JavaBeans: Events

- Example
- The Good
  - Event Patterns
    - Event Sources, Event Classes, Event Listeners
  - Event Adaptors
    - Filters, Demultiplexors
- The Bad
  - Data Transport
  - Anything else?
- The UVA
  - Exceptions
  - Security
  - Concurrency

JavaBeans: Example Event Source

```
public class Timer extends Thread implements java.io.Serializable {
  private int speed = 10;
  private java.util.Vector listeners = new java.util.Vector();
  public Timer() { start(); }

  public void addTimerListener(TimerListener l) { listeners.add(l); }
  public void removeTimerListener(TimerListener l) { listeners.remove(l);}

  public synchronized int getSpeed() { return speed; }
  public synchronized void setSpeed(int speed) { this.speed = speed; }

  public synchronized void run() {
    while (true) {
      try { wait(1000/speed); } catch (InterruptedException e) { }
      for (int i = 0; i < listeners.size(); i++) {
        ((TimerListener)listeners.get(i)).tick(new TimerEvent(this));
      }
    }
  }
}
```

JavaBeans: Example Event Class and Event Listener

```
public class TimerEvent extends java.util.EventObject {
  public TimerEvent(Timer timer) { super(timer); }
}

public interface TimerListener extends java.util.EventListener {
  public void tick(TimerEvent e);
}

public class Ticker extends java.awt.Canvas implements TimerListener {
  private int k = 0;
  private java.util.Vector ms = new java.util.Vector();
  public Ticker() { }

  public synchronized void addMessage(int i, String m) { ms.add(i, m); }
  public synchronized void setMessage(int i, String m) { ms.set(i, m); }
  public synchronized void removeMessage(int i) { ms.remove(i); }

  . . .
```

JavaBeans: Example Event Listener (Cont'd)

```
  . . .

  public synchronized void tick(TimerEvent e) {
    if (ms.size() > 0) {
      k = (k + 1) % ms.size();
      repaint();
    }
  }

  public synchronized void paint(java.awt.Graphics g) {
    if (k < ms.size()) {
      Dimension d = getSize();
      FontMetrics fm = getFontMetrics(getFont());
      int x = (d.width - fm.stringWidth((String)ms.get(k))) / 2;
      int y = (fm.getHeight() + d.height) / 2;
      g.drawString((String)ms.get(k), x, y);
    }
  }
}
```

JavaBeans: Event Patterns

- Event Source: Multicast or Unicast

```
public class <EventSource> implements java.io.Serializable {
  public void add<EventListener>(<EventListener> listener)
    throws java.util.TooManyListenersException;
  public void remove<EventListener>(<EventListener> listener);
}
```

- Event Class

```
public class <EventClass> extends java.util.EventObject {
  public <EventClass>(<EventSource> src, . . .) { super(src); . . . }
}
```

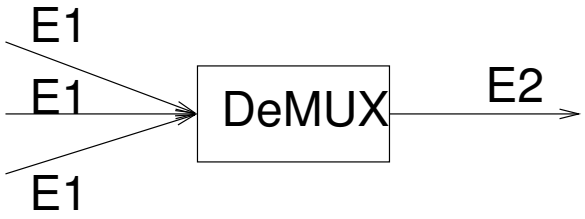
- Event Listener

```
public interface <EventListener> extends java.util.EventListener {
  public void <eventMethod>(<EventClass> e);
}
```

JavaBeans: Event Adaptors

Filters convert event types

Demultiplexors let listeners distinguish between events of the same type from different sources



What happened to **multiplexors**?

JavaBeans: Events Have Limitations

What if you want to send a stream of data? Here is a solution:

```
public class DataEvent extends java.util.EventObject {
  private java.io.InputStream data;

  public void DataEvent(EventSource src, java.io.InputStream data) {
    super(src);
    this.data = data;
  }

  public java.io.InputStream getData() { return data; }
}
```

Does it work?

Do you see any other limitations?

JavaBeans: Events Have Severe Limitations

What happens when procedure calls are used for message passing?

- Exceptions—an exception in a bean can propagate up to unrelated beans that just happened to be on the call stack. `ArrayIndexOutOfBoundsException` is especially troublesome. (Compare to `repaint()`.)
- Security—JVM grants any function the least privileges of all functions on the call stack. (Compare to sockets.)
- Concurrency—JavaBeans neither enforces nor encourages any global synchronization. (Compare to MPI.)

Bottom line: events in JavaBeans are intended for quick notifications. They are not intended for data transfer or 'serious' data processing.