G Java & Component Models & Jini

G.1 Overview

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

G.2 References

Szy98. Clemens Szyperski. *Component Software — Beyond Object-Oriented Programming.* Addison-Wesley, Harlow, England, 1998.

Grif98. Frank Griffel. *Componentware — Konzepte und Techniken eines Softwareparadigmas.* dpunkt-Verlag, Heidelberg, 1998.

JavaBeans. http://www.sun.com/beans/

Jini. http://www.sun.com/jini/specs/

G.3 Component Models

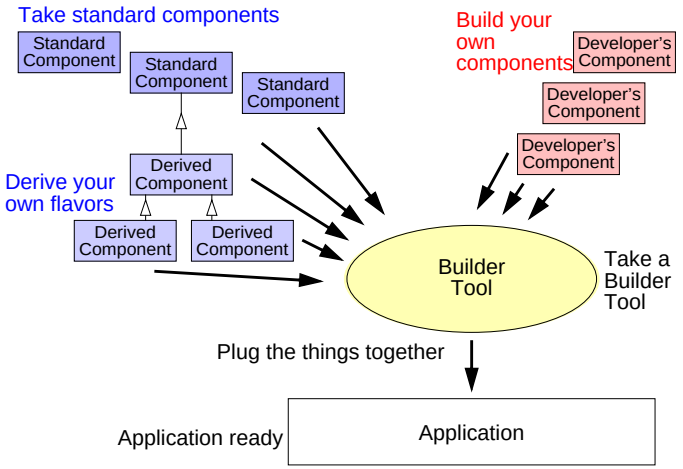
1 What is a Software Component?

- 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



G.4 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*

G.5 Java — Key Properties for Components

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

G.6 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
  - ➤ allow customization of the Bean
- Methods
  Events
  - ➤ the wiring points 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 for accessor methods (methods of the Bean class)
  - ➤ get method for reading

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

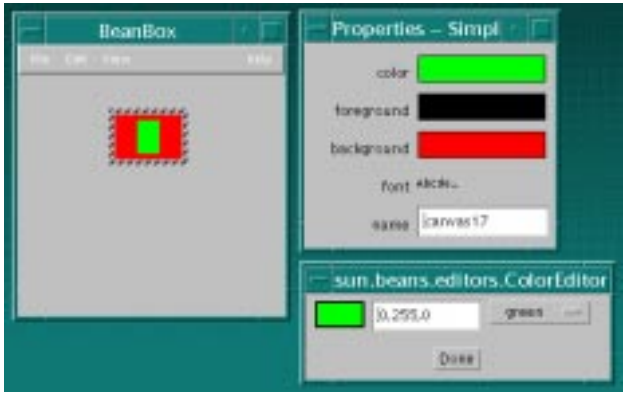
  - ➤ set method for modification

```
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
  - ➤ represent a single value, access by set/get methods
- Indexed properties
  - ➤ represent an array of values, set/get methods take an index parameter
- Bound properties
  - ➤ notify other objects when the value changes (PropertyChange event)
- Constrained properties
  - ➤ proposed changes may be rejected if invalid

5 Events

- Builds on Source-Listener Pattern
  - ➤ source object notifies listener(s) about state changes
  - EventSource
    - ➤ offers methods for listener(s) to register/unregister themselves

same name

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

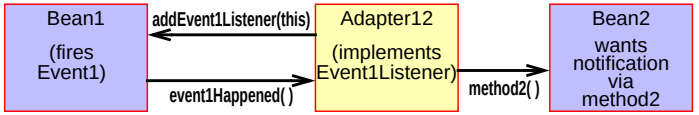
add/remove

TimerListener must implement an EventListener interface

  - EventObject
    - ➤ information about the event
  - EventListener
    - ➤ Class that implements EventListener interface (subtype 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

5 Events (2)

- 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

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

G.7 Jini

1 Overview

- Architecture of a distributed system that supports
  - ➤ federating groups of users and required resources
    - – devices
    - – software components
    - – users
  - ➤ flexible and easy admininstration of dynamically changing networks
- Key concepts
  - ➤ Services
  - ➤ Leasing
  - ➤ Events
  - ➤ Security
  - ➤ Lookup Service (the naming service)
  - ➤ Java RMI
  - ➤ Transactions

2 Jini Services

- Members of a Jini community (*djinn*) federate to share access to services
- Service = an enitity that can be used by a *person*, a *program* or *another service*
  - ➤ computation
  - ➤ storage
  - ➤ communication channel
  - ➤ software filter
  - ➤ hardware device
  - ➤ another user
- Services are registered with the Lookup Service
  - ➤ clients may obtain reference (stub of of an RMI remote ref. or *smart proxy*)
- Services are composed for performance of a particular task

3 Leasing

- Partial failure in distributed systems causes special problems
  - ➤ references become invalid
  - ➤ resources cannot be freed
- Services in Jini are *leased* based on time
  - ➤ Lease = grant of guaranteed access to a service over a time period
  - ➤ may be renewed if needed longer
  - ➤ renewal can be denied by the service provider
  - ➤ if client does not renew (or cannot in case of failures) lease times out
  - ➤ after timeout resources can be freed

4 Events

- JavaBeans supports only local events
  - ➤ object firing the event and all receiving objects must be ion the same VM
- Jini supports distributed events
  - ➤ registration by RMI call and passing a stub for the listener
- Jini events are only delivered as long as the lease is valid

5 Security

- Security model based on *principal information* and *access control lists*