**Constrained Properties**

* **A *constrained* property is a special kind of bound property.**
* **For a constrained property, the bean keeps track of a set of *veto* listeners.**
* **When a constrained property is about to change, the listeners are consulted about the change.**
* **Any one of the listeners has a chance to veto the change, in which case the property remains unchanged.**
* **The java.beans package includes a VetoableChangeSupport class to implement constrained properties.**

**import java.beans.\*;**

**public class AccountBean {**

**private int amount = 1000;**

**private PropertyChangeSupport mPcs =**

**new PropertyChangeSupport(this);**

**private VetoableChangeSupport mVcs =**

**new VetoableChangeSupport(this);**

**public int getAmount() {**

**return amount;**

**}**

**public void setAmount(int amt) throws**

**PropertyVetoException {**

**int oldamount = amount;**

**mVcs.fireVetoableChange("amount",**

**new Integer(oldamount), new Integer(amt));**

**amount = amt;**

**mPcs.firePropertyChange("amount",**

**new Integer(oldamount), new Integer(amt));**

**}**

**public void**

**addPropertyChangeListener(PropertyChangeListener**

**listener) {**

**mPcs.addPropertyChangeListener(listener);**

**}**

**public void**

**removePropertyChangeListener(PropertyChangeListener**

**listener) {**

**mPcs.removePropertyChangeListener(listener);**

**}**

**public void**

**addVetoableChangeListener(VetoableChangeListener**

**listener) {**

**mVcs.addVetoableChangeListener(listener);**

**}**

**public void**

**removeVetoableChangeListener(VetoableChangeListener**

**listener) {**

**mVcs.removeVetoableChangeListener(listener);**

**}**

**}**

**class Test implements VetoableChangeListener {**

**public static void main(String[] args) {  
        AccountBean bean = new AccountBean();**

**bean.addVetoableChangeListener(this);  
        bean.setAmount(2000);  
        bean.setAmount(1500);  
          
        bean.setAmount(3000);  
    }  
    public void vetoableChange(PropertyChangeEvent evt)  throws PropertyVetoException {  
    String propertyName = evt.getPropertyName();  
    if (propertyName.equalsIgnoreCase("amount")) {**

**int amount = ((Integer) evt.getNewValue()).intValue();  
 if (amount > 2000) {  
 throw new PropertyVetoException("Amount must be below**

**or equal to 2000", evt);                  
 }  
 System.out.println("Amount applied = " + amount);  
 }  
 }  
}**