Технология JavaBeans

Для определения лучше всего привести цитату из [официальной документации](http://download.oracle.com/otn-pub/jcp/7224-javabeans-1.01-fr-spec-oth-JSpec/beans.101.pdf?AuthParam=1446559340_27dcb099d0fc350738d834fa06b1bd73): “A Java Bean is a reusable software component that can be manipulated visually in a builder tool.”

В простейшем случае JavaBean - это отдельный класс, представляющий определенную компоненту. В более сложных случаях - это набор взаимосвязанных классов, каждый из которых играет определенную роль. Так многие классы стандартной библиотеки Java являются бинами, например, JLabel, JTextField и др.

JavaBean может иметь или не иметь графический интерфейс. Он может работать локально или в связке с другими компонентами.

JavaBeans обладают такими отличительными чертами:

* Поддержка самодиагностики (“introspection”). Самодиагностика позволяет окружению, которое использует компонент, определять его характеристики. То есть компонент предоставляет информацию о себе.   
  Это осуществляется двумя способами:
  + Следование соглашениям об именовании переменных и методов;
  + Использованием вспомогательного класса BeansInfo, который предоставляет характеристики явно.
* Для конфигурирования можно применять вспомогательное ПО, которое позволяет настраивать внешний вид (если есть) и поведение компонента;
* Поддержка обработки событий. Компонент может получать события от других объектов и может сообщать о событиях;
* Настройки компонента (его состояние) можно сохранять на постоянном носителе и считывать это состояние при необходимости. Эта возможность достигается имплементацией интерфейса java.io.Serializable.

Чтобы класс мог работать как JavaBean, он должен соответствовать определённым соглашениям об именах методов, конструкторе и поведении. Эти соглашения дают возможность создания инструментов, которые могут использовать, замещать и соединять JavaBeans.

Основные правила такие:

* Класс должен иметь конструктор без параметров, с модификатором доступа public. Такой конструктор позволяет инструментам создать объект без дополнительных сложностей с параметрами;
* Свойства класса должны быть доступны только через getters/setters (так называемые методы доступа), которые должны подчиняться стандартному соглашению об именах. Это легко позволяет инструментам автоматически определять и обновлять содержание JavaBean;
* Класс должен поддерживать сериализацию. Это даёт возможность надёжно сохранять и восстанавливать состояние JavaBean независимым от платформы и виртуальной машины способом;

Сравнение с другими средами

На мой взгляд JavaBeans более всего похожи на компоненты Delphi (за исключением кроссплатформенности). Прямой аналогии в мире С++ и С# нет.

По достигаемым результатам аналогом в мире Windows является технология COM. Но есть существенные различия:

* JavaBeans можно считать частью языка Java, поскольку для работы JavaBeans ничего другого не нужно. Там где есть виртуальная машина там автоматически поддерживается JavaBeans;
* СОМ не связана с каким-либо языком. Компоненты можно писать и использовать на разных языках и средах программирования. Но при этом компоненты используют ОС как посредника для своей работы;