programa. A criação de arquiteturas flexíveis, nas quais se pode acrescentar funcionalidade de forma mais fácil, é algo crítico para aplicações de comando e controle, foco de uma das áreas do programa. Orientado pelo professor Clovis Torres Fernandes, começou uma pesquisa que envolveu a análise de frameworks existentes, a abstração de técnicas e práticas, a criação de novos frameworks de código aberto e a execução de experimentos para avaliar os conceitos propostos.

Foi nessa época que Eduardo começou a participar e se envolveu na comunidade de padrões. Em 2008 submeteu para o SugarLoafPLoP em Fortaleza os primeiros padrões que identificou em frameworks que utilizavam metadados. Nesse momento ele se empolgou com o espírito de colaboração da comunidade de padrões, onde recebeu um imenso feedback do trabalho que estava realizando. Desde então se tornou um membro ativo da comunidade, publicando artigos com novos padrões em eventos no Brasil e no exterior. Além disso, em 2011 participou da organização do

MiniPLoP Brasil, em 2012 foi o primeiro brasileiro a ser chair da principal conferência internacional de padrões, o PLoP, e em 2013 está também participando também da organização do próximo MiniPLoP.

Enquanto realizava seu doutorado, Eduardo foi incorporado, ainda como militar, no corpo docente do Departamento de Ciência da Computação do ITA, onde pôde desenvolver uma nova paixão: ensinar! Durante esse período, ministrou aulas na graduação e em cursos de especialização, e orientou uma série de trabalhos que, de certa forma, complementavam e exploravam outros aspectos do que estava desenvolvendo em sua tese. Em consequência do bom trabalho realizado, foi convidado por três vezes consecutivas para ser o professor homenageado da turma de graduação Engenharia da Computação e duas vezes para a turma de especialização em Engenharia de Software.

Em 2010, apresentou seu doutorado chamado "A Conceptual Model for Metadata-based Frameworks e concluiu com sucesso essa jornada que deixou diversas contribuições. Devido a relevância do tema, foi incentivado pelos membros da banca a continuar os estudos que vem realizando nessa área. Apesar da tese ter trazido grandes avanços, ele tem consciência que ainda há muito a ser feito. Uma de suas iniciativas nessa área foi o projeto Esfinge (http://esfinge.sf.net) , que é um projeto guarda-chuva para a criação de diversos frameworks com essa abordagem baseada em metadados. Até o momento já existem três frameworks disponíveis e vários artigos científicos em cima de inovações realizadas nesses projetos. No mesmo ano que terminou seu doutorado, veio sua segunda filhinha, a Ana Beatriz.

Em 2012, Eduardo prestou concurso para o Instituto Nacional de Pesquisas Espaciais, onde assumiu o cargo de pesquisador no início 2013. Hoje ele segue com sua pesquisa na área de arquitetura, testes e design de software, buscando aplicar as técnicas que desenvolve para projetos na área espacial. Adicionalmente, atua como docente na pós-graduação de Computação Aplicada desse instituto, onde busca passar o conhecimento que adquiriu e orientar alunos para contribuírem com essas áreas.

Por que mais um livro de padrões?

O primeiro livro de padrões "Design Patterns: Elements of Reusable Object-Oriented Software", conhecido como Gang of Four (GoF), já foi lançado a mais de 15 anos e de forma alguma seu conteúdo está ultrapassado. Em minha opinião, esse livro foi algo muito a frente de seu tempo, sendo que muitas de suas práticas só foram ser utilizadas de forma efetiva pela indústria alguns anos depois. Durante esses anos, diversos livros sobre esses padrões foram escritos, apresentando sua implementação em outras linguagens ou ensinando-os de forma mais didática. Desse contexto podem surgir as seguintes perguntas: será que é preciso mais um livro sobre padrões? O que esse livro trás de diferente?

Durante esses últimos anos tive diversos tipos de experiência que me fizerem ter diferentes visões sobre os padrões. Dentre essas experiências eu posso citar o ensino de modelagem de software e padrões, a utilização de padrões para o desenvolvimento de frameworks e aplicações, a identificação de novos padrões a partir do estudo de implementações existentes e discussões na comunidade sobre padrões e sua aplicabilidade. A partir disso acredito que tive a oportunidade de ter uma visão diferenciada sobre esses padrões, e é essa visão que procuro passar nesse livro. As seções a seguir descrevem alguns diferenciais desse livro em relação a outros existentes.

Uma Sequência Didática para Aprendizado

Percebi que muitos cursos sobre padrões, sendo eles dados por universidades ou empresas de treinamento, vão ensinando os padrões um por um. Mas qual é a melhor ordem para o ensino e a assimilação dos padrões? Isso era algo que era decidido por cada professor. Uma ordem inadequada nesse aprendizado pode impedir que o aluno não compreenda os princípios por trás da solução do padrão, o que dificulta sua compreensão a respeito de sua aplicabilidade e suas consequências. Isso também pode causar uma visão limitada da solução do padrão, o que impede sua adaptação para outros contextos similares.

Esse livro procura organizar os padrões em uma sequência didática para o aprendizado. Cada capítulo agrupa os padrões pelo tipo de técnica que utilizam em sua solução ou pelo tipo de problema que resolvem. Dessa forma, ao ver uma mesma técnica ser utilizada em padrões diferentes é possível compreender melhor a sua dinâmica e de que formas diferentes ela pode ser utilizada. Em capítulos que focam em tipos de problema, a visão das diferentes alternativas de solução e das suas diferenças, cria um senso crítico a respeito das possíveis alternativas de modelagem e quais os critérios que devem ser considerados para sua escolha.

Isso faz com que esse livro vá além da apresentação dos padrões e seja na verdade a respeito de técnicas para modelagem orientada a objetos. Os padrões são utilizados como uma referência para a discussão de cada uma dessas técnicas. Dessa forma, esse livro é recomendado para utilização como material base em cursos de graduação ou pós-graduação a respeito de técnicas de programação orientada a objetos ou programação orientada a objetos avançada.

Relação entre Padrões

Outro problema que percebo é que os padrões tendem a ser olhados de forma individual. Isso é uma consequência do formato que os padrões possuem e do fato de serem autocontidos, ou seja, toda informação sobre o padrão estar contida em sua descrição. Isso por um lado é bom pois permite que alguém obtenha todas as informações sobre um determinado padrão em apenas um local. Porém, por outro lado, perde-se um pouco a ideia a respeito do relacionamento entre os padrões. Apesar de haver uma seção que descreve os padrões relacionados, ainda é uma descrição pequena para uma questão de tamanha importância.

Apresentar o relacionamento entre os padrões foi um dos objetivos desse livro. Apesar de cada padrão apresentar uma solução independente, é da combinação entre eles que é possível criar uma sinergia para criação de soluções ainda melhores. Dessa forma, a medida que o livro vai seguindo, e novos padrões vão sendo apresentados, existe uma preocupação em mostrar como eles podem ser combinados com os padrões anteriores ou como eles podem ser utilizados como uma alternativa a um outro padrão. Esse tipo de discussão é importante, pois além de saber a solução do padrão, também é essencial saber quando aplicá-la.

Refatoração para Padrões

A modelagem de uma aplicação é uma questão dinâmica que evolui a medida

que o desenvolvimento do software vai seguindo seu curso. Apesar de ainda ser comum a modelagem das classes da aplicação a partir de diagramas antes do inicio da implementações, muitas vezes os problemas aparecem somente depois. Sendo assim, tão importante quanto saber como utilizar os padrões na modelagem inicial, é saber como refatorar um código existente para uma solução que utilize um determinado padrão. A refatoração constante do código é hoje uma das bases para a manutenção de sua qualidade ao longo do projeto.

Dessa forma, além de apresentar os padrões, esse livro também se preocupa em mostrar quais os passos que precisariam ser feitos para refatorar um código existente em direção a um determinado padrão. Este assunto, que já foi tema de um livro dedicado somente a ele, aqui é apresentado de forma fluida juntamente com os padrões. Isso é importante para que o leitor possa ter uma visão, não apenas estática em relação a utilização de um padrão, mas de como um código existente pode evoluir e ser melhorado a partir de sua introdução a partir de pequenos passos.

Exemplos Realistas e do Dia-a-dia

Por mais que diagramas sejam úteis para que se tenha uma visão geral de uma solução, os desenvolvedores ainda tem muito a cultura do "*show me the code*", querendo ver na prática como um padrão pode ser implementado. Por mais que o GoF tenha sido um livro a frente de seu tempo, ele utiliza exemplos que hoje estão distantes de um desenvolver comum, como a criação de uma suite de componentes gráficos ou um editor de texto. Alguns outros livros acabam trazendo "*toy examples*", que seriam exemplos simples e didáticos, mas que são fora do contexto de aplicações reais.

Acho muito importante que o leitor se identifique com os exemplos e faça uma ligação com o tipo de software que desenvolve. Ao se ver próximo aos exemplos e possuir uma experiência anterior com aquele tipo de problema, é muito mais fácil compreender as questões que estão envolvidas e as alternativas de solução com suas respectivas consequências. Nesse livro, busquei trabalhar com exemplos recorrentes em aplicações desenvolvidas por grande parte dos programadores, como geração de relatórios, carrinho de compras em aplicações de e-commerce e geração de arquivos para integração entre aplicações. Acredito que assim ficará bem mais claro como cada um poderá utilizar esses padrões em seu dia-a-dia.

Dicas e Referências para Implementações em Java

Muitos padrões apresentados por esse livro são amplamente utilizados em fra-

meworks e APIs Java. Muitas vezes, apenas por seguir as regras de uma determinada API, sem saber você já está utilizando um determinado padrão. A flexibilidade que aquele framework consegue para ser instanciado na sua aplicação, muitas vezes é conseguido justamente pelo uso do padrão! O conhecimento de uma situação em que aquele padrão foi utilizado e, muitas vezes sem saber, você acabou o utilizando, também ajuda muito em sua compreensão.

Dessa forma, esse livro também cita diversas classes de APIs padrão da linguagem Java e de frameworks amplamente utilizados pela comunidade de desenvolvimento. Isso vai permitir que o desenvolvedor possa compreender melhor aquela solução inteligente utilizada naquele contexto, e trazer a mesma ideia para questões do seu próprio código.

Além disso, como o próprio nome do livro diz, os padrões de projeto são apresentados na linguagem Java e, dessa forma, acabam trazendo algumas dicas mais específicas da linguagem para sua implementação. Essas dicas vão desde a utilização dos próprios recursos da linguagem, como a utilização de sua biblioteca de classes e, até mesmo, de frameworks externos.

Lista de Discussão

O design de software é um tema pelo qual me apaixonei justamente por ser algo em que o uso da criatividade é essencial, e cada aplicação tem suas particularidades, o que sempre traz novas questões a serem discutidas. O mesmo ocorre com padrões! Eles já estão aí há muitos anos e até hoje ainda existe muita discussão sobre eles! Sendo assim, eu gostaria que esse livro não fosse apenas uma comunicação de mão única, onde eu escrevo e vocês leem, mas o início de um canal de comunicação para fomentar discussões e conversas sobre padrões e design de software em geral.

Sendo assim, criei uma lista de discussão com o nome **"Design Patterns em Java: Projeto orientado a objetos guiado por padrões"** no endereço **`projeto-oo-guiado-por-padroes@googlegroups.com`**.

Se você quer discutir, colocar suas dúvidas e saber eventos e novidades no mundo dos padrões, deixo aqui o meu convite para a participação no grupo!

Prefácio

Por Joseph Yoder

Já faz cerca de 20 anos que o **Gang of Four** escreveu o livro inicial de padrões chamado **"Design Patterns: Elements of Reusable Object-Oriented Software"**. Desde aquela época, os padrões de projeto se tornaram muito conhecidos e uma parte essencial de uma boa modelagem em sistemas orientados a objetos. Adicionalmente, os padrões ganharam uma grande aceitação na comunidade de desenvolvimento de software, o que pode ser observado pela existência de diversas publicações e casos de sucesso. Existem também diversas conferências ao redor do mundo criadas aos moldes da primeira conferência sobre padrões, a **Pattern Languages of Programs** (**PLoP**).

Mesmo nos anos iniciais do Java, é possível observar como os padrões influenciaram a linguagem. Algumas dessas influências podem ser vistas claramente nas primeiras APIs do Java, como as interfaces `Iterator` e `Observer`, enquanto outras foram implementadas como parte da evolução dessas APIs, como a implementação dos **listeners**, que são uma implementação mais atual do padrão `Observer`.

Porém, é importante notar que um bom design não acontece por acidente, pois ele exige trabalho duro e evolução. De forma complementar, também não quer dizer que usando um padrão necessariamente se tem um bom design. Por exemplo, a linguagem Java foi lançada apenas alguns anos depois da publicação do primeiro livro de design patterns. Por causa disso, muitas das primeiras bibliotecas da linguagem foram influenciadas pelo livro e muitos padrões foram incorporados no design de suas principais bibliotecas e frameworks. No entanto, isso nem sempre guiou as soluções para o melhor design e foram necessárias várias evoluções em suas classes depois que diversos problemas foram encontrados nas primeiras versões.

Um bom exemplo pode ser visto no tratamento de eventos do Java AWT. No AWT 1.0, o tratamento de eventos utilizava o padrão **`Chain of Responsibility`**. A princípio essa parecia uma solução adequada ao olhar para os

requisitos. O problema com essa implementação era que precisava-se da utilização de um **`Mediator`** ou da criação de diversas subclasses. Isso poderia causar problemas de desempenho, visto que para o evento ser tratado era necessário percorrer toda a cadeia para descobrir qual era a classe que deveria processá-lo. Isso muitas vezes direcionava o desenvolvedor a criação de uma solução muito complexa devido ao grande número de subclasses que precisava criar.

A partir disso, o AWT 1.1 passou a tratar os eventos de forma mais eficiente utilizando os padrões **`Observer`** e **`Adapter`**. Essa modelagem evoluiu depois para a utilização de listeners. Essa acabou sendo uma solução muito mais limpa e eficiente, visto que apenas as classes interessadas no evento eram notificadas quando ele acontecia. Essa história claramente mostra que o design de um software deve levar em consideração as consequências positivas e negativas de cada decisão, e só porque ele utiliza padrões não quer dizer que aquela é a melhor alternativa de design. Veja que isso não quer dizer que utilizar padrões leva a decisões erradas de design, mas que, pelo contrário, a utilização de um outro padrão mais adequado se mostrou uma solução melhor.

Apenas saber **como** aplicar os padrões não é o suficiente para um bom design. Entender quando e onde aplicá-los é tão importante quanto, senão mais importante. De fato, no decorrer do tempo, mesmo um bom design inicial pode ser comprometido por sucessivas revisões arquiteturais. Em 1998, foi feita a afirmação que a arquitetura que realmente predominava na prática era a **`Big Ball of Mud`** (traduzindo Grande Bola de Lama). E mesmo com muito trabalho e dedicação ainda é possível acabar com um **`Big Ball of Mud`** se você não está comprometido a manter o código sempre limpo.

Existem muitas forças e bons motivos que podem levar a uma arquitetura extremamente complexa. De fato, arquitetos e times de desenvolvimento experientes estão constantemente fazendo exatamente o que deveria ser feito quando se deparam com "lama" e complexidade desnecessária em seu sistema.

Quando se tenta manter uma arquitetura, é importante tentar proteger certas partes do design. Grandes sistemas possuem partes que mudam em diferentes velocidades. Existem certas ações e padrões que você pode utilizar para isolar e definir divisões em volta dessas diferentes partes, tanto para tornar a arquitetura estável, quanto para possibilitar as mudanças onde elas são necessárias. De fato, essa é uma premissa importante dos padrões de projeto. Isso se torna uma consideração ainda mais importante quando evoluímos a estrutura das aplicações para um mundo onde parte delas está localizada na nuvem.

Eu conheço o autor desse livro (Eduardo) a muitos anos. Eduardo tem grande experiência em padrões, design orientado a objetos e Java. Eu já colaborei e trabalhei com ele em projetos orientados a objetos e publicações, incluindo a implementação de diversos padrões de projeto e a construção de frameworks. Eduardo tem muita experiência na construção de frameworks, atividade que exige um conhecimento profundo de padrões, além de quando e onde utilizá-los. Ele tem mais do que uma compreensão de "como" implementar um padrão em Java. Eduardo claramente compreende o que está em jogo em um bom design orientado a objetos e como utilizar os padrões para um design mais limpo, mais reutilizável e mais fácil de mudar.

Qualquer desenvolvedor para se tornar proficiente em um design orientado a objetos precisará compreender não apenas o básico sobre padrões, mas também princípios mais avançados a respeito de suas consequências e o contexto em que devem ser utilizados. Este livro é mais do que um livro de receitas sobre como aplicar os padrões, pois, pelo contrário, ele traz também importantes princípios e se aprofunda nessas técnicas avançadas de design. O leitor irá descobrir nesse livro um guia eficiente a respeito de como aplicar os padrões, como escolher o padrão mais adequado e como saber quando um design deve ser refatorado para a introdução de um novo padrão.

Após ler esse livro, é possível continuar essa jornada sobre o conhecimento de padrões através de conferências sobre o assunto, como o SugarLoafPLoP no Brasil. Este assunto tem se tornado de extrema importância para todas as facetas dos sistemas de software, principalmente aqueles que estão em constante evolução para se manterem atualizados frente a novos requisitos.

Sumário

1 Entendendo Padrões de Projeto **1**

1.1 Conceitos da Orientação a Objetos 2
1.2 Mas esses conceitos não são suficientes? 7
1.3 O Primeiro Problema: Cálculo do Valor do Estacionamento 8
1.4 Strategy: o Primeiro Padrão! 14
1.5 O que são padrões? 15
1.6 Como o Livro Está Organizado 18

2 Reuso Através de Herança **21**

2.1 Exemplo de Padrão que Utiliza Herança - Null Object 22
2.2 Hook Methods 26
2.3 Revisando modificadores de métodos 28
2.4 Passos diferentes na Mesma Ordem - Template Method 29
2.5 Refatorando na Direção da Herança 35
2.6 Criando Objetos na Subclasse - Factory Method 39
2.7 Considerações Finais do Capítulo 43

3 Delegando Comportamento com Composição **45**

3.1 Tentando Combinar Opções do Gerador de Arquivos 46
3.2 Bridge - Uma Ponte entre Duas Variabilidades 48
3.3 Hook Classes 53
3.4 State - Variando o Comportamento com o Estado da Classe 56
3.5 Substituindo Condicionais por Polimorfismo 63
3.6 Compondo com Múltiplos Objetos - Observer 66
3.7 Considerações Finais do Capítulo 74

4 Composição Recursiva **77**
4.1 Compondo um Objeto com sua Abstração 78
4.2 Composite - Quando um Conjunto é um Indivíduo 81
4.3 Encadeando Execuções com Chain of Responsibility 87
4.4 Refatorando para Permitir a Execução de Múltiplas Classes 93
4.5 Considerações Finais do Capítulo 96

5 Envolvendo Objetos **99**
5.1 Proxies e Decorators 100
5.2 Exemplos de Proxies 106
5.3 Extraindo um Proxy 111
5.4 Adaptando Interfaces 114
5.5 Considerações Finais do Capítulo 120

6 Estratégias de Criação de Objetos **121**
6.1 Limitações dos Construtores 122
6.2 Criando Objetos com Métodos Estáticos 124
6.3 Um Único Objeto da Classe com Singleton 127
6.4 Encapsulando Lógica Complexa de Criação com Builder 130
6.5 Relacionando Famílias de Objetos com Abstract Factory 136
6.6 Considerações Finais do Capítulo 140

7 Modularidade **141**
7.1 Fábrica Dinâmica de Objetos 142
7.2 Injeção de Dependências 148
7.3 Service Locator 156
7.4 Service Locator versus Dependency Injection 161
7.5 Considerações Finais do Capítulo 163

8 Adicionando Operações **165**
8.1 Classes que Representam Comandos 166
8.2 Cenários de Aplicação do Command 173
8.3 Double Dispatch - Me chama, que eu te chamo! 179
8.4 Padrão Visitor 184
8.5 Considerações Finais do Capítulo 196

9 Gerenciando Muitos Objetos **199**

9.1 Criando uma Fachada para suas Classes 200
9.2 Separando Código Novo de Código Legado 207
9.3 Mediando a Interação entre Objetos 208
9.4 Reaproveitando Instâncias com Flyweight 215
9.5 Considerações Finais do Capítulo 225

10 Indo Além do Básico **227**

10.1 Frameworks 228
10.2 Utilizando Tipos Genéricos com os Padrões 233
10.3 Padrões com Test-driven Development 236
10.4 Posso Aplicar esses Padrões na Arquitetura? 241
10.5 Comunidade de Padrões 243
10.6 E agora? 245

Índice Remissivo **248**

Bibliografia **251**

Versão: 17.1.16