05 - JavaBeans

JavaBeans

A Java Bean is a reusable software component that can be manipulated visually in a builder tool. They are Java classes written in a standardized way and they are used to encapsulate multiple objects in one single object (bean), so that they can be passed more easily. In order for a class to be a bean, it must:

- Have a public default constructor (no arguments);
- Implements the interface `java.io.Serializable`;
- Be in a jar file (archive of java classes) with manifest file containing `Java-Bean: True`.

Features:

- Properties: generally fields of the class, but not always (e.g. class with field a, b and one method getSum(), then the sum is a property, but not a field);
- Events: invoking a *set* method on an object could change the state of other objects, so we might want to generate events to inform the others:
  - Bound property: after it changes inform the others. The generated event is of type `PropertyChangeEvent`;
  - Constrained property: before applying the change, ask the observers for confirms. The generated event is of type `PropertyChangeEvent`, while observers implements the `VetoableChangeListener` interface;
  The event-based communication mechanism is how beans communication happen. It is based on the Observer design pattern, which allows components to communicate in a non-coupled way. In Java, this pattern is based on Events and Event Listeners: the events are objects created by an event source and propagated to the registered event listeners. The semantics is by default multicast, but unicast can be enforced by tagging the event source. There could also be Event Adaptors, objects that behave as listeners towards the source and as sources towards the listeners: they could be useful for asynchronous communication and implementation of filters;
- Customization: GUI to modify the bean, generally done through a builder (tool);
- Persistence: the customized state of a bean can be saved and reloaded later;
- Introspection: a builder tool can analyse the capabilities of the bean (e.g. get the properties). It is based on Java Reflection.

There are guidelines about how to define properties:

- From *get* and *set* methods, we can infer existences of PropertyName of type PropertyType: `public <PropertyType> get<PropertyName>();` - `public void set<PropertyName>(<PropertyType> a);`. Remember that if the property is an array, the methods can take an index or the whole array;
- From *add* and *remove* methods, we can infer existences of Event Generation of name EventListType: `public void add<EventListType>(<EventListType> a)` - `public void remove<EventListType>(<EventListType> a)`. Unicast semantics is assumed if the *add* method is declared to throw `java.util.TooManyListenersException`;