If on a winter's night a programmer (with apologies to Italo Calvino) You've just picked up the new book by Greg Meredith, Pro Scala. Perhaps you've heard about it on one of the mailing lists or seen it advertised on the `Scala` site or at Amazon. You're wondering if it's for you. Maybe you've been programming in functional languages or even `Scala` for as long as you can remember. Or maybe you've been a professional programmer for quite some time. Or maybe you're a manager of programmers, now and you're trying to stay abreast of the latest technology. Or, maybe you're a futurologist who looks at technology trends to get a sense of where things are heading. Whoever you are, if you're like most people, this book is going to make a lot more sense to you if you've already got about five to ten thousand hours of either `Scala` or some other functional language programming under your belt [1]. There may be nuggets in here that provide some useful insights for people with a different kind of experience; and, of course, there are those who just take to the ideas without needing to put in the same number of hours; but, for most, that's probably the simplest gauge of whether this book is going to make sense to you at first reading.

On the other hand, just because you've got that sort of experience under your belt still doesn't mean this book is for you. Maybe you're just looking for a few tips and tricks to make `Scala` do what you want for the program you're writing right now. Or maybe you've got a nasty perf issue you want to address and are looking here for a resolution. If that's the case, then maybe this book isn't for you because this book is really about a point of view, a way of looking at programming and computation. In some sense this book is all about programming and complexity management because that's really the issue that the professional programmer is up against, today. On average the modern programmer building an Internet-based application is dealing with no less than a dozen technologies. They are attempting to build applications with nearly continuous operation, 24x7 availability servicing 100's to 1000's of concurrent requests. They are overwhelmed by complexity. What the professional programmer really needs are tools for complexity management. The principle aim of this book is to serve that need in that community.

The design patterns expressed in this book have been developed for nearly *fifty* years to address exactly those concerns. Since `Scala` isn't nearly fifty years old you can guess that they have origins in older technologies, but `Scala`, it turns out, is an ideal framework in which both to realize them and to talk about their ins and outs and pros and cons. However, since they don't originate in `Scala`, you can also

[1] Now, i've been told that this is too much to expect of a would-be reader; but, when i whip out my calculator, i see that (5000 hrs / 25 hrs/wk) / 52 wks/yr) = 3.84615384615 years. That means that if you've put in under four years at a hobbyist level, you've met this requirement. Alternatively, if you've put in less than two years as a professional working solely in functional languages, you've met the requirement. Honestly, we don't have to give in to inflationary trends in the meanings of terms. If we say something is aimed at a pro, we could mean what we say.

guess that they have some significant applicability to the other eleven technologies the modern professional programmer is juggling.