

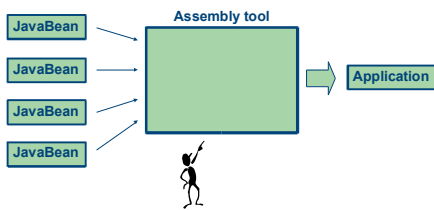
JavaBeans

Jens Gustavsson

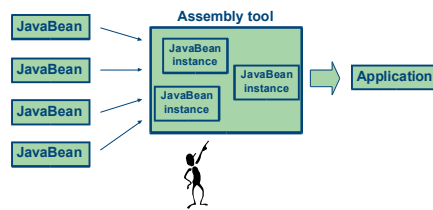
Introduction

- Component model for Java
- A JavaBean is a component
- Visual composition
- Several tool and component developers
- JavaBeans != Enterprise JavaBeans

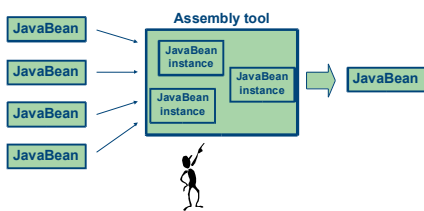
Overview



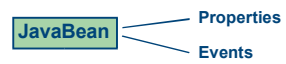
Overview



Overview



Customization and Connections



Example: Alarm Clock

- Properties
 - Current time
 - Alarm time
 - Alarm status (set/not set)
- Events
 - Alarm (source)



What does a bean look like?

- Public constructor without parameters
- Implement interface Serializable
- Design patterns defines properties and events

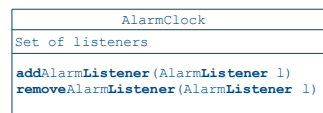
Properties

```
class AlarmClock implements Serializable {
    public AlarmClock() {...}

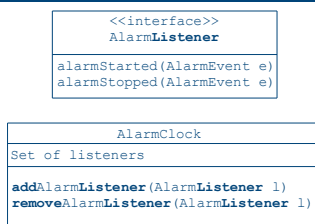
    public boolean getAlarmStatus() {...}

    public void setAlarmStatus(boolean value) {...}
    ...
}
```

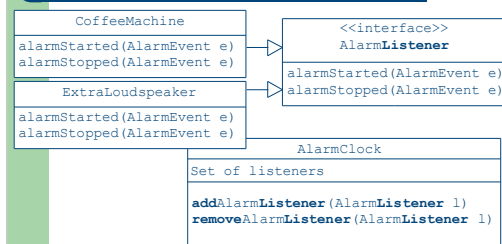
Events



Events



Events



Advanced features

- Optional class: `AlarmClockInfo`
- Optional class: `AlarmClockCustomizer`
- Indexed properties
- Bound properties
- Constrained properties

How does it work?

- Java techniques:
 - Serialization makes persistence possible
 - Dynamic class loading
 - Reflection makes naming convention based scheme possible

Miscellaneous

- Swing components are JavaBeans
- Bridge JavaBean - ActiveX

Evaluation

- Strengths
 - Simple - easy to use
 - Standard - mix vendors
 - Applicable for GUI development
- Weaknesses
 - Only suitable for GUI development
 - Not usable for non-programmers
 - Weak component market