УДК 004.9
DOI: 10.18101/978-5-9793-1626-0-139-143

ИСПОЛЬЗОВАНИЕ МОДЕЛЕЙ ДЛЯ ОРГАНИЗАЦИИ ТАБЛИЦ В JAVA

© **Шадрина Наталья Николаевна**
кандидат физико-математических наук, старший преподаватель
кафедры вычислительной техники и информатики,
Бурятский государственный университет имени Доржи Банзарова
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
E-mail: shadrinann8@yandex.ru

Аннотация. Статья посвящена одной из моделей для формирования и расширенной обработки таблиц, а именно интерфейсу TableModel. Данный интерфейс позволяет работать отдельно с каждой ячейкой таблицы, вносить в ячейки таблиц объекты, устанавливать и получать значения ячеек. Кроме того, данными ячеек могут стать объекты. Подробно рассматривается интерфейс TableModel и некоторые возможности интерфейсов TableColumnModel, SelectionModel, позволяющие применять в таблицах различные эффекты.
Ключевые слова: программирование на Java, объект JTable, интерфейс TableModel

Для цитирования
Шадрина Н. Н. Использование моделей для организации таблиц в Java // Информационные системы и технологии в образовании, науке и бизнесе: материалы региональной научно-практической конференции с международным участием (Улан-Удэ, 1 июля 2021 г.) / отв. ред. А. А. Тонхоноева, науч. ред. Е. Р. Урмакшинова. Улан-Удэ: Изд-во Бурят. гос. ун-та, 2021. С. 139–143.

Данная работа является продолжением статьи, опубликованной годом ранее. Речь идет об организации таблиц, и в предыдущей работе был рассмотрен способ создания таблицы на основе массива с использованием встроенных конструкторов **JTable**. Этот способ идеально подходит для создания элементарных таблиц с минимальным набором функций, он достаточно прост и удобен в исполнении.

Однако не всегда задача ограничивается просмотром данных и несложными вычислениями. В некоторых случаях необходимо дать пользователю дополнительные возможности: выбирать данные из выпадающего списка, менять местами столбцы, добавлять в таблицы пиктограммы или рисунки. Для расширения возможностей обработки таблиц можно использовать интерфейсы **TableModel, TableColumnModel, SelectionModel**. Рассмотрим подробнее интерфейс **TableModel.**

Если по ходу решения поставленной задачи появляется необходимость хранить в ячейках таблиц объекты, а не только простые элементы, то для реализации этой идеи конструкторов **JTable** будет недостаточно. Данную проблему возможно решить, используя интерфейс **TableModel**. Применение интерфейса **TableModel** позволит хранить дополнительную служебную информацию о ячейках, содержащих объекты. Как правило, в ячейке отображается не весь объект, а только один из его атрибутов. Для такого представления применяются специальные отображающие объекты, с помощью которых можно настраивать формат и стиль

видимости данных в ячейке. Отображающие объекты наследуют свойства **DefaultTableCellRenderer**.

С использованием интерфейса **TableModel** появляется возможность описывать каждую ячейку таблицы отдельно и более подробно. Метод **Object getValueAt(i,j)** считывает данные из ячейки таблицы с индексами **i, j** и возвращает ссылку на базовый объект Object. Для определения того, что ячейку таблицы с индексами **i, j** можно редактировать, применяется метод **isCellEditable(i,j)**. Для того чтобы в ячейку таблицы с индексами **i, j** установить значение **a**, используется метод **setValueAt(a,i,j).**

Также интерфейс **TableModel** содержит методы для работы со строками и столбцами. С их помощью можно установить количество строк и столбцов в таблице (методы **int getRowCount(), int getColumnCount()),** определить имя столбца, который будет помещен в заголовок таблицы (метод **setColumnName(столбец)).** Заголовок таблицы JTableHeader появляется, если таблица размещена в панели прокрутки.

Рассмотрим пример создания двух таблиц. Пакет исходных кодов содержит два файла, первый из них **Main.java** является запускаемым. В нем добавляются в проект необходимые модули и создается новый объект класса **TableModelTest.java**. Наибольший интерес представляет класс **TableModelTest.java**. Этот класс наследует объект **JFrame**, в нем создается и активизируется форма.

В разделе описания переменных создаются переменные двух типов: **tableModel** как объект класса **DefaultTableModel:**

private DefaultTableModel tableModel;

и **table1** как объект класса **JTable:**

private JTable table1.

Данные для первой таблицы хранятся в двумерном массиве объектов:

```
private Object[][] array = new String[][] {{ "Курумкан" , "332", "8ч" },
                    { "Максимиха" , "233", "6ч" },
                    { "Турка", "167" , "4ч" }};,
```

данные для второй таблицы формируются в цикле

```
for (int i = 0; i < array.length; i++)
                        tableModel.addRow(array[i]);
```

и добавляются в таблицу с использованием функции **addRow()**.

Заголовки столбцов для первой таблицы формируются в одномерном массиве **columdHeader:**

```
private Object[] columnsHeader = new String[]
   {"Населенный пункт", "Расстояние", "Время в пути"};,
```

для второй таблицы с использованием интерфейса **TableModel:**

tableModel.setColumnIdentifiers(columnsHeader).

В итоге получаем две таблицы, для сравнения размещенные на одной форме (рис. 1).

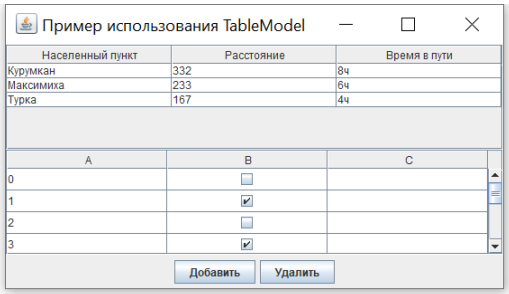

Рис. 1. Пример использования ***TableModel и JTable***

В таблицу, созданную на основе класса **JTable,** можно добавлять и удалять данные с помощью кнопок **Добавить** и **Удалить**.

```
// Создание кнопки добавления строки таблицы
JButton add = new JButton("Добавить");
add.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
// Номер выделенной строки
int idx = table1.getSelectedRow();
// Вставка новой строки после выделенной
tableModel.insertRow(idx + 1, new String[]
{"Северобайкальск" + String.valueOf(table1.getRowCount()),
 "456", "10"});
}
});
```

Рис. 2. Создание кнопки ***Добавить***

```
// Создание кнопки удаления строки таблицы
JButton remove = new JButton("Удалить");
remove.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
// Номер выделенной строки
int idx = table1.getSelectedRow();
// Удаление выделенной строки
tableModel.removeRow(idx);}
});
```

Рис. 3. Создание кнопки ***Удалить***

На рисунках 2 и 3 мы видим пример применения методов **getSelectedRow(), insertRow()** и **removeRow(idx)** класса **DefaultTableModel.**

Перейдем к рассмотрению следующей таблицы. В классе **TableModel-Test.java** размещен внутренний класс **SimpleModel,** наследующий интерфейс **AbstractTableModel.** На основе этой модели данных будет создана вторая таблица. Во внутреннем классе задается количество строк, столбцов, тип хранимых данных, функция определения данных ячейки.

В основном классе создается экземпляр класса **SimpleModel:**

JTable table2 = new JTable(new SimpleModel());

формируется интерфейс панели, добавляются кнопки, и форма выводится на экран.

```
// Создание таблицы на основе модели данных
JTable table2 = new JTable(new SimpleModel());
// Определение высоты строки
table2.setRowHeight(24);

// Формирование интерфейса
Box contents = new Box(BoxLayout.Y_AXIS);
contents.add(new JScrollPane(table1));
contents.add(new JScrollPane(table2));
getContentPane().add(contents);

JPanel buttons = new JPanel();
buttons.add(add);
buttons.add(remove);
getContentPane().add(buttons, "South");
// Вывод окна на экран
setSize(400, 300);
setVisible(true);
```

Рис. 4. Создание таблицы на основе модели данных

Таким образом, применяя различные способы построения таблиц, можно решить поставленную задачу в наиболее удобном для себя виде.

Литература

1. Шилдт Г. Java 8: Руководство для начинающих: перевод с английского. 6-е изд. Москва: Вильямс, 2015. 720 с. Текст: непосредственный.

2. Шилдт Г. Java 8: полное руководство; перевод. с англ. 9-е изд. Москва: Вильямс, 2015. 1376 с. Текст: непосредственный.

3. Эккель Б. Философия Java. Санкт-Петербург: Питер, 2016. 1168 с. Текст: непосредственный.

4. Хорстманн К., Корнелл Г. Java. Библиотека профессионала. Том 1. Основы. Москва: Вильямс, 2016. 866 с. Текст: непосредственный.

5. Мархакшинов А. Л., Шадрина Н. Н. Практикум по программированию на языке Java: практикум. Улан-Удэ: Изд-во Бурят. гос. ун-та, 2017. 65 с. Текст: непосредственный.

6. Шадрина Н. Н. К вопросу о формировании структуры учебного курса «Программирование на Java» // Информационные системы и технологии в образовании, науке и бизнесе: материалы всероссийской научно-практической конференции с международным участием (Улан-Удэ, 5 июля 2019 г.). Улан-Удэ: Изд-во Бурят. гос. ун-та, 2019. С. 44–48. Текст: непосредственный.

7. Шадрина Н. Н. Организация и обработка данных в виде таблиц в Java // Информационные системы и технологии в образовании, науке и бизнесе: материалы региональной научно-практической конференции с международным участием (Улан-Удэ, 1 июля 2020 г.). Улан-Удэ: Изд-во Бурят. гос. ун-та, 2019. С. 106–110. Текст: непосредственный.

ORGANIZATION AND PROCESSING OF DATA IN THE FORM OF TABLES IN JAVA

Natalia N. Shadrina
Cand. Sci. (Phys. and Math.), Senior Lecturer,
Department of Computer Science and Informatics,
Dorzhi Banzarov Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: shadrinann8@yandex.ru

Abstract. The article is devoted to one of the models for the formation and advanced processing of tables, namely the TableModel interface. This interface allows you to work separately with each table cell, add objects to table cells, and set and get cell values. In addition, the cell data can be objects. The TableModel interface is discussed in detail, and some features of the TableColumnModel and selectionModel interfaces that allow you to apply various effects.
Keywords: Java programming, JTable object, TableModel interface