8.6 Default and No-Argument Constructors

Every class *must* have at least *one* constructor. If you do not provide any in a class's declaration, the compiler creates a *default constructor* that takes *no* arguments when it's invoked. The default constructor initializes the instance variables to the initial values specified in their declarations or to their default values (zero for primitive numeric types, `false` for `boolean` values and `null` for references).

Recall that if your class declares constructors, the compiler will *not* create a default constructor. In this case, you must declare a no-argument constructor if default initialization is required. Like a default constructor, a no-argument constructor is invoked with empty parentheses. The `Time2` no-argument constructor (lines 11–13 of Fig. 8.5) explicitly initializes a `Time2` object by passing to the three-argument constructor 0 for each parameter. Since 0 is the default value for `int` instance variables, the no-argument constructor in this example could actually be declared with an empty body. In this case, each instance variable would receive its default value when the no-argument constructor is called. If we were to omit the no-argument constructor, clients of this class would not be able to create a `Time2` object with the expression `new Time2()`.

Error-Prevention Tip 8.2

Ensure that you do not *include a return type in a constructor definition. Java allows other methods of the class besides its constructors to have the same name as the class and to specify return types. Such methods are* not *constructors and will* not *be called when an object of the class is instantiated.*

Common Programming Error 8.3

A compilation error occurs if a program attempts to initialize an object of a class by passing the wrong number or types of arguments to the class's constructor.