3.1 Introduction[1]

[1]. This chapter depends on the terminology and concepts of object-oriented programming introduced in Section 1.5, Introduction to Object Technology.

In Chapter 2, you worked with *existing* classes, objects and methods. You used the *predefined* standard output object `System.out`, *invoking* its methods `print`, `println` and `printf` to display information on the screen. You used the *existing* `Scanner` class to create an object that reads into memory integer data typed by the user at the keyboard. Throughout the book, you'll use many more *preexisting* classes and objects—this is one of the great strengths of Java as an object-oriented programming language.

In this chapter, you'll create your own classes and methods. Each new class you create becomes a new *type* that can be used to declare variables and create objects. You can declare new classes as needed; this is one reason why Java is known as an *extensible* language.

We present a case study on creating and using a simple, real-world bank-account class—`Account`. Such a class should maintain as *instance variables* attributes, such as its `name` and `balance`, and provide *methods* for tasks such as querying the balance (`getBalance`), making deposits that increase the balance (`deposit`) and making withdrawals that decrease the balance (`withdraw`). We'll build the `getBalance` and `deposit` methods into the class in the chapter's examples and you’ll add the `withdraw` method in the exercises.

In Chapter 2, we used the data type `int` to represent integers. In this chapter, we introduce data type `double` to represent an account balance as a number that can contain a *decimal point* —such numbers are called *floating-point numbers*. In Chapter 8, when we get a bit deeper into object technology, we’ll begin representing monetary amounts precisely with class `BigDecimal` (package `java.math`), as you should do when writing industrial-strength monetary applications. [Alternatively, you could treat monetary amounts as whole numbers of pennies, then break the result into dollars and cents by using division and remainder operations, respectively, and insert a period between the dollars and the cents.]

Typically, the apps you develop in this book will consist of two or more classes. If you become part of a development team in industry, you might work on apps that contain hundreds, or even thousands, of classes.