H Java & Component Models

H.1 Overview

- Component models
- Java — Design goals & key properties
- JavaBeans
  - ➤ Architecture
  - ➤ Properties
  - ➤ Events
  - ➤ Introspection

H.2 Component Models

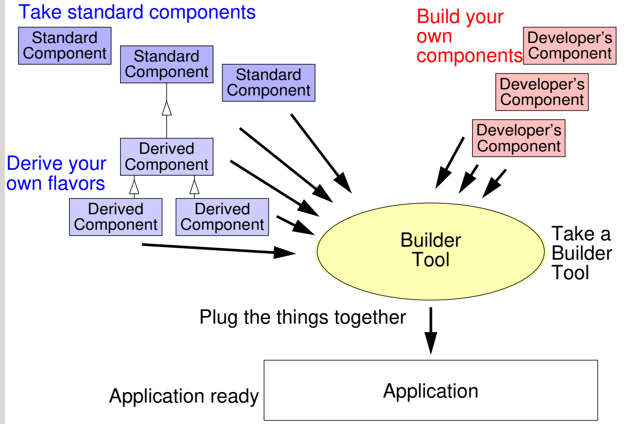
1 What is a Software Component?

- A reusable piece of software that:
  - ➤ has a well-specified public interface
  - ➤ can be used in unpredictable combinations
  - ➤ is a stand-alone, marketable entity
- Software components achieve reuse by following standard conventions
- Software components can be combined (visually) to complex applications
  - ➤ software kit
  - ➤ visual programming with builder tools
- Self-describing
  - ➤ automatic analysis of interface and properties possible

2 Rival Component Architectures

- JavaBeans
- ActiveX
  - ➤ not portable, proprietary
- OpenDOC
  - ➤ pretty much dead
- Proprietary Solutions
  - ➤ GUI builder class libraries

3 Philosophy



H.3 Java — Design Goals?

- Solve today's problems with development and distribution of software
  - ◆ various operating systems (Unix, Windows, MacOS, …)
  - ◆ various hardware architectures
- Java: language and environment for *secure*, *high performance*, and highly *robust* applications on *multiple platforms* in *heterogeneous, distributed networks*

H.4 Java — Key Properties for Components

- Object oriented
- Polymorphism based on class/interface conformance
- Highly dynamic (loading & linking)
- Reflection/introspection mechanisms

H.5 JavaBeans

1 Definition

- JavaBeans is an API specification for creating reusable software components using Java
  - ➤ defines the Java software components
  - ➤ and how they fit together
- A Bean is any Java class that follows the JavaBeans conventions
- The official definition:

  A Java Bean is a reusable software component that can be visually manipulated in builder tools.

2 Beans — Architecture

- Properties
  - ➤ allows customization of the Bean
- Methods
- Events
  - ➤ the wiring that allows Beans to be interconnected
- Adapters
  - ➤ if Beans do not fit together
- Introspection
  - ➤ instead of a repository — just look into the beans

3 Example Beans

- Visual Beans
  - ➤ Custom GUI components
  - ➤ HTML Rendering Bean
  - ➤ OpenGL Canvas
- Non-visual Beans
  - ➤ database connectivity
  - ➤ timer bean
    - – triggers events at certain time intervals
    - – may encapsulate complex date/time logic

4 Properties

- Describe properties of components
- Each property has
  - ➤ a **name** — symbolic description of the property (e. g. Color, Font …)
    `private Color color;` (instance variable of the Bean class)
  - ➤ a **type** — Java class, encapsulating the value(s)
  - ➤ constraints (optional) — e. g. read-only or write-only
- Naming convention
  - ➤ get method for reading

```
public Color getColor(){ return color; }
```

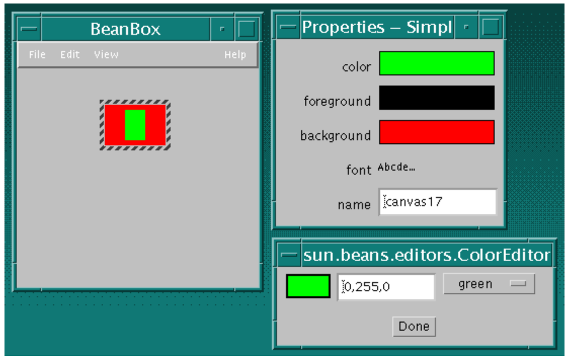
  - ➤ set method for modification

```
public void setColor(Color newColor){
        color = newColor;
        repaint();
}
```

- Examination & modification in property dialogue

4 Properties (2)

- Example: Property Color as property of Bean "SimpleBean"
  - ➤ SimpleBean loaded in BeanBox, clicking on color opens ColorEditor



4 Properties (3)

- Simple Properties
  - ➤ set/get methods
- Bound Properties
  - ➤ fire off a notification when they are changed
- Constrained Properties
  - ➤ proposed changes may be rejected if invalid
- Indexed Properties
  - ➤ used for arrays of properties

5 Events

- Builds on Source-Listener Pattern
  - ➤ EventSource, EventObject, EventListener framework
- PropertyChangeSupport
  - ➤ Allows a Bean to send out notifications whenever a property value changes
- VetoableChangeSupport
  - ➤ Allows Beans to reject property values that are out of range
- Interested Beans register as EventListener

5 Events (2)

- Bean events are not limited to properties.
- You can define your own events using a simple naming convention:
  - ➤ Example

same name

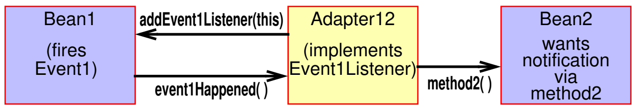
```
public void addTimerListener(TimerListener l)
public void removeTimerListener(TimerListener l)
```

add/remove

TimerListener must be a subclass of java.util.EventListener

6 Adaptors

- Adaptation of events of one bean to methods of another bean



  - ➤ Adapter12 implements appropriate Event-Listener interface
  - ➤ Adapter12 registers for Event1
  - ➤ Event1 happens, Bean1 invokes the method, which was defined in the event interface (`event1Happend()`), at all registered event listeners
  - ➤ `event1Happend()` in Adapter12 invokes `method2()` at Bean2
- Adapter may manipulate event data
- Automatic generation of simple adapters possible

7 Introspection

- Allows automatic analysis of beans
- Java1.1 Reflection API
  - ➤ analysis of Java classes at runtime
  - ➤ members: name & type
  - ➤ methods: name, parameters & return type
- JavaBeans naming conventions
  - ➤ get/set methods -> properties
  - ➤ add/remove methods -> events
  - ➤ other methods -> ordinary methods
- Alternative: Information in BeanInfo class
  - ◆ some sort of interface repository
    - ➤ developer may explicitly specify properties and events