14.2 What Are Beans?

As we said, beans are simply Java classes that are written in a standard format. Full coverage of JavaBeans is beyond the scope of this book, but for the purposes of use in JSP, all you need to know about beans are the three simple points outlined in the following list. If you want more details on beans in general, pick up one of the many books on the subject or see the documentation and tutorials at http://java.sun.com/products/javabeans/docs/.

- **A bean class must have a zero-argument (default) constructor.** You can satisfy this requirement either by explicitly defining such a constructor or by omitting all constructors, which results in a zero-argument constructor being created automatically. The default constructor will be called when JSP elements create beans. In fact, as we see in Chapter 15 (Integrating Servlets and JSP: The Model View Controller (MVC) Architecture), it is quite common for a servlet to create a bean, from which a JSP page merely looks up data. In that case, the requirement that the bean have a zero-argument constructor is waived.

- **A bean class should have no public instance variables (fields).** To be a bean that is accessible from JSP, a class should use accessor methods instead of allowing direct access to the instance variables. We hope you already follow this practice since it is an important design strategy in object-oriented programming. In general, use of accessor methods lets you do three things without users of your class changing their code: (a) impose constraints on variable values (e.g., have the `setSpeed` method of your `Car` class disallow negative speeds); (b) change your internal data structures (e.g., change from English units to metric units internally, but still have `getSpeedInMPH` and `getSpeedInKPH` methods); (c) perform side effects automatically when values change (e.g., update the user interface when `setPosition` is called).

- **Persistent values should be accessed through methods called`get`*`Xxx`* and `set`*`Xxx`*.** For example, if your `Car` class stores the current number of passengers, you might have methods named `getNumPassengers` (which takes no arguments and returns an `int`) and `setNumPassengers` (which takes an `int` and has a `void` return type). In such a case, the `Car` class is said to have a *property* named `numPassengers` (notice the lowercase `n` in the property name, but the uppercase `N` in the method names). If the class has a `get`*`Xxx`* method but no corresponding `set`*`Xxx`*, the class is said to have a read-only property named *`xxx`*.

  The one exception to this naming convention is with boolean properties: they are permitted to use a method called `is`*`Xxx`* to look up their values. So, for example, your `Car` class might have methods called `isLeased` (which takes no arguments and returns a `boolean`) and `setLeased` (which takes a `boolean` and has a `void` return type), and would be said to have a `boolean` property named `leased` (again, notice the lowercase leading letter in the property name).

  Although you can use JSP scriptlets or expressions to access arbitrary methods of a class, standard JSP actions for accessing beans can only make use of methods that use the `get`*`Xxx`*/`set`*`Xxx`* or `is`*`Xxx`*/`set`*`Xxx`* naming convention.