Wrapping Up

The functional style of programming is less complex when compared to the imperative style of programming. Using higher-order functions and functional composition, we can write fluent code that is easier to understand and to maintain. Lambdas are functions with no name and may be easily passed around as arguments to functions. Kotlin provides a variety of options to write lambdas. Where a lambda is expected, you may also pass function references to reuse functions or methods. Whereas lambdas are stateless, closures carry state. However, avoid messing with mutable state as that may lead to potential errors and confusing behavior in code. Kotlin has strict rules about using `return` from within lambdas; it permits the use of labeled `returns`, and non-local `returns` under special situations. In addition, Kotlin provides a facility to eliminate function and lambda call overhead with the help of the `inline` keyword.

You've learned how to create lambdas in this chapter. In the next chapter, we'll use lambdas for internal iteration and working with sequences.