8.7 Notes on *Set* and *Get* Methods

As you know, client code can manipulate a class's `private` fields *only* through the class's methods. A typical manipulation might be the adjustment of a customer's bank balance (e.g., a `private` instance variable of a class `BankAccount`) by a method `computeInterest`. *Set* methods are also commonly called **mutator methods**, because they typically *change* an object's state—i.e., *modify* the values of instance variables. *Get* methods are also commonly called **accessor methods** or **query methods**.

Set and Get Methods vs. `public` Data

It would seem that providing *set* and *get* capabilities is essentially the same as making a class's instance variables `public`. This is one of the subtleties that makes Java so desirable for software engineering. A `public` instance variable can be read or written by any method that has a reference to an object containing that variable. If an instance variable is declared `private`, a `public` *get* method certainly allows other methods to access it, but the *get* method can *control* how the client can access it. For example, a *get*

method might control the format of the data it returns, shielding the client code from the actual data representation. A `public` *set* method can—and should—carefully scrutinize attempts to modify the variable's value and throw an exception if necessary. For example, attempts to *set* the day of the month to 37 or a person's weight to a negative value should be rejected. Thus, although *set* and *get* methods provide access to `private` data, the access is restricted by the implementation of the methods. This helps promote good software engineering.

Software Engineering Observation 8.6

Classes should never have `public` *nonconstant data, but declaring data* `public static final` *enables you to make constants available to clients of your class. For example, class* `Math` *offers* `public static final` *constants* `Math.E` *and* `Math.PI`.

Validity Checking in Set Methods

The benefits of data integrity do not follow automatically simply because instance variables are declared `private`—you must provide validity checking. A class's *set* methods

could determine that attempts were made to assign invalid data to objects of the class. Typically *set* methods have `void` return type and use exception handling to indicate attempts to assign invalid data. We discuss exception handling in detail in Chapter 11.

Software Engineering Observation 8.7

When appropriate, provide `public` *methods to change and retrieve the values of* `private` *instance variables. This architecture helps hide the implementation of a class from its clients, which improves program modifiability.*

Error-Prevention Tip 8.3

Using set *and* get *methods helps you create classes that are easier to debug and maintain. If only one method performs a particular task, such as setting an instance variable in an object, it's easier to debug and maintain the class. If the instance variable is not being set properly, the code that actually modifies instance variable is localized to one* set *method. Your debugging efforts can be focused on that one method.*

Predicate Methods

Another common use for accessor methods is to test whether a condition is *true* or *false*—such methods are often called **predicate methods.** An example would be class `ArrayList`'s `isEmpty` method, which returns `true` if the `ArrayList` is empty and `false` otherwise. A program might test `isEmpty` before attempting to read another item from an `ArrayList`.

Good Programming Practice 8.1

By convention, predicate method names begin with `is` *rather than* `get`.