Someone struggles

Jonathan Ball

ungovernable press
2009

cover image

Orlando. Photo: Jonathan Ball

Product Description

You're not alone.

At any given moment, somewhere in the world someone struggles with the same software design problems you have. You know you don't want to reinvent the wheel (or worse, a flat tire), so you look to Design Patterns—the lessons learned by those who've faced the same problems. With Design Patterns, you get to take advantage of the best practices and experience of others, so that you can spend your time on . . . something else. Something more challenging. Something more complex. Something more fun.

You want to learn about the patterns that matter—why to use them, when to use them, how to use them (and when NOT to use them). But you don't just want to see how patterns look in a book, you want to know how they look "in the wild." In their native environment. In other words, in real world applications. You also want to learn how patterns are used in the Java API, and how to exploit Java's built-in pattern support in your own code.

You want to learn the real OO design principles and why everything your boss told you about inheritance might be wrong (and what to do instead). You want to learn how those principles will help the next time you're up a creek without a design pattern.

Most importantly, you want to learn the "secret language" of Design Patterns so that you can hold your own with your co-worker (and impress cocktail party guests) when he casually mentions his stunningly clever use of Command, Facade, Proxy, and Factory in between sips of a martini. You'll easily counter with your deep understanding of why Singleton isn't as simple as it sounds, how the Factory is so often misunderstood, or on the real relationship between Decorator, Facade and Adapter.

With *Head First Design Patterns*, you'll avoid the embarrassment of thinking Decorator is something from the "Trading Spaces" show. Best of all, in a way that won't put you to sleep! We think your time is too important (and too short) to spend it struggling with academic texts.

If you've read a Head First book, you know what to expect—a visually rich format designed for the way your brain works. Using the latest research in neurobiology, cognitive science, and learning theory, *Head First Design Patterns* will load patterns into your brain in a way that sticks. In a way that lets you put them to work immediately. In a way that makes you better at solving software design problems, and better at speaking the language of patterns with others on your team.

The Process

Be wrong (and what of patterns with others NOT to use them). Someone struggles somewhere in the world science, and learning theory, and Adapter. With Head First applications. You you want to learn the book, you want them to work immediately. And Adapter. With Head First when to use them, how Design Patterns, you'll avoid at speaking the language deep understanding of why to do instead). You want you don't want to learned by those learned by those else. Something more patterns—the lessons You want to learn the matter—why to use them, on your team. The next time you're will load patterns into your is so often misunderstood. Most importantly, more complex. You have. You know a design paddle pattern. Matter—why to use them, patterns look in when to use them, how Head First book, you know of Design Patterns so brain in a way that sticks. And how to exploit advantage your time on . . . something want to see how challenging. Something (and impress cocktail party guests)of patterns with others the patterns that Java's built-in pattern Head First book, you know it struggling with academic what to expect—a visually-rich environment. In other also want to learn. In a way that lets you put at speaking the language deep understanding of why better at solving software you want to learn the Singleton isn't as simple as it. With Design Patterns, else. Something more of patterns with others. Decorator is something from a book, you want more complex. You have. You know support in your own code. Learned by those put you to sleep! We think you have. You know, and how to exploit NOT to use them). Put you to sleep! We think. But you don't just Head First Design Patterns else. Something more. In a way that lets you put. You want to learn the used in the Java API of the best practices the latest research in be wrong (and what You'll easily counter with your (and too short) to spend). Best of all, in a way that won't. Java's built-in pattern you don't want to. Most importantly, on your team. When to use them, how the next time you're applications. You format designed for the way what to expect—a visually-rich the next time you're and experience of others, neurobiology, cognitive want to see how the patterns that deep understanding of why you get to take principles will help. You want to learn about, and how to exploit will load patterns into your when to use them, how be wrong (and what patterns—the lessons same problems. And experience of others, what to expect—a visually-rich environment. In other the "Trading Spaces" show. Something more fun. You want to learn the how patterns are more complex. To do instead). You want real OO design principles. In a way that makes you with someone struggles advantage challenging. Something in between sips of a martini. To learn how those what to expect—a visually-rich. Best of all, in a way that won't sounds, how the Factory you want to learn the patterns, the lessons of the best practices. And how to exploit at speaking the language them to work immediately. Deep understanding of why and why everything, and how to exploit in between sips of a martini. It struggling with academic is so often misunderstood, you don't want to. In a way that makes you deep understanding of why same problems. The next time you're someone struggles look "in the wild." Put you to

sleep! We think. You want to learn about to learn how those and experience of others, applications. You you have. You know own with your co-worker same problems. You're not Decorator is something from his stunningly clever use of Command, else. Something more when to use them, how. You want to learn about. You'll easily counter with your advantage so you look to Design and experience of others, in a way that lets you put between Decorator, Facade or on the real relationship. Most importantly, and Adapter. With Head First. Something more fun. To use them (and when Design Patterns, you'll avoid and experience of others, the patterns that you don't want to. Singleton isn't as simple as it better at solving software. In a way that makes you and Adapter. With Head First the latest research in also want to learn to learn how those. Best of all, in a way that won't own with your co-worker matter—why to use them, about inheritance might Java's built-in pattern applications. You sounds, how the Factory your time is too important neurobiology, cognitive will load patterns into your up a creek without with own with your co-worker. Most importantly, (and too short) to spend somewhere in the world in between sips of a martini. Your time on . . . something you have. You know. Best of all, in a way that won't. Something more fun. Them to work immediately. At speaking the language more complex. Texts. If you've read a (and impress cocktail party guests) your time is too important and Adapter. With Head First. You'll easily counter with your the patterns that design problems, and better between Decorator. Facade words, in real world you have. You know support in your own code. Look "in the wild." And Adapter. With Head First you don't want to your boss told you the patterns that format designed for the way. With Design Patterns, Singleton isn't as simple as it principles will help deep understanding of why matter—why to use them, more complex. Else. Something more with who've faced the put you to sleep! We think of the best practices want to see how on your team. Environment. In other with. Something more fun. About inheritance might learned by those your brain works. Using put you to sleep! We think. But you don't just Java's built-in pattern be wrong (and what to know how they Head First design. Patterns your brain works. Using someone struggles. Patterns—the lessons support in your own code. Design. Patterns, you'll avoid (or worse, a flat tire), patterns look in applications. You. Most importantly, else. Something more. Something more fun. Java's built-in pattern of patterns with others and Adapter. With Head First look "in the wild." You want to learn about. Best of all, in a way that won't your time on . . . something you have. You know your time is too important learned by those reinvent the wheel you have. You know. Decorator is something from what to expect—a visually-rich book, you want. With Design Patterns, the latest research in about inheritance might your brain works. Using. In their native look "in the wild." Used in the Java API patterns look in look "in the wild." Else. Something more will load patterns into your. You want to learn the. You want to learn the that you can hold your at speaking the language that you can hold your. But you don't just. Applications. You sounds, how the Factory design problems principles will help alone. At any given moment. In their native design problems more complex. Also want to learn. You're not to learn how those who've

faced the on your team. You want to learn the texts. If you've read a deep understanding of why "secret language" when to use them, how. You want to learn the or on the real relationship the "Trading Spaces" show. Decorator is something from the embarrassment of thinking matter—why to use them, Java's built-in pattern (with who've faced the better at solving software who've faced the to do instead). You want own (with your co-worker to do instead). You want the patterns that. You're not a book, you want want to see how in between sips of a martini. Best of all, in a way that won't challenging. Something challenging. Something to learn how those. You'll easily counter with your of Design Patterns so somewhere in the world learned by those used in the Java API "secret language" (or worse, a flat tire), you get to take. Most importantly, reinvent the wheel. Head First Design Patterns up a creek without between Decorator, Facade matter—why to use them, or on the real relationship neurobiology, cognitive it struggling with academic you don't want to want to see how in between sips of a martini. Will load patterns into your matter—why to use them, sounds, how the Factory somewhere in the world and why everything. In a way that makes you sounds, how the Factory Singleton isn't as simple as neurobiology, a cognitive book, you want who've faced the "secret language" NOT to use them). You'll easily counter with your it struggling with academic when he casually mentions Singleton isn't as simple as it your brain works. Using put you to sleep! We think that you can (hold your to do instead). You want advantage at speaking the language neurobiology, cognitive your time on . . . something words, in real world. You want to learn the same problems. Is so often misunderstood, when to use them, how someone struggles is so often misunderstood, the next time you're about inheritance might Java's built-in pattern when to use them, how "secret language." With Design Patterns, be wrong (and what texts. If you've read a on your team. Or worse, a flat tire), Java's built-in pattern Head First book, you know on your team. Real OO design principles about inheritance might to know how they patterns look in real OO design principles else. Something more will load patterns into your of the best practices with. Something more fun. Sounds, how the Factory Façade, Proxy, and Factory the same software patterns look in. Something more fun. And Adapter. With Head First the patterns that. Best of all, in a way that won't your time on . . . something (and impress cocktail party guests) design problems, and better who've faced the them to work immediately. Brain in a way that sticks. It struggling with academic when to use them, how reinvent the wheel you have. You know You'll easily counter with your when to use them, how matter—why to use them, his stunningly clever use of Command, learned by those what to expect—a visually-rich. Something more fun. Texts. If you've read a Head First Design Patterns you have. You know your boss told you and experience of others, own with your co-worker the embarrassment of thinking in between sips of a martini. Own with your co-worker up a creek without. With Design Patterns, you want to learn the "secret language." In a way that makes you you have. You know design problems, and better to learn how those Design Patterns, you'll avoid with. You'll easily counter with your advantage sounds, how the Factory you want to learn (his stunningly clever use of

Command, learned by those patterns look in Design Patterns, you'll avoid) you want to learn the Head First Design Patterns that you can hold your the "Trading Spaces" show. How patterns are design problems matter—why to use them, his stunningly clever use of Command, used in the Java API more complex. But you don't just learned by those so you look to Design how patterns are. Something more fun. Else. Something more somewhere in the world matter—why to use them, used in the Java API of the best practices advantage someone struggles of patterns with others your brain works. Using. In their native you don't want to. You're not alone.