

```

package boundProp;

import java.beans.*;

public class Client {
    Boiler itsBoiler = new Boiler();

    public Client() {
        try {
            jbInit();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Client client = new Client();
        for(int i = 0; i < 10; ++i) {
            client.itsBoiler.setTemperature((double)i);
        }
    }

    private void jbInit() throws Exception {
        itsBoiler.addPropertyChangeListener(new java.beans.PropertyChangeListener() {

            public void propertyChange(PropertyChangeEvent e) {
                itsBoiler_propertyChange(e);
            }
        });
    }

    void itsBoiler_propertyChange(PropertyChangeEvent e) {
        System.out.println("Property "+e.getPropertyName()+
            " changed from: "+e.getOldValue() +
            " to: "+ e.getNewValue());
    }
}

/* --- output ---
Property temperature changed from: 0.0 to: 1.0
Property temperature changed from: 1.0 to: 2.0
Property temperature changed from: 2.0 to: 3.0
Property temperature changed from: 3.0 to: 4.0
Property temperature changed from: 4.0 to: 5.0
Property temperature changed from: 5.0 to: 6.0
Property temperature changed from: 6.0 to: 7.0
Property temperature changed from: 7.0 to: 8.0
Property temperature changed from: 8.0 to: 9.0
--- end output --- */

```

```

//Title:      Bound Property Test
//Version:
//Copyright:  Copyright (c) 2000
//Author:     Furrukh Khan
//Company:    OSU
//Description:

package boundProp;

import java.io.*;
import java.beans.*;

public class Boiler implements Serializable {

    public Boiler() {
    }

    void readObject(ObjectInputStream ois) throws ClassNotFoundException, IOException {
        ois.defaultReadObject();
    }

    void writeObject(ObjectOutputStream oos) throws IOException {
        oos.defaultWriteObject();
    }

    public void setTemperature(double newTemperature) {
        double oldTemperature = temperature;
        temperature = newTemperature;
        propertyChangeListeners.firePropertyChange("temperature", new Double(oldTemperature), new Double(newTemperature));
    }

    public double getTemperature() {
        return temperature;
    }

    public synchronized void removePropertyChangeListener(PropertyChangeListener l) {
        propertyChangeListeners.removePropertyChangeListener(l);
    }

    public synchronized void addPropertyChangeListener(PropertyChangeListener l) {
        propertyChangeListeners.addPropertyChangeListener(l);
    }

    private double temperature;
    private transient PropertyChangeSupport propertyChangeListeners = new PropertyChangeSupport(this);
}

```