```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Vista;

/*
 * Copyright (c) 1995, 2008, Oracle and/or its affiliates. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 *   - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *
 *   - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *
 *   - Neither the name of Oracle or the names of its
 *     contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
 * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */


/*
 * TableRenderDemo.java requires no other files.
 */

import javax.swing.DefaultCellEditor;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.TableCellRenderer;
import javax.swing.table.TableColumn;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridLayout;

/**
 * TableRenderDemo is just like TableDemo, except that it
 * explicitly initializes column sizes and it uses a combo box
 * as an editor for the Sport column.
 */
public class TableRenderDemo extends JPanel {
    private boolean DEBUG = false;

    public TableRenderDemo() {
        super(new GridLayout(1,0));

        JTable table = new JTable(new MyTableModel());
        table.setPreferredScrollableViewportSize(new Dimension(500, 70));
        table.setFillsViewportHeight(true);

        //Create the scroll pane and add the table to it.
        JScrollPane scrollPane = new JScrollPane(table);

        //Set up column sizes.
        initColumnSizes(table);

        //Fiddle with the Sport column's cell editors/renderers.
        setUpSportColumn(table, table.getColumnModel().getColumn(2));

        //Add the scroll pane to this panel.
        add(scrollPane);
    }

    /*
     * This method picks good column sizes.
     * If all column heads are wider than the column's cells'
     * contents, then you can just use column.sizeWidthToFit().
     */
    private void initColumnSizes(JTable table) {
        MyTableModel model = (MyTableModel)table.getModel();
        TableColumn column = null;
        Component comp = null;
        int headerWidth = 0;
        int cellWidth = 0;
        Object[] longValues = model.longValues;
        TableCellRenderer headerRenderer =
            table.getTableHeader().getDefaultRenderer();

        for (int i = 0; i < 5; i++) {
            column = table.getColumnModel().getColumn(i);

            comp = headerRenderer.getTableCellRendererComponent(
                                 null, column.getHeaderValue(),
                                 false, false, 0, 0);
            headerWidth = comp.getPreferredSize().width;

            comp = table.getDefaultRenderer(model.getColumnClass(i)).
                             getTableCellRendererComponent(
                                 table, longValues[i],
                                 false, false, 0, i);
            cellWidth = comp.getPreferredSize().width;

            if (DEBUG) {
                System.out.println("Initializing width of column "
                                   + i + ". "
                                   + "headerWidth = " + headerWidth
                                   + "; cellWidth = " + cellWidth);
            }

            column.setPreferredWidth(Math.max(headerWidth, cellWidth));
        }
    }

    public void setUpSportColumn(JTable table,
                                 TableColumn sportColumn) {
        //Set up the editor for the sport cells.
        JComboBox comboBox = new JComboBox();
        comboBox.addItem("Snowboarding");
        comboBox.addItem("Rowing");
        comboBox.addItem("Knitting");
        comboBox.addItem("Speed reading");
        comboBox.addItem("Pool");
        comboBox.addItem("None of the above");
        sportColumn.setCellEditor(new DefaultCellEditor(comboBox));

        //Set up tool tips for the sport cells.
        DefaultTableCellRenderer renderer =
                new DefaultTableCellRenderer();
        renderer.setToolTipText("Click for combo box");
        sportColumn.setCellRenderer(renderer);
    }

    class MyTableModel extends AbstractTableModel {
```

```java
        private String[] columnNames = {"First Name",
                                        "Last Name",
                                        "Sport",
                                        "# of Years",
                                        "Vegetarian"};
        private Object[][] data = {
        {"Kathy", "Smith",
         "Snowboarding", new Integer(5), new Boolean(false)},
        {"John", "Doe",
         "Rowing", new Integer(3), new Boolean(true)},
        {"Sue", "Black",
         "Knitting", new Integer(2), new Boolean(false)},
        {"Jane", "White",
         "Speed reading", new Integer(20), new Boolean(true)},
        {"Joe", "Brown",
         "Pool", new Integer(10), new Boolean(false)}
        };

        public final Object[] longValues = {"Jane", "Kathy",
                                            "None of the above",
                                            new Integer(20), Boolean.TRUE};

        public int getColumnCount() {
            return columnNames.length;
        }

        public int getRowCount() {
            return data.length;
        }

        public String getColumnName(int col) {
            return columnNames[col];
        }

        public Object getValueAt(int row, int col) {
            return data[row][col];
        }

        /*
         * JTable uses this method to determine the default renderer/
         * editor for each cell.  If we didn't implement this method,
         * then the last column would contain text ("true"/"false"),
         * rather than a check box.
         */
        public Class getColumnClass(int c) {
            return getValueAt(0, c).getClass();
        }

        /*
         * Don't need to implement this method unless your table's
         * editable.
         */
        public boolean isCellEditable(int row, int col) {
            //Note that the data/cell address is constant,
            //no matter where the cell appears onscreen.
            if (col < 2) {
                return false;
            } else {
                return true;
            }
        }

        /*
         * Don't need to implement this method unless your table's
         * data can change.
         */
        public void setValueAt(Object value, int row, int col) {
            if (DEBUG) {
                System.out.println("Setting value at " + row + "," + col
                                   + " to " + value
                                   + " (an instance of "
                                   + value.getClass() + ")");
            }

            data[row][col] = value;
            fireTableCellUpdated(row, col);

            if (DEBUG) {
                System.out.println("New value of data:");
                printDebugData();
            }
        }

        private void printDebugData() {
            int numRows = getRowCount();
            int numCols = getColumnCount();

            for (int i=0; i < numRows; i++) {
                System.out.print("    row " + i + ":");
                for (int j=0; j < numCols; j++) {
                    System.out.print("  " + data[i][j]);
                }
                System.out.println();
            }
            System.out.println("--------------------------");
        }
    }

    /**
     * Create the GUI and show it.  For thread safety,
     * this method should be invoked from the
     * event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Create and set up the window.
        JFrame frame = new JFrame("TableRenderDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Create and set up the content pane.
        TableRenderDemo newContentPane = new TableRenderDemo();
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);

        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```