

## Chapter 21 JavaBeans

1. A JavaBeans component is a serializable public class with a default public constructor. Not every GUI class is a JavaBeans component. For example, `java.awt.Color` is not a JavaBeans component. Every GUI user interface class is a JavaBeans component. However, a JavaBeans component may be a non-Swing class.
2. By convention, the accessor method is named `get<PropertyName>()`, which takes no parameters and returns a primitive type value or an object of a type identical to the property type. For a property of `boolean` type, the accessor method should be named `is<PropertyName>()`, which returns a `boolean` value. The mutator method should be named `set<PropertyName>(dataType p)`, which takes a single parameter identical to the property type and returns `void`.
3. By convention, the registration method is named `add<Event>Listener(<Event>Listener listener)` and a deregistration method is named `remove<Event>Listener(<Event>Listener listener)`.
4. Since an event class and its listener interface are coexistent, they are often referred to as an *event set* or *event pair*. The event listener interface must be named as `XListener` for the `XEvent`. For example, the listener interface for `ActionEvent` is `ActionListener`.

You can declare a custom event class by extending `java.util.EventObject` or a subclass of `java.util.EventObject`.

You can declare a custom event listener interface by extending `java.util.EventListener` or a subinterface of `java.util.EventListener`.

5. The Java event model is flexible, allowing modifications and variations. One useful variation of the model is the addition of adapters. When an event occurs, the source object notifies the adapter. The adapter then delegates the handling of the event to the actual processing object, which is referred to as an *adaptee*.
6. See Section 22.8.
7. There are three types of event adapters: standard adapters, inner class adapters, and anonymous adapters.
  - **standard adapter** A named class that extends a convenience listener adapter or implements a listener interface.
  - **inner class adapter** Standard adapters can be shortened using inner classes.
  - **anonymous adapter** **Inner class adapters can be further shortened using anonymous inner classes.**

8. The model-view-controller (MVC) approach is a way of developing components by separating data storage and handling from the visual representation of the data. The component for storing and handling data, known as a *model*, contains the actual contents of the component. The component for presenting the data, known as a *view*, handles all essential component behaviors. It is the view that comes to mind when you think of the component. It does all the displaying of the components. The controller is a component that is usually responsible for obtaining data.
9. The JDK event delegation model provides a superior architecture for supporting MVC component development. The model can be implemented as a source with appropriate event and event listener registration methods. The view can be implemented as a listener. Thus, if data are changed in the model, the view will be notified. To enable the selection of the model from the view, simply add the model as a property in the view with a set method.
10. A common variation of the model-view-controller architecture is to combine the controller with the view. In this case, a view not only presents the data, but is also used as an interface to interact with the user and accept user input.