

# 1. 概述

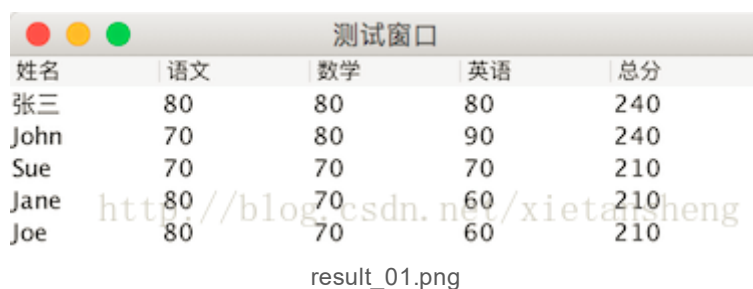
官方JavaDocsApi: [javax.swing.JTable](#)

**JTable**, 表格。JTable 是用来显示和编辑常规二维单元表。

## 2. 创建简单的表格

```
1 package com.xiets.swing;
2 import javax.swing.*;
3 import java.awt.*;
4 public class Main{
5     public static void main(String[] args){
6         JFrame jf =new JFrame("测试窗口");
7         jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
8         // 创建内容面板，使用边界布局
9         JPanel panel =new JPanel(new BorderLayout());
10        // 表头（列名）
11        Object[] columnNames ={"姓名","语文","数学","英语","总分"};
12        // 表格所有行数据
13        Object[][] rowData ={
14            {"张三",80,80,80,240},
15            {"John",70,80,90,240},
16            {"Sue",70,70,70,210},
17            {"Jane",80,70,60,210},
18            {"Joe",80,70,60,210}
19        };
20        // 创建一个表格，指定 所有行数据 和 表头
21        JTable table =new JTable(rowData, columnNames);
22        // 把 表头 添加到容器顶部（使用普通的中间容器添加表格时，表头 和 内容 需要分开添加）
23        panel.add(table.getTableHeader(), BorderLayout.NORTH);
24        // 把 表格内容 添加到容器中心
25        panel.add(table, BorderLayout.CENTER);
26        jf.setContentPane(panel);
27        jf.pack();
28        jf.setLocationRelativeTo(null);
29        jf.setVisible(true);
30    }
31 }
32
```

结果展示:



姓名	语文	数学	英语	总分
张三	80	80	80	240
John	70	80	90	240
Sue	70	70	70	210
Jane	80	70	60	210
Joe	80	70	60	210

result\_01.png

表格组件和其他普通组件一样，需要添加到中间容器中才能显示，**添加表格到容器中有两种方式**:

1. 添加到普通的中间容器中（如上面代码实例所示的添加到JPanel），此时添加的JTable只是表格的行内容，表头(JTable.getTableHeader())需要额外单独添加。此添加方式适合表格行数确定，数据量较小，能一次性显示完的表格；
2. 添加到JScrollPane滚动容器中，此添加方式不需要额外添加表头，JTable添加到JScrollPane中后，表头自动添加到滚动容器的顶部，并支持行内容的滚动（滚动行内容时，表头会始终在顶部显示）。

### 3. JTable 常用的操作方法

JTable常用构造方法:

```
1 // 创建空表格，后续再添加相应数据
2 JTable()
3 // 创建指定行列数的空表格，表头名称默认使用大写字母（A, B, C ...）依次表示
4 JTable(int numRows,int numColumns)
5 // 创建表格，指定 表格行数据 和 表头名称
6 JTable(Object[][] rowData, Object[] columnNames)
7 // 使用表格模型创建表格
8 JTable(TableModel dm)
```

JTable **字体** 和 **网格** 颜色设置:

```
1 // 设置内容字体
2 void setFont(Font font)
3 // 设置字体颜色
4 void setForeground(Color fg)
5 // 设置被选中的行前景（被选中时字体的颜色）
6 void setSelectionForeground(Color selectionForeground)
7 // 设置被选中的行背景
```

```

8 void setSelectionBackground(Color selectionBackground)
9 // 设置网格颜色
10 void setGridColor(Color gridColor)
11 // 设置是否显示网格
12 void setShowGrid(boolean showGrid)
13 // 水平方向网格线是否显示
14 void setShowHorizontalLines(boolean showHorizontalLines)
15 // 竖直方向网格线是否显示
16 void setShowVerticalLines(boolean showVerticalLines)

```

## JTable 表头 设置:

```

1 // 获取表头
2 JTableHeader jTableHeader = jTable.getHeader();
3 // 设置表头名称字体样式
4 jTableHeader.setFont(Font font);
5 // 设置表头名称字体颜色
6 jTableHeader.setForeground(Color fg);
7 // 设置用户是否可以通过在头间拖动来调整各列的大小。
8 jTableHeader.setResizingAllowed(boolean resizingAllowed);
9 // 设置用户是否可以拖动列头，以重新排序各列。
10 jTableHeader.setReorderingAllowed(boolean reorderingAllowed);

```

## JTable 行列 相关设置:

```

1 // 设置所有行的行高
2 void setRowHeight(int rowHeight)
3 // 设置指定行的行高void setRowHeight(int row,int rowHeight)
4 /**
5  * 设置当手动改变某列列宽时，其他列的列宽自动调整模式，可选值：
6  * JTable.AUTO_RESIZE_ALL_COLUMNS 在所有的调整大小操作中，按比例调整所有的
  列。
7  * JTable.AUTO_RESIZE_LAST_COLUMN 在所有的调整大小操作中，只对最后一列进行调整。
8  * JTable.AUTO_RESIZE_NEXT_COLUMN 在 UI 中调整了一个列时，对其下一列进行相反
  方向的调整。
9  * JTable.AUTO_RESIZE_OFF 不自动调整列的宽度；使用滚动条。
10  * JTable.AUTO_RESIZE_SUBSEQUENT_COLUMNS 在 UI 调整中，更改后续列以保持总宽
  度不变；此为默认行为。
11  */
12 void setAutoResizeMode(int mode)
13 /* * 调整列宽 */

```

```

14 // 先获取到某列
15 TableColumn tableColumn = jTable.getColumnModel().getColumn(int columnIndex);
16 // 设置列的宽度、首选宽度、最小宽度、最大宽度
17 tableColumn.setWidth(int width);
18 tableColumn.setPreferredWidth(int preferredWidth);
19 tableColumn.setMinWidth(int minWidth);
20 tableColumn.setMaxWidth(int maxWidth);
21 // 调整该列的列宽，以适合其标题单元格的宽度。
22 tableColumn.setWidthToFit();
23 // 是否允许手动改变该列的列宽
24 tableColumn.setResizable(boolean isResizable);
25 // 设置该列的表头名称
26 tableColumn.setHeaderValue(Object headerValue);

```

## JTable 数据 相关操作:

```

1 /* * 表格数据的简单设置和获取 */
2 // 设置表格中指定单元格的数据
3 jTable.getModel().setValueAt(Object aValue,int rowIndex,int columnIndex);
4 // 获取表格中指定单元格的数据
5 Object value = jTable.getModel().getValueAt(int rowIndex,int
columnIndex);

```

# 4. 创建带滚动条的表格

## 创建带滚动条的表格基本步骤:

```

1 // 创建表格
2 JTable table =newJTable(...);
3 /* 设置表格相关数据 */
4 // 设置滚动面板视口大小（超过该大小的行数据，需要拖动滚动条才能看到）
5 table.setPreferredScrollableViewportSize(newDimension(int width,int height));
6 // 创建滚动面板，把 表格 放到 滚动面板 中（表头将自动添加到滚动面板顶部）
7 JScrollPane scrollPane =newJScrollPane(table);
8 /* 再把滚动面板 scrollPane 添加到其他容器中显示 */

```

## 完整实例代码:

```

1 package com.xiets.swing;
2 import javax.swing.*.*;
3 import java.awt.*.*;

```

```
4 public class Main{
5     public static void main(String[] args){
6         JFrame jf =new JFrame("测试窗口");
7         jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
8         // 创建内容面板
9         JPanel panel =new JPanel();
10        // 表头（列名）
11        String[] columnNames ={"序号","姓名","语文","数学","英语","总分"};
12        // 表格所有行数据
13        Object[][] rowData ={
14            {1,"张三",80,80,80,240},
15            {2,"John",70,80,90,240},
16            {3,"Sue",70,70,70,210},
17            {4,"Jane",80,70,60,210},
18            {5,"Joe_05",80,70,60,210},
19            {6,"Joe_06",80,70,60,210},
20            {7,"Joe_07",80,70,60,210},
21            {8,"Joe_08",80,70,60,210},
22            {9,"Joe_09",80,70,60,210},
23            {10,"Joe_10",80,70,60,210},
24            {11,"Joe_11",80,70,60,210},
25            {12,"Joe_12",80,70,60,210},
26            {13,"Joe_13",80,70,60,210},
27            {14,"Joe_14",80,70,60,210},
28            {15,"Joe_15",80,70,60,210},
29            {16,"Joe_16",80,70,60,210},
30            {17,"Joe_17",80,70,60,210},
31            {18,"Joe_18",80,70,60,210},
32            {19,"Joe_19",80,70,60,210},
33            {20,"Joe_20",80,70,60,210}
34        };
35        // 创建一个表格，指定 表头 和 所有行数据
36        JTable table =new JTable(rowData, columnNames);
37        // 设置表格内容颜色
38        table.setForeground(Color.BLACK);
39        // 字体颜色
40        table.setFont(newFont(null, Font.PLAIN,14));
41        // 字体样式
42        table.setSelectionForeground(Color.DARK_GRAY);
43        // 选中后字体颜色
```

```
44 table.setSelectionBackground(Color.LIGHT_GRAY);
45 // 选中后字体背景
46 table.setGridColor(Color.GRAY);
47 // 网格颜色
48 // 设置表头
49 table.getTableHeader().setFont(newFont(null, Font.BOLD, 14));
50 // 设置表头名称字体样式
51 table.getTableHeader().setForeground(Color.RED);
52 // 设置表头名称字体颜色
53 table.getTableHeader().setResizingAllowed(false);
54 // 设置不允许手动改变列宽
55 table.getTableHeader().setReorderingAllowed(false);
56 // 设置不允许拖动重新排序各列
57 // 设置行高
58 table.setRowHeight(30);
59 // 第一列列宽设置为40
60 table.getColumnModel().getColumn(0).setPreferredWidth(40);
61 // 设置滚动面板视口大小（超过该大小的行数据，需要拖动滚动条才能看到）
62 table.setPreferredScrollableViewportSize(newDimension(400, 300));
63 // 把 表格 放到 滚动面板 中（表头将自动添加到滚动面板顶部）
64 JScrollPane scrollPane =new JScrollPane(table);
65 // 添加 滚动面板 到 内容面板
66 panel.add(scrollPane);
67 // 设置 内容面板 到 窗口
68 jf.setContentPane(panel);
69 jf.pack();
70 jf.setLocationRelativeTo(null);
71 jf.setVisible(true);
72 }
73 }
74
```

结果展示:



序号	姓名	语文	数学	英语	总分
1	张三	80	80	80	240
2	John	70	80	90	240
3	Sue	70	70	70	210
4	Jane	80	70	60	210
5	Joe_05	80	70	60	210
6	Joe_06	80	70	60	210
7	Joe_07	80	70	60	210
8	Joe_08	80	70	60	210
9	Joe_09	80	70	60	210
10	Joe_10	80	70	60	210

result\_02.png

## 5. 表格模型 (TableModel)

TableModel 接口指定了 JTable 用于询问表格式数据模型的方法。

TableModel 封装了表格中的各种数据，为表格显示提供数据。上面案例中直接使用行数据和表头创建表格，实际上 JTable 内部自动将传入的行数据和表头封装成了 TableModel。

只要数据模型实现了 TableModel 接口，就可以通过以下两行代码设置 JTable 显示该模型：

```
1 TableModel myData = new MyTableModel();
2 JTable table = new JTable(myData);
```

TableModel 接口中的方法：

```
1 package javax.swing.table;
2 import javax.swing.*;
3 import javax.swing.event.*;
4 public interface TableModel{
5     /** 返回总行数 */
6     public int getRowCount();
7     /** 返回总列数 */
8     public int getColumnCount();
9     /** 返回指定列的名称（表头名称） */
10    public String getColumnName(int columnIndex);
```

```

11  /** 针对列中所有的单元格值，返回最具体的超类。JTable 使用此方法来设置列的默认
    渲染器和编辑器。 */
12  public Class<?>getColumnClass(int columnIndex);
13  /** 判断指定单元格是否可编辑 */
14  public boolean isCellEditable(int rowIndex,int columnIndex);
15  /** 获取指定单元格的值 */
16  public Object getValueAt(int rowIndex,int columnIndex);
17  /** 设置指定单元格的值 */
18  public void setValueAt(Object aValue,int rowIndex,int columnIndex);
19  /** 添加表格模型监听器 */
20  public void addTableModelListener(TableModelListener l);
21  /** 移除表格模型监听器 */
22  public void removeTableModelListener(TableModelListener l);
23  }

```

JRE 中常用的已实现 TableModel 接口的类有两个:

### (1) [javax.swing.table.AbstractTableModel](#)

此抽象类为 TableModel 接口中的大多数方法提供默认实现。它负责管理侦听器，并为生成 TableModelEvents 以及将其调度到侦听器提供方便。要创建一个具体的 TableModel 作为 AbstractTableModel 的子类，只需提供对以下三个方法的实现：

```

1  public int getRowCount();
2  public int getColumnCount();
3  public Object getValueAt(int row,int column);

```

### (2) [javax.swing.table.DefaultTableModel](#)

这是 TableModel 的一个实现，它使用一个 Vector 来存储单元格的值对象，该 Vector 由多个 Vector 组成。DefaultTableModel 还增加了许多方便操作表格数据的方法，例如 **支持 添加 和 删除 行列 等操作**。

下面使用 AbstractTableModel 创建一个表格：

```

1  package com.xiets.swing;
2  import javax.swing.*;
3  import javax.swing.table.AbstractTableModel;
4  import java.awt.*;
5  public class Main{
6      public static void main(String[] args){
7          JFrame jf =new JFrame("测试窗口");
8          jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

```



```

9 // 创建内容面板，使用边界布局
10 JPanel panel =new JPanel(newBorderLayout());
11 // 使用表格模型创建一个表格
12 JTable table =new JTable(newMyTableModel());
13 // 把 表头 添加到容器顶部（使用普通的中间容器添加表格时，表头 和 内容 需要分开添加）
14 panel.add(table.getTableHeader(), BorderLayout.NORTH);
15 // 把 表格内容 添加到容器中心
16 panel.add(table, BorderLayout.CENTER);
17 jf.setContentPane(panel);
18 jf.pack();
19 jf.setLocationRelativeTo(null);
20 jf.setVisible(true);}
21 /**
22  * 表格模型实现，表格显示数据时将调用模型中的相应方法获取数据进行表格内容的显示
23  */
24 publicstaticclassMyTableModelextendsAbstractTableModel{
25 /**
26  * 表头（列名）
27  */
28 private Object[] columnNames ={"姓名","语文","数学","英语","总分"};
29 /** 表格所有行数据 */
30 private Object[][] rowData ={
31 {"张三",80,80,80,240},
32 {"John",70,80,90,240},
33 {"Sue",70,70,70,210},
34 {"Jane",80,70,60,210},
35 {"Joe",80,70,60,210}
36 };
37 /** 返回总行数*/
38 @Override
39 public int getRowCount(){
40 return rowData.length;
41 }
42 /**返回总列数*/
43 @Override
44 public int getColumnCount(){
45 return columnNames.length;
46 }
47 /**返回列名称（表头名称），AbstractTableModel 中对该方法的实现默认是以

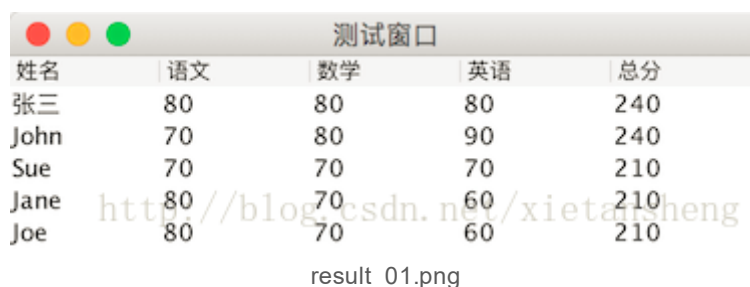
```

```

48  * 大写字母 A 开始作为列名显示，所以这里需要重写该方法返回我们需要的列名。
49  */
50  @Override
51  public String getColumnName(int column){
52  return columnNames[column].toString();
53  }
54  /**
55  * 返回指定单元格的显示的值
56  */
57  @Override
58  public Object getValueAt(int rowIndex,int columnIndex){
59  return rowData[rowIndex][columnIndex];
60  }
61  }
62  }

```

结果展示:



姓名	语文	数学	英语	总分
张三	80	80	80	240
John	70	80	90	240
Sue	70	70	70	210
Jane	80	70	60	210
Joe	80	70	60	210

result\_01.png

用鼠标点击相应的单元格，会发现单元格不可编辑，因为 AbstractTableModel 中对 isCellEditable(...) 方法的实现是返回 false，即单元格不可编辑。如果某些单元格需要支持编辑，可以重写 isCellEditable(...) 方法针对相应的单元格返回 true 即可。

## 6. 表格数据改变的监听 (TableModelListener)

表格的数据维护，对数据的监听，都是由 表格模型 (TableModel) 来维护，通过设置表格模型监听器 (TableModelListener)，可以监听表格单元格数据的更改，表格行列的增加和移除。

设置表格模型监听器主要代码:

```

1  // 先获取 表格模型 对象
2  TableModel tableModel = table.getModel();
3  // 在 表格模型上 添加 数据改变监听器
4  tableModel.addTableModelListener(new TableModelListener(){

```

```

5  @Override
6  public void tableChanged(TableModelEvent e){
7  // 第一个 和 最后一个 被改变的行（只改变了一行，则两者相同）
8  int firstRow = e.getFirstRow();
9  int lastRow = e.getLastRow();
10 // 被改变的列
11 int column = e.getColumn();
12 // 事件的类型，可能的值有：
13 // TableModelEvent.INSERT 新行或新列的添加
14 // TableModelEvent.UPDATE 现有数据的更改
15 // TableModelEvent.DELETE 有行或列被移除
16 int type = e.getType();
17 }
18 });
19

```

## 代码实例:

```

1  package com.xiets.swing;
2  import javax.swing.*.*;
3  import javax.swing.event.TableModelEvent;
4  import javax.swing.event.TableModelListener;
5  import javax.swing.table.AbstractTableModel;
6  import javax.swing.table.TableModel;
7  import java.awt.*.*;
8  public class Main{
9  public static void main(String[] args){
10 JFrame jf =new JFrame("测试窗口");
11 jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
12 // 创建内容面板，使用边界布局
13 JPanel panel =new JPanel(newBorderLayout());
14 // 表头（列名）
15 final Object[] columnNames ={"姓名","语文","数学","英语","总分"};
16 // 表格所有行数据
17 final Object[][] rowData ={
18 {"张三",80,80,80,240},
19 {"John",70,80,90,240},
20 {"Sue",70,70,70,210},
21 {"Jane",80,70,60,210},
22 {"Joe",80,70,60,210}
23 };

```

```

24 // 自定义表格模型，创建一个表格
25 JTable table =new JTable(newAbstractTableModel(){
26 @Override
27 public int getRowCount(){
28 return rowData.length;
29 }
30 @Override
31 public int getColumnCount(){
32 return rowData[0].length;
33 }
34 @Override
35 public String getColumnName(int column){
36 return columnNames[column].toString();
37 }
38 @Override
39 public boolean isCellEditable(int rowIndex,int columnIndex){
40 // 总分列的索引为 4，总分列不允许编辑，其他列允许编辑，
41 // 总分列的数值由 语文、数学、英语 这三列的值相加得出，并同步更新
42 return columnIndex !=4;
43 }
44 @Override
45 public Object getValueAt(int rowIndex,int columnIndex){
46 return rowData[rowIndex][columnIndex];
47 }
48 @Override
49 public void setValueAt(Object newValue,int rowIndex,int columnIndex){
50 // 设置新的单元格数据时，必须把新值设置到原数据数值中，
51 // 待更新UI重新调用 getValueAt(...) 获取单元格值时才能获取到最新值
52 rowData[rowIndex][columnIndex]= newValue;
53 // 设置完数据后，必须通知表格去更新UI（重绘单元格），否则显示的数据不会改变
54 fireTableCellUpdated(rowIndex, columnIndex);
55 }
56 });
57 /** 上面的继承 AbstractTableModel 实现自定义表格模型，功能并不完整，还有很多
    需要自己
58 * 去实现（例如更新数据，通知UI更新，列名称获取等），建议使用 DefaultTableMod
    el 类，
59 * 该类对 TableModel 做了较为完善的实现，支持自动更新数据处理，支持UI自动更
    新，列名称
60 * 处理，添加/移除行列等。无特殊要求不需要重写方法，直接使用即可，如下两行代码
    即可：

```

```

61  */
62  // DefaultTableModel tableModel = new DefaultTableModel(rowData, column
Names); // JTable table = new JTable(tableModel);
63  // 获取 表格模型
64  final TableModel tableModel = table.getModel();
65  // 在 表格模型上 添加 数据改变监听器
66  tableModel.addTableModelListener(new TableModelListener(){
67  @Override
68  public void tableChanged(TableModelEvent e){
69  // 获取 第一个 和 最后一个 被改变的行（只改变了一行，则两者相同）
70  int firstRow = e.getFirstRow();
71  int lastRow = e.getLastRow();
72  // 获取被改变的列
73  int column = e.getColumn();
74  // 事件的类型，可能的值有：
75  // TableModelEvent.INSERT 新行或新列的添加
76  // TableModelEvent.UPDATE 现有数据的更改
77  // TableModelEvent.DELETE 有行或列被移除
78  int type = e.getType();
79  // 针对 现有数据的更改 更新其他单元格数据
80  if(type == TableModelEvent.UPDATE){
81  // 只处理 语文、数学、英语 这三列（索引分别为1、2、3）的分数的更改
82  if(column < 1 || column > 3){
83  return;
84  }
85  // 遍历每一个修改的行，单个学科分数更改后同时更新总分数
86  for(int row = firstRow; row <= lastRow; row++){
87  // 获取当前行的 语文、数学、英语 的值
88  Object chineseObj = tableModel.getValueAt(row, 1);
89  Object mathObj = tableModel.getValueAt(row, 2);
90  Object englishObj = tableModel.getValueAt(row, 3);
91  // 把对象值转换为数值
92  int chinese = 0;
93  try{
94  chinese = Integer.parseInt(""+ chineseObj);
95  }
96  catch(Exception ex){
97  ex.printStackTrace();
98  }
99  int math = 0;


```

```

100  try{
101      math = Integer.parseInt(""+ mathObj);
102  }
103  catch(Exception ex){
104      ex.printStackTrace();
105  }
106  int english =0;
107  try{
108      english = Integer.parseInt(""+ englishObj);
109  }
110  catch(Exception ex){
111      ex.printStackTrace();
112  }
113  // 重新计算新的总分数
114  int totalScore = chinese + math + english;
115  // 将新的分数值设置到总分单元格（总分数的列索引为4）
116  tableModel.setValueAt(totalScore, row,4);}}}});
117  // 把 表头 添加到容器顶部（使用普通的中间容器添加表格时，表头 和 内容 需要分
    开添加）
118  panel.add(table.getHeader(), BorderLayout.NORTH);
119  // 把 表格内容 添加到容器中心
120  panel.add(table, BorderLayout.CENTER);
121  jf.setContentPane(panel);
122  jf.pack();
123  jf.setLocationRelativeTo(null);
124  jf.setVisible(true);
125  }
126  }

```

结果展示，修改单科分数，按回车，看总分变化：



姓名	语文	数学	英语	总分
张三	80	80	80	240
John	70	80	90	240
Sue	70	70	70	210
Jane	80	70	60	210
Joe	80	70	60	210

result\_03.gif

## 7. 表格选择器 (ListSelectionModel)

表格数据的选择使用 **ListSelectionModel** 选择器模型维护，允许用户以不同的模式选中表格中的数据。

ListSelectionModel 的使用:

```
1 // 创建表格
2 final JTable table =newJTable(...);
3 // 设置是否允许单元格单个选中，默认为
4 false;table.setCellSelectionEnabled(boolean cellSelectionEnabled);
5 // 首先通过表格对象 table 获取选择器模型
6 ListSelectionModel selectionModel = table.getSelectionModel();
7 // 设置选择器模式，参数可能的值为：
8 // ListSelectionModel.MULTIPLE_INTERVAL_SELECTION 一次选择一个或多个连续的索引范围（默认）
9 // ListSelectionModel.SINGLE_INTERVAL_SELECTION 一次选择一个连续的索引范围
10 // ListSelectionModel.SINGLE_SELECTION 一次只能选择一个列表索引
11 selectionModel.setSelectionMode(int selectionMode);
12 // 添加选择模型监听器（选中状态改变时回调）
13 selectionModel.addListSelectionListener(new ListSelectionListener(){
14     @Override
15     public void valueChanged(ListSelectionEvent e){
16         // 获取选中的第一行
17         int selectedRow = table.getSelectedRow();
18         // 获取选中的第一列
19         int selectedColumn = table.getSelectedColumn();
20         // 获取选中的所有行
21         int[] selectedRows = table.getSelectedRows();
22         // 获取选中的所有列
23         int[] selectedColumns = table.getSelectedColumns();
24     }
25 });
```

## 8. 单元格的渲染器 (TableCellRenderer)

单元格渲染器用于指定每一个单元格的显示样式。

下面代码实例实现表格的 偶数行背景设置为白色，奇数行背景设置为灰色，第一列的内容水平居中对齐，最后一列的内容水平右对齐，其他列的内容水平左对齐。

```
1 package com.xiets.swing;
2 import javax.swing.*.*;
```

```
3 import javax.swing.table.DefaultTableCellRenderer;
4 import javax.swing.table.TableColumn;
5 import java.awt.*;
6 public class Main{
7     public static void main(String[] args){
8         JFrame jf =newJFrame("测试窗口");
9         jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
10        // 创建内容面板，使用边界布局
11        JPanel panel =new JPanel(newBorder Layout());
12        // 表头（列名）
13        Object[] columnNames ={"姓名","语文","数学","英语","总分"};
14        // 表格所有行数据
15        Object[][] rowData ={
16            {"张三",80,80,80,240},
17            {"John",70,80,90,240},
18            {"Sue",70,70,70,210},
19            {"Jane",80,70,60,210},
20            {"Joe_01",80,70,60,210},
21            {"Joe_02",80,70,60,210},
22            {"Joe_03",80,70,60,210},
23            {"Joe_04",80,70,60,210},
24            {"Joe_05",80,70,60,210}
25        };
26        // 创建一个表格，指定 所有行数据 和 表头
27        JTable table =newJ Table(rowData, columnNames);
28        // 创建单元格渲染器
29        MyTableCellRenderer renderer =new MyTableCellRenderer();
30        // 遍历表格的每一列，分别给每一列设置单元格渲染器
31        for(int i =0; i < columnNames.length; i++){
32            // 根据 列名 获取 表格列
33            TableColumn tableColumn = table.getColumn(columnNames[i]);
34            // 设置 表格列 的 单元格渲染器
35            tableColumn.setCellRenderer(renderer);
36        }
37        // 如果需要自定义表头样式，也可以给表头设置一个自定义渲染
38        // table.getTableHeader().setDefaultRenderer(headerRenderer);
39        // 把 表头 添加到容器顶部（使用普通的中间容器添加表格时，表头 和 内容 需要分开添加）
40        panel.add(table.getTableHeader(), BorderLayout.NORTH);
41        // 把 表格内容 添加到容器中心
```



```

42  panel.add(table, BorderLayout.CENTER);
43  jf.setContentPane(panel);
44  jf.pack();
45  jf.setLocationRelativeTo(null);
46  jf.setVisible(true);
47  }
48  /**
49   * 单元格渲染器，继承已实现渲染器接口的默认渲染器DefaultTableCellRenderer
50   */
51  public static class
MyTableCellRendererextendsDefaultTableCellRenderer{、
52  /**
53   * 返回默认的表单元格渲染器，此方法在父类中已实现，直接调用父类方法返回，在返回
    前做相关参数的设置即可
54   */
55  @Override
56  public Component getTableCellRendererComponent(
57  JTable table, Object value,boolean isSelected,boolean hasFocus,int
    row,int column){
58  // 偶数行背景设置为白色，奇数行背景设置为灰色
59  if(row %2==0){
60  setBackground(Color.WHITE);
61  }else{
62  setBackground(Color.LIGHT_GRAY);
63  }
64  // 第一列的内容水平居中对齐，最后一列的内容水平右对齐，其他列的内容水平左对齐
65  if(column ==0){
66  setHorizontalAlignment(
67  SwingConstants.CENTER);
68  }else if(column ==(table.getColumnCount()-1)){
69  setHorizontalAlignment(SwingConstants.RIGHT);
70  }else{
71  setHorizontalAlignment(SwingConstants.LEFT);
72  }
73  // 设置提示文本，当鼠标移动到当前(row, column)所在单元格时显示的提示文本
74  setToolTipText("提示的内容: "+ row +", "+ column);
75  // PS: 多个单元格使用同一渲染器时，需要自定义的属性，必须每次都设置，否则将自
    动沿用上一次的设置。
76  /**
77   * 单元格渲染器为表格单元格提供具体的显示，实现了单元格渲染器的 DefaultTableCe
    llRenderer 继承自

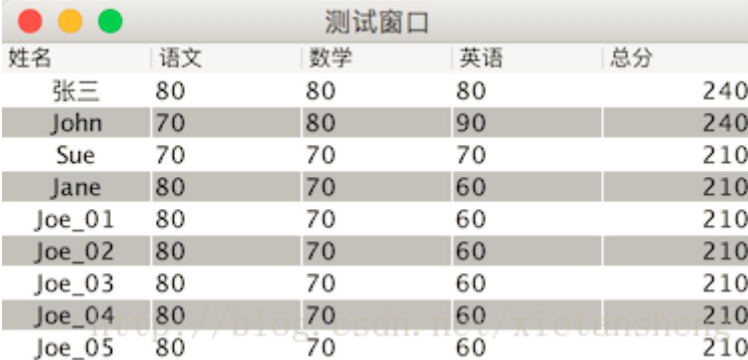
```

```

78  * 一个标准的组件类 JLabel，因此 JLabel 中相应的 API 在该渲染器实现类中都可以
    使用。
79  *
80  * super.getTableCellRendererComponent(...) 返回的实际上是当前对象
    (this)，即 JLabel 实例，
81  * 也就是以 JLabel 的形式显示单元格。
82  *
83  * 如果需要自定义单元格的显示形式（比如显示成按钮、复选框、内嵌表格等），可以在
    此自己创建一个标准组件
84  * 实例返回。
85  */
86  // 调用父类的该方法完成渲染器的其他设置
87  return super.getTableCellRendererComponent(table, value, isSelected, has
    Focus, row, column);
88  }
89  }
90  }

```

结果展示:



姓名	语文	数学	英语	总分
张三	80	80	80	240
John	70	80	90	240
Sue	70	70	70	210
Jane	80	70	60	210
Joe_01	80	70	60	210
Joe_02	80	70	60	210
Joe_03	80	70	60	210
Joe_04	80	70	60	210
Joe_05	80	70	60	210

result\_04.png

## 9. 单元格数据的编辑器 (TableCellEditor)

渲染器是用于正常显示单元格数据时提供显示组件，**编辑器**则是用于编辑单元格数据时显示(使用)的组件。

使用编辑器，可控制单元格内输入的内容格式，监听输入的内容变化等。

常用的编辑组件为 JTextField(文本框)，也可使用 JCheckBox(复选框)、JComboBox(文本框) 等组件作为编辑组件。

编辑器接口为 **TableCellEditor**，只有一个方法，即为指定的单元格提供一个编辑组件。实际使用时通常使用已实现了该接口的默认编辑器 **DefaultCellEditor**。

下面代码使用 DefaultCellEditor 作为基类自定义一个只能输入数字的编辑器:

```

1 package com.xiets.swing;

```

```

2 import javax.swing.*;
3 import javax.swing.table.TableColumn;
4 import java.awt.*;
5 public class Main{
6     public static void main(String[] args){
7         JFrame jf =newJFrame("测试窗口");
8         jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
9         // 创建内容面板，使用边界布局
10        JPanel panel =newJPanel(newBorderLayout());
11        // 表头（列名）
12        Object[] columnNames ={"姓名","语文","数学","英语","总分"};
13        // 表格所有行数据
14        Object[][] rowData ={
15            {"张三",80,80,80,240},
16            {"John",70,80,90,240},
17            {"Sue",70,70,70,210},
18            {"Jane",80,70,60,210},
19            {"Joe",80,70,60,210}
20        };
21        // 创建一个表格，指定 所有行数据 和 表头
22        JTable table =newJTable(rowData, columnNames);
23        // 创建单元格编辑器，使用文本框作为编辑组件
24        MyCellEditor cellEditor =newMyCellEditor(newJTextField());
25        // 遍历表格中所有数字列，并设置列单元格的编辑器
26        for(int i =1; i < columnNames.length; i++){
27            // 根据 列名 获取 表格列
28            TableColumn tableColumn = table.getColumn(columnNames[i]);
29            // 设置 表格列 的 单元格编辑器
30            tableColumn.setCellEditor(cellEditor);
31        }
32        // 把 表头 添加到容器顶部（使用普通的中间容器添加表格时，表头 和 内容 需要分开添加）
33        panel.add(table.getTableHeader(), BorderLayout.NORTH);
34        // 把 表格内容 添加到容器中心
35        panel.add(table, BorderLayout.CENTER);
36        jf.setContentPane(panel);
37        jf.pack();
38        jf.setLocationRelativeTo(null);
39        jf.setVisible(true);}
40    /**

```

```

41  * 只允许输入数字的单元格编辑器
42  */
43  public static class MyCellEditor extends DefaultCellEditor{
44  public MyCellEditor(JTextField textField){
45  super(textField);
46  }
47  public MyCellEditor(JCheckBox checkBox){
48  super(checkBox);
49  }
50  public MyCellEditor(JComboBox comboBox){
51  super(comboBox);
52  }
53  @Override
54  public boolean stopCellEditing(){
55  // 获取当前单元格的编辑器组件
56  Component comp = getComponent();
57  // 获取当前单元格编辑器输入的值
58  Object obj = getCellEditorValue();
59  // 如果当前单元格编辑器输入的值不是数字，则返回 false（表示数据非法，不允许设置，无法保存）
60  if(obj == null || !obj.toString().matches("[0-9]*")){
61  // 数据非法时，设置编辑器组件内的内容颜色为红色
62  comp.setForeground(Color.RED); return false; }
63  // 数据合法时，设置编辑器组件内的内容颜色为黑色
64  comp.setForeground(Color.BLACK);
65  // 合法数据交给父类处理
66  return super.stopCellEditing();
67  }
68  }
69  }

```

结果展示:

测试窗口				
姓名	语文	数学	英语	总分
张三	80	80	80	240
John	70	80	90	240
Sue	70	70	70	210
Jane	80	70	60	210
Joe	80	70	60	210

result\_05.gif

## 10. 表格数据的排序 (RowSorter)

JTable 还支持单击表头按该列进行升序或降序的排序，只需要给表格设置一个表格行排序器 **TableRowSorter**（按字典顺序排序）。

代码实例:

```
1 package com.xiets.swing;
2 import javax.swing.*;
3 import javax.swing.table.DefaultTableModel;
4 import javax.swing.table.TableModel;
5 import javax.swing.table.TableRowSorter;
6 import java.awt.*;
7 public class Main{
8     public static void main(String[] args){
9         JFrame jf =new JFrame("测试窗口");
10        jf.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
11        // 创建内容面板，使用边界布局
12        JPanel panel =new JPanel(new BorderLayout());
13        // 表头（列名）
14        Object[] columnNames ={"姓名","语文","数学","英语","总分"};
15        // 表格所有行数据
16        Object[][] rowData ={
17            {"张三",80,80,80,240},
18            {"John",70,80,90,240},
19            {"Sue",70,70,70,210},
20            {"Jane",80,70,60,210},
21            {"Joe",80,70,60,210}
22        };
23        // 创建 表格模型，指定 所有行数据 和 表头
24        TableModel tableModel =new DefaultTableModel(rowData, columnNames);
25        // 使用 表格模型 创建 表格
26        JTable table =new JTable(tableModel);
27        // 使用 表格模型 创建 行排序器（TableRowSorter 实现了 RowSorter）
28        RowSorter<TableModel> rowSorter =new TableRowSorter<TableModel>(tableModel);
29        // 给 表格 设置 行排序器
30        table.setRowSorter(rowSorter);
31        // 把 表头 添加到容器顶部（使用普通的中间容器添加表格时，表头 和 内容 需要分开添加）
32        panel.add(table.getTableHeader(), BorderLayout.NORTH);
33        // 把 表格内容 添加到容器中心
34        panel.add(table, BorderLayout.CENTER);
35        jf.setContentPane(panel);
```

```
36  jf.pack();
37  jf.setLocationRelativeTo(null);
38  jf.setVisible(true);
39  }
40 }
```

结果展示:



姓名	语文	数学	英语	总分
张三	80	80	80	240
John	70	80	90	240
Sue	70	70	70	210
Jane	80	70	60	210
Joe	80	70	60	210

result\_06.gif