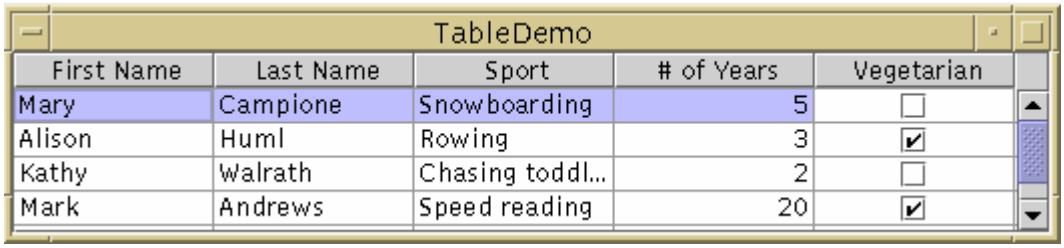
PENGGUNAAN JTABLE

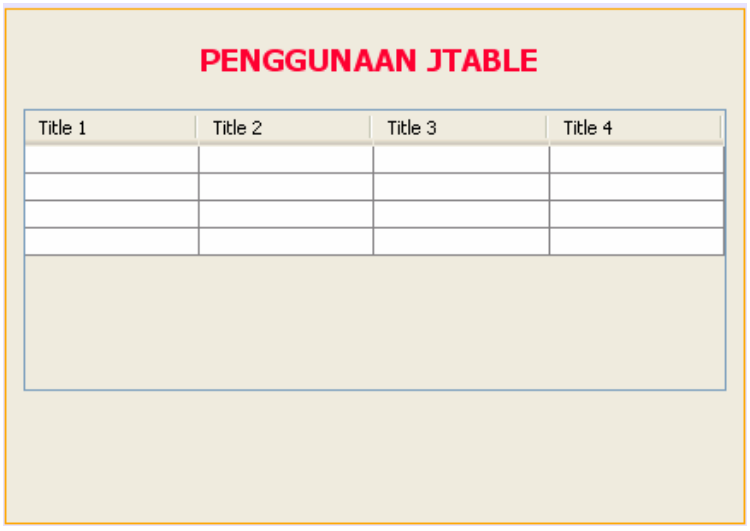
Dengan class JTable kita dapat menampilkan data, juga membolehkan user untuk mengedit data. Gambar 1 merupakan contoh penggunaan JTable. Setiap cell menampilkan sebuah item data. Setiap header kolom merupakan kolom dari tabel. Setiap kolom mempunyai tipe data yang sama. Header tabel menampilkan header kolom.



Gambar 1

Aplikasi 1 : TestTable1.java

Buatlah sebuah class dengan nama TestTable1. Desain form seperti gambar 2. Pada aplikasi terdapat label dan JTable beri nama dengan jTable. Buatlah isi dari TestTable1.java seperti program di bawah ini. Pada TestTable1 menggunakan class MyTableModel yang merupakan subclass dari class AbstractTableModel. Output program ditunjukkan pada gambar 3.



Gambar 2

```
public class TestTable1 extends javax.swing.JFrame {

    /** Creates new form TestTable1 */
    public TestTable1() {
        super("TestTable1");
```

```
        initComponents();
        jTable.setModel(new MyTableModel());
        jTable.setPreferredScrollableViewportSize(new Dimension(500,
70));

        TableColumn column = null;
        for (int i = 0; i < 5; i++) {
            column = jTable.getColumnModel().getColumn(i);
            if (i == 2) {
                column.setPreferredWidth(100); //sport column is bigger
            } else {
                column.setPreferredWidth(50);
            }
        }
    }

  public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new TestTable1().setVisible(true);
            }
        });
    }
   // Variables declaration - do not modify
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable;
}

public class MyTableModel extends AbstractTableModel {
    private String[] columnNames = {"First Name",
                                    "Last Name",
                                    "Sport",
                                    "# of Years",
                                    "Vegetarian"};
        private Object[][] data = {
            {"Mary", "Campione",
             "Snowboarding", new Integer(5), new Boolean(false)},
            {"Alison", "Huml",
             "Rowing", new Integer(3), new Boolean(true)},
            {"Kathy", "Walrath",
             "Knitting", new Integer(2), new Boolean(false)},
            {"Sharon", "Zakhour",
             "Speed reading", new Integer(20), new Boolean(true)},
            {"Philip", "Milne",
             "Pool", new Integer(10), new Boolean(false)}
        };

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

Output Program :

Gambar 3

Membuat Table Model

Setiap table mendapatkan data dari sebuah object yang mengimplementasikan interface **TableModel. Cara mengimplementasikan table model sangat sederhana, yaitu dengan membuat subclass dari AbstractTableModel (class MyTableModel).** Pada fungsi isCellEditable, kolom 0 dan 1 tidak bisa diedit sedangkan kolom lainnya bisa diedit.

Gambar 4

```
public class MyTableModel extends AbstractTableModel {
    private String[] columnNames = {"First Name",
                                    "Last Name",
                                    "Sport",
                                    "# of Years",
                                    "Vegetarian"};
        private Object[][] data = {
            {"Mary", "Campione",
             "Snowboarding", new Integer(5), new Boolean(false)},
            {"Alison", "Huml",
             "Rowing", new Integer(3), new Boolean(true)},
            {"Kathy", "Walrath",
             "Knitting", new Integer(2), new Boolean(false)},
            {"Sharon", "Zakhour",
             "Speed reading", new Integer(20), new Boolean(true)},
            {"Philip", "Milne",
```

```
             "Pool", new Integer(10), new Boolean(false)}
        };

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

Mengubah Lebar Kolom

Secara default lebar kolom pada table adalah sama, dan kolom-kolom secara otomatis akan mengisi keseluruhan lebar table. Jika table kita kecilkan atau diperbesar (pada saat user ingin mengubah window), semua lebar kolom juga akan berubah. User juga bisa mengubah lebar kolom dengan melakukan drag pada kolom.

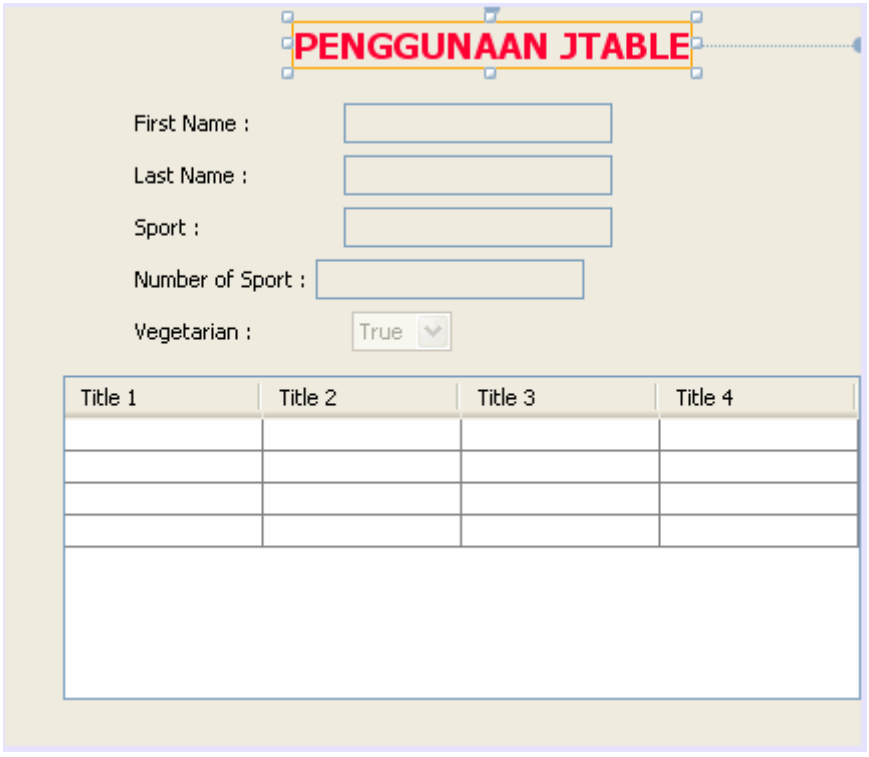
Kita juga dapat mengatur lebar kolom, gunakan fungsi setPreferredWidth() pada setiap kolom table. Pada source code TestTable1 terdapat program seperti di bawah ini, dimana kolom 3 mempunyai lebar kolom yang paling besar dibandingkan kolom lainnya.

```
TableColumn column = null;
for (int i = 0; i < 5; i++) {
    column = jTable.getColumnModel().getColumn(i);
    if (i == 2) {
        column.setPreferredWidth(100); //sport column is bigger
    } else {
        column.setPreferredWidth(50);
    }
}
```

Pada program diatas, setiap kolom pada table dinyatakan dengan object TableColumn. Pada TableColumn menyediakan methods getXXX dan setXXX untuk minumum, maksimum dan yang diinginkan dari lebar kolom.

Aplikasi 2 : TestTable2.java

Pada aplikasi 2 ini jika baris pada table kita pilih (baris yang kita pilih tunggal) maka data akan ditampilkan pada textfield. Desainlah aplikasi seperti Gambar 5 dan berilah nama text field dan combo box seperti table 1. Output program ditunjukkan pada gambar 6.



Gambar 5

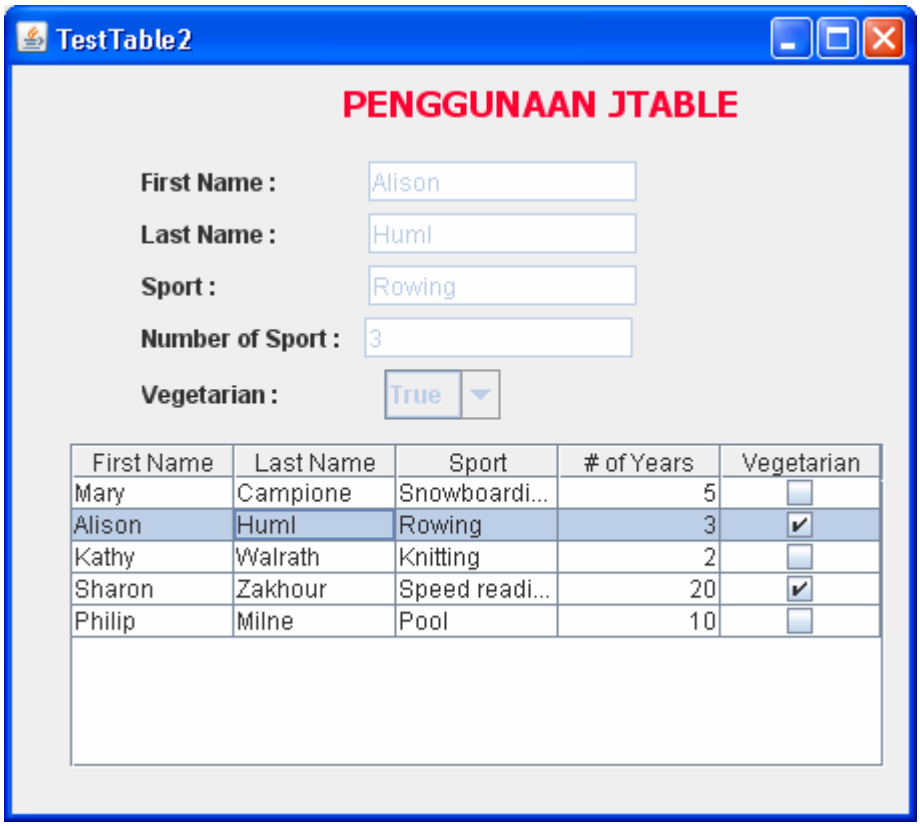
Tabel 1 Nama Variabel pada Aplikasi 2

	Nama TextField
First Name	tfFirstName
Last Name	tfLastName
Sport	tfSport
Number of Sport	tfNum
	Nama ComboBox
Vegetarian	cbVegetarian

```
public class TestTable2 extends javax.swing.JFrame {

    /** Creates new form TestTable2 */
    public TestTable2() {
        super("TestTable2");
        initComponents();
        jTable.setModel(new MyTableModel());
        jTable.setPreferredScrollableViewportSize(new Dimension(500, 70));
        pilih();
    }

    void pilih (){
        jTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        ListSelectionModel rowSM = jTable.getSelectionModel();
        rowSM.addListSelectionListener(new MyList());
    }

    class MyList implements ListSelectionListener{

        public void valueChanged(ListSelectionEvent e) {

            if (e.getValueIsAdjusting()) return;

            ListSelectionModel lsm = (ListSelectionModel) e.getSource();
            int selectedRow = lsm.getMinSelectionIndex();

            TableModel tm = jTable.getModel() ;
            tfFirstName.setText(tm.getValueAt(selectedRow,0).toString()) ;
            tfLastName.setText(tm.getValueAt(selectedRow,1).toString()) ;
            tfSport.setText(tm.getValueAt(selectedRow,2).toString()) ;
            tfNum.setText(tm.getValueAt(selectedRow,3).toString()) ;
            if (tm.getValueAt(selectedRow,4).toString().equals("true"))
                cbVegetarian.setSelectedIndex(0);
            else
                cbVegetarian.setSelectedIndex(1);
        }
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new TestTable2().setVisible(true);
            }
        });
    }
```

```
    // Variables declaration - do not modify
    private javax.swing.JComboBox cbVegetarian;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable;
    private javax.swing.JTextField tfFirstName;
    private javax.swing.JTextField tfLastName;
    private javax.swing.JTextField tfNum;
    private javax.swing.JTextField tfSport;
    // End of variables declaration

}
```

Output Program :



Gambar 6

Mendeteksi Pemilihan User

Gunakan fungsi setSelectionMode untuk menentukan mode pemilihan (Single Selection, Multiple Interval Selection dan Single Interval Selection). Untuk menentukan single selection adalah sbb :

```
void pilih (){
    //untuk menentukan single selection
    jTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    ListSelectionModel rowSM = jTable.getSelectionModel();
    rowSM.addListSelectionListener(new MyList());
}

class MyList implements ListSelectionListener{

    public void valueChanged(ListSelectionEvent e) {
        //untuk menghilangkan informasi ekstra
   if (e.getValueIsAdjusting()) return;
        ListSelectionModel lsm = (ListSelectionModel) e.getSource();
        int selectedRow = lsm.getMinSelectionIndex();

        TableModel tm = jTable.getModel() ;
        tfFirstName.setText(tm.getValueAt(selectedRow,0).toString()) ;
        tfLastName.setText(tm.getValueAt(selectedRow,1).toString()) ;
        tfSport.setText(tm.getValueAt(selectedRow,2).toString()) ;
        tfNum.setText(tm.getValueAt(selectedRow,3).toString()) ;
        if (tm.getValueAt(selectedRow,4).toString().equals("true"))
            cbVegetarian.setSelectedIndex(0);
        else
            cbVegetarian.setSelectedIndex(1);
    }
}
```

Aplikasi 3 : TestTable3.java

Aplikasi sama dengan aplikasi 2, tapi jika kita klik cell pada kolom Sport akan keluar combo box. Output program ditunjukkan pada gambar 7.

Gambar 7

```
public TestTable3() {
        super("TestTable3");
        initComponents();
        jTable.setModel(new MyTableModel());
        jTable.setPreferredScrollableViewportSize(new Dimension(500, 70));
        setUpSportColumn(jTable, jTable.getColumnModel().getColumn(2));


        pilih();

    }

    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));


    }
```

Menggunakan Combo Box sebagai Editor.

Program di bawah ini untuk mensetting combo box sebagai editor sederhana.Isilah combobox dengan string. Kemudian tentukan kolom yang akan diberikan editor (untuk aplikasi 3 kolom2, tabel kolom diberi nama sportColumn). Untuk memberikan editor pada setiap baris pada kolom tersebut gunakan fungsi setCellEditor (TableCellEditor).

```
TableColumn sportColumn = table.getColumnModel().getColumn(2);
...
JComboBox comboBox = new JComboBox();
comboBox.addItem("Snowboarding");
comboBox.addItem("Rowing");
comboBox.addItem("Chasing toddlers");
comboBox.addItem("Speed reading");
comboBox.addItem("Teaching high school");
comboBox.addItem("None");
sportColumn.setCellEditor(new DefaultCellEditor(comboBox));
```