Exercise: Undoable Actions and 'Swing'

The Exercise

This very simple exercise is designed to give you a 'taste' of the Undo/Redo facility provided in the javax.swing.undo package that is part of Java 1.2 and greater.

Building the Application

This is a screen shot of the application that you will build in this exercise:

The code given below should start you off. You will need to examine those comments given in ***bold italics*** and add the necessary code. Useful resources for this exercise include the standard Javadoc pages on the javax.swing.undo package and http://www.javaworld.com/javaworld/jw-06-1998/jw-06-undoredo.html.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.undo.*;

public class UndoableExample extends JFrame
  {
  private long model = 0;
  private JTextField view;
  private UndoManager undoManager = new UndoManager ();
  private JButton undo = new JButton (),
                  redo = new JButton (),
                  controller = new JButton ("Increment");

  private void updateButtons ()
    {
    // enable the undo & redo buttons according to the Undo Manager
    // set the Text of the buttons according to the Undo Manager
    }

  public UndoableExample ()
    {
    setTitle ("UndoableExample");

    Container c = getContentPane ();
    c.setLayout (new FlowLayout ());

    view = new JTextField (model + "", 10);
    c.add (view);
    controller.addActionListener
      (
      new ActionListener ()
        {
        public void actionPerformed (ActionEvent e)
          {
          Incrementer i = new Incrementer ();
          undoManager.addEdit (i);
          updateButtons ();
          }
        }
      );
    c.add (controller);
    ActionListener undoRedoListener = new ActionListener ()
      {
      public void actionPerformed (ActionEvent e)
        {
        if (((JButton) e.getSource ()) == undo)
          // tell the Undo Manager instance to undo the operation at the top
          // of it's action stack
        else
          // tell the Undo Manager instance to redo
        updateButtons ();
```

```
        }
      };

    undo.setEnabled (false);
    undo.addActionListener (undoRedoListener);
    c.add (undo);
    redo.setEnabled (false);
    redo.addActionListener (undoRedoListener);
    c.add (redo);
    updateButtons ();
    }

  private class Incrementer extends AbstractUndoableEdit
    {
    long value;
    public Incrementer ()
      {
      // save the current state of 'model'
      // perform the increment and then update view appropriately
      }
    public void undo () throws CannotUndoException
      {
      // restore 'model' to the value saved and then update view appropriately
      }
    public void redo () throws CannotRedoException
      {
      // set 'model' to value + 1 and then update view appropriately
      }
    public boolean canUndo () { return (true); }
    public boolean canRedo () { return (true); }
    public String getPresentationName () { return ("Increment"); }
    }

  public static void main (String [] args)
    {
    try
      {
      UIManager.setLookAndFeel (UIManager.getSystemLookAndFeelClassName ());
      UndoableExample d = new UndoableExample ();
      d.addWindowListener
        (
        new WindowAdapter ()
          {
          public void windowClosing (WindowEvent e) { System.exit (0); }
          }
        );

      d.setSize (new Dimension (500, 60));
      d.setVisible (true);
      }
    catch (Exception e)
      {
      e.printStackTrace (System.err);
      }
    }
  }
```

Further Work

Augment this example with an undoable Decrement operation.