Wrapping Up

Fluency can make or break a language. The flexibility Kotlin provides to create fluent code is one of the reasons why programmers love the language. By using the highly contentious feature of operator overloading cautiously, we can provide concise operators for user-defined classes where it makes sense. With the ability to use extension functions, properties, and operators, we can add our own convenience members to third-party classes to make code more intuitive to use. This is possible even when those classes aren't open for inheritance. The infix notation removes the noise of dot and parenthesis. Along with implicit receivers on lambdas, these features greatly influence the power of creating internal DSLs in Kotlin.

You'll see the fluency of Kotlin shine and many of the features in this chapter come to life when you learn to create internal DSLs in the next chapter.