Меню

Меню — еще один основной элемент построения GUI на основе библиотеки Swing. Меню являются неотъемлемой частью многих приложений, поскольку они представляют пользователю функциональные возможности прикладной программы. В силу своего особого значения меню нашли широкую поддержку в библиотеке Swing. Именно в меню проявляется истинный потенциал библиотеки Swing. Система меню в Swing поддерживает следующие главные элементы меню.

Строка меню, отображающая главное меню прикладной программы.

Стандартное меню, которое может содержать выбираемые пункты или другие меню, иначе называемые подменю.

Всплывающее или контекстное меню, которое обычно активизируется щелчком правой кнопкой мыши.

Панель инструментов, предоставляющая быстрый доступ к функциональным возможностям прикладной программы (зачастую параллельно с пунктами меню).

Действие, позволяющее одному объекту управлять разными (двумя или больше) компонентами. Действия обычно употребляются вместе с меню и панелями инструментов.

Кроме того, в Swing поддерживают оперативные клавиши, позволяющие быстро выбирать пункты, не активизируя само меню, а также мнемонику для выбора пунктов меню нажатием клавиш после отображения самого меню.

Основные положения о меню

Система меню в Swing опирается на ряд связанных с ней классов. В табл. перечислены классы, составляющие основу системы меню в Swing. Эти меню допускают специальную настройку, если таковая требуется, в самых широких пределах, но, как правило, классы меню применяются без изменений, поскольку они поддерживают все необходимые варианты построения меню. Например, в меню совсем не трудно ввести клавиатурные сокращения.

Таблица. Основные классы меню в Swing

JMenuBar	Объект этого класса содержит меню верхнего уровня для приложения
JMenu	Стандартное меню, состоящее из одного или нескольких пунктов в виде объектов типа JMenuItems
JMenuItem	Объект этого класса представляет пункты, наполняющие меню
JCheckBoxMenuItem	Отмечаемый флажком пункт меню
JRadioButtonMenuItem	Отмечаемый кнопкой-переключателем пункт меню
JSeparator	Визуальный разделитель пунктов меню
JPopupMenu	Меню, которое обычно активизируется щелчком правой кнопкой мыши

Чтобы построить меню верхнего уровня для приложения, сначала следует создать объект класса JMenuBar. Этот класс служит контейнером для меню. В экземпляр класса JMenuBar обычно вводятся экземпляры класса JMenu, причем каждый объект типа JMenu определяет отдельное меню. Это означает, что каждый объект типа JMenu содержит один или несколько выбираемых пунктов. А пункты, отображаемые объектами типа JMenu, в свою очередь, являются объектами класса JMenuItem. Следовательно, класс JMenuItem определяет пункт меню, выбираемый пользователем.

Помимо меню, раскрывающихся из строки меню, можно создавать автономные всплывающие меню. С этой целью следует сначала создать объект типа JPopupMenu, а затем ввести в него объекты типа JMenuItem в виде пунктов меню. Как правило, всплывающее (или контекстное) меню активизируется щелчком правой кнопкой мыши, когда курсор находится на том компоненте, для которого определено всплывающее меню.

В меню можно вводить не только стандартные пункты, но и пункты, отмечаемые флажками или кнопками-переключателями. В частности, отмечаемый флажком пункт меню создается средствами класса JCheckBoxMenuItem, а отмечаемый кнопкой-переключателем пункт меню — средствами класса JRadioButtonMenuItem. Оба эти класса расширяют класс JMenuItem. Их можно применять в стандартных и всплывающих меню.

Класс JToolBar позволяет создать панель инструментов в виде автономного компонента, связанного с меню. Такой компонент нередко служит для быстрого доступа к функциональным возможностям, имеющимся в меню приложения. Например, панель инструментов может предоставлять быстрый доступ к командам форматирования, которые поддерживаются в текстовом редакторе. А служебный класс JSeparator позволяет создать линию, разделяющую пункты меню.

В отношении меню Swing следует иметь в виду, что каждый пункт меню представлен объектом класса, расширяющего класс AbstractButton. Напомним, что класс AbstractButton служит суперклассом для всех компонентов кнопок в библиотеке Swing, в том числе и компонента типа JButton. Следовательно, все пункты меню, по существу, являются кнопками. Очевидно, что пункты не похожи внешне на кнопки, когда выбираются из меню, но они действуют, главным образом, как кнопки. Так, если выбрать пункт меню, то событие действия будет сгенерировано таким же образом, как и при нажатии экранной кнопки.

Следует также иметь в виду, что класс JMenuItern служит суперклассом для класса JMenu. Это дает возможность создавать подменю, т. е. одни меню, вложенные в другие. Чтобы построить подменю, следует сначала создать меню в виде объекта типа JMenu и наполнить его пунктами, а затем ввести его в другой объект типа JMenu. Этот процесс демонстрируется ниже.

Как упоминалось выше, при выборе пункта меню генерируется событие действия. Символьная строка с командой действия, связанная с этим событием действия, по умолчанию обозначает наименование выбранного пункта меню. Следовательно, проанализировав команду действия, можно выяснить, какой именно пункт меню был выбран. Безусловно, для обработки событий действия от каждого пункта меню можно воспользоваться отдельными анонимными классами. В этом случае выбранный пункт меню уже известен, и поэтому нет нужды анализировать команду действия, чтобы выяснить, какой именно пункт меню был выбран.

Меню могут генерировать и другие типы событий. Например, всякий раз, когда активизируется или выбирается меню или же отменяется его выбор, генерируется событие типа MenuEvent, которое может отслеживаться приемником событий из интерфейса MenuListener. К числу других событий, связанных с меню, относятся события типа MenuKeyEvent, MenuDragMouseEvent и PopupMenuEvent. Но, как правило, отслеживать требуется только события действия, и поэтому будет рассмотрена обработка только этих событий.

Краткий обзор классов JMenuBar, JMenu и JMenuItem

Прежде чем строить меню, нужно иметь хотя бы какое-то представление о трех основных классах меню: JMenuBar, JMenu и JMenuItem. Эти классы составляют тот минимум средств, которые требуются для построения главного меню приложения. Кроме того, классы JMenu и JMenuItem служат для построения всплывающих меню. Таким образом, эти классы образуют основание системы меню в Swing.

Класс JMenuBar

Как упоминалось выше, класс JMenuBar, по существу, служит контейнером для меню. Как и все остальные классы компонентов, он наследует от класса JCornponent, а тот — от классов Container и Cornponent. У этого класса имеется единственный конструктор по умолчанию. Следовательно, строка меню первоначально будет пустой, и поэтому ее придется наполнить меню, прежде чем ею воспользоваться. У каждого приложения имеется одна и только одна строка меню. В классе JMenuBar определяется несколько методов, но на практике применяется только метод add(). Этот метод вводит меню в виде объекта типа JMenu в строку меню. Ниже приведена его общая форма.

```
JMenu add(JMenu меню)
```

Здесь параметр меню обозначает экземпляр класса JMenu, вводимый в строку меню. Меню располагаются в строке меню слева направо в том порядке, в каком они вводятся. Если же меню требуется ввести в конкретном месте, следует воспользоваться приведенным ниже вариантом метода add(), наследуемого из класса Container.

```
Component add(Component меню, int индекс)
```

Здесь параметр меню обозначает конкретное меню, вводимое по указанному индексу. Индексирование начинается с нуля, причем нулевой индекс обозначает крайнее слева меню в строке меню. Иногда из строки меню необходимо удалить меню, которое больше не требуется. Для этого достаточно вызвать метод remove(), наследуемый из класса Container. У этого метода имеются следующие общие формы:

```
void remove(Component меню)
void remove(int индекс)
```

Здесь параметр меню обозначает ссылку на удаляемое меню, а параметр индекс — указанный индекс удаляемого меню. Индексация меню начинается с нуля. Иногда полезным оказывается и метод getMenuCount(). Ниже приведена его общая форма. Этот метод возвращает количество элементов, содержащихся в строке меню.

```
int getMenuCount()
```

В классе JMenuBar определяются и другие методы, которым можно найти специальное применение. Например, вызвав метод getSubElements(), можно получить массив ссылок на меню, находящихся в строке меню, а, вызвав метод isSelected(), — определить, выбрано ли конкретное меню. Как только строка меню будет создана и наполнена отдельными меню, ее можно ввести в контейнер типа JFrame, вызвав метод setJMenuBar() для экземпляра класса JFrame. (Строки меню не вводятся на панели содержимого). Ниже приведена общая форма метода setJMenuBar().

```
void setJMenuBar(JMenuBar строка_меню)
```

Здесь параметр строка_меню обозначает ссылку на конкретную строку меню. Указанная строка меню отображается на позиции, определяемой стилем оформления GUI. Как правило, меню располагается вдоль верхнего края окна приложения.

КлассJMenu

Класс JMenu инкапсулирует меню, наполняемое пунктами в виде объектов типа JMenuItem. Как упоминалось ранее, этот класс наследует от класса JMenuItem. Это означает, что одно меню типа JMenu можно выбирать из другого. Следовательно, одно меню может служить подменю для другого. В классе JMenu определяется целый ряд конструкторов. Мы же будем использовать следующий конструктор:

```
JMenu(String имя)
```

Этот конструктор создает меню с заголовком, обозначаемым параметром имя. Разумеется, присваивать наименование меню совсем не обязательно. Для создания безымянного меню можно воспользоваться следующим конструктором по умолчанию:

```
JMenu()
```

В классе JMenu поддерживаются и другие конструкторы. Но в любом случае создаваемое с их помощью меню будет пустым до тех пор, пока в него не будут введены отдельные пункты. Кроме того, в классе JMenu определяется немало методов. Ниже вкратце описываются лишь наиболее употребительные из них. Для ввода пункта в меню служит метод add(), у которого имеется несколько общих форм. Ниже приведены две его общие формы.

```
JMenuItem add(JMenuItem пункт)
JMenuItem add(Component пункт, int индекс)
```

Здесь параметр пункт обозначает пункт, вводимый в меню. В первой форме заданный пункт вводится в конце меню, а во второй форме — по указанному индексу. Как и следовало ожидать, индексирование пунктов меню начинается с нуля. В обеих рассматриваемых здесь формах метод add() возвращает ссылку на вводимый пункт меню. Для ввода пунктов в меню можно также воспользоваться методом insert(). В меню можно ввести разделитель (объект типа JSeparator), вызвав метод addSeparator(), как показано ниже. Разделитель вводится в конце меню.

```
void addSeparator()
```

А для ввода разделителя в нужном месте меню можно вызвать метод insertSeparator(). Ниже приведена его общая форма, где параметр индекс обозначает отсчитываемый от нуля индекс, по которому вводится разделитель.

```
void insertSeparator(int индекс)
```

Вызвав метод remove(), можно удалить пункт из меню. Ниже приведены две его общие формы, где параметр меню обозначает ссылку на удаляемый пункт меню, а параметр индекс — указанный индекс удаляемого пункта меню.

```
void remove(JMenuItem меню)
void remove(int индекс)
```

Вызвав метод getMenuComponentCount(), можно получить количество пунктов в меню:

```
int getMenuComponentCount()
```

А если вызвать метод getMenuComponents(), то можно получить массив из пунктов меню, как показано ниже. Этот метод возвращает массив, содержащий компоненты меню.

```
Component[] getMenuComponents()
```

Класс JMenuItem

Класс JMenuItem инкапсулирует пункт меню. Выбор этого элемента GUI может быть связан с некоторым действием прикладной программы, например с сохранением данных или закрытием файла, или же с отображением подменю. Как упоминалось ранее, класс JMenuItem является производным от класса AbstractButton, и поэтому каждый пункт меню можно рассматривать как особого рода кнопку. Следовательно, когда выбирается пункт меню, событие действия генерируется таким же образом, как и при нажатии экранной кнопки типа JButton. В классе JMenuItem определено немало конструкторов. Ниже приведены конструкторы данного класса.

```
JMenuItem(String имя)
JMenuItem(Icon изображение)
JMenuItem(String имя, Icon изображение)
JMenuItem(String имя, int мнемоника)
JMenuItem(Action действие)
```

Первый конструктор создает пункт меню, обозначаемый параметром имя. Второй конструктор создает пункт меню, отображающий указанное изображение. Третий конструктор создает пункт меню, обозначаемый параметром имя и отображающий указанное изображение. Четвертый конструктор создает пункт меню, обозначаемый параметром имя и использующий указанную клавиатурную мнемонику. Эта мнемоника позволяет выбрать пункт меню нажатием указанной клавиши. И последний конструктор создает пункт меню, используя сведения, обозначаемые параметром действие. В данном классе поддерживается также конструктор по умолчанию. Благодаря тому, что класс JMenuItem наследует от класса AbstractButton, функциональные возможности последнего доступны для пунктов меню. В частности, для меню нередко оказывается полезным метод setEnabled(), позволяющий включать или отключать отдельные пункты меню. Ниже приведена общая форма этого метода.

```
void setEnabled(boolean включить)
```

Если параметр включить принимает логическое значение true, то пункт меню включается. А если этот параметр принимает логическое значение false, то пункт меню отключается и не может быть выбран.

Создание главного меню

По традиции наиболее употребительным считается главное меню. Оно доступно из строки меню и определяет все (или почти все) функциональные возможности приложения. К счастью, библиотека Swing значительно упрощает создание главного меню и управление им. Ниже будет показано, как создать элементарное главное меню и ввести в него отдельные варианты выбора. Процесс создания главного меню состоит из нескольких стадий. Сначала создается объект типа JMenuBar, который будет содержать отдельные меню. Затем создается каждое меню, располагаемое в строке меню. В общем, построение меню начинается с создания объекта типа JMenu, в который затем вводятся пункты меню в виде объектов типа JMenuItem. Как только отдельные меню будут созданы, они вводятся в строку меню. А саму строку меню следует ввести во фрейм, вызвав метод setJMenuBar(). И, наконец, каждый пункт меню должен быть дополнен приемником действий, обрабатывающим событие действия, наступающее при выборе отдельного пункта из меню. Процесс создания меню и управления ими станет понятнее, если продемонстрировать его на конкретном примере. Ниже приведена прикладная программа, в которой создается простая строка меню, состоящая из трех меню. Первым из них является меню File (Файл), состоящее из выбираемых пунктов Open (Открыть), Close (Закрыть), Save (Сохранить) и Exit (Выход). Второе меню называется Options (Параметры) и состоит из двух подменю: Colors (Цвета) и Priority (Приоритет). Третье меню называется Help (Справка) и состоит из единственного пункта About (О программе). Когда выбирается пункт меню, его наименование отображается в метке на панели содержимого.

```
//продемонстрировать простое главное меню
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class MenuDemo implements ActionListener {
      JLabel jlab;
      MenuDemo() {
            //создать новый контейнер типа JFrame
            JFrame jfrm = new JFrame("Menu Demo");
            //демонстрация меню
            //указать диспетчер поточной компоновки типа FlowLayout
            jfrm.setLayout(new FlowLayout());
            //задать исходные размеры фрейма
```

```
        jfrm.setSize(220, 200);
        //завершить прикладную программу, как только
        //пользователь закроет ее окно
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //создать метку для отображения результатов выбора из меню
        jlab = new JLabel();

        //создать строку меню
        JMenuBar jmb = new JMenuBar();
        //создать меню File
        JMenu jmFile = new JMenu("File"); //Файл
        JMenuItem jmiOpen = new JMenuItem("Open"); //Открыть
        JMenuItem jmiClose = new JMenuItem("Close"); //Закрыть
        JMenuItem jmiSave = new JMenuItem("Save"); //Сохранить
        JMenuItem jmiExit = new JMenuItem("Exit"); //Выход
        jmFile.add(jmiOpen);
        jmFile.add(jmiClose);
        jmFile.add(jmiSave);
        jmFile.addSeparator();
        jmFile.add(jmiExit);
        jmb.add(jmFile);
        //создать меню Options
        JMenu jmOptions = new JMenu("Options"); //Параметры
        //создать подменю Colors
        JMenu jmColors = new JMenu("Colors"); //Цвета
        JMenuItem jmiRed = new JMenuItem("Red"); //Красный
        JMenuItem jmiGreen = new JMenuItem ("Green"); //Зеленый
        JMenuItem jmiBlue = new JMenuItem ("Blue"); //Синий
        jmColors.add(jmiRed);
        jmColors.add(jmiGreen);
        jmColors.add(jmiBlue);
        jmOptions.add(jmColors);
        //создать подменю Priority
        JMenu jmPriority = new JMenu("Priority"); //Приоритет
        JMenuItem jmiHigh = new JMenuItem("High"); //Высокий
        JMenuItem jmiLow = new JMenuItem("Low"); //Низкий
        jmPriority.add(jmiHigh);
        jmPriority.add(jmiLow);
        jmOptions.add(jmPriority);
        //создать пункт меню Reset
        JMenuItem jmiReset = new JMenuItem("Reset"); //Сбросить
        jmOptions.addSeparator();
        jmOptions.add(jmiReset);
        //ввести все выбираемые меню в строку меню
        jmb.add(jmOptions);
        //создать меню Help
        JMenu jmHelp = new JMenu("Help"); //Справка
        JMenuItem jmiAbout = new JMenuItem("About"); //О программе
        jmHelp.add(jmiAbout);
        jmb.add(jmHelp);
        //ввести приемники действий от пунктов меню
        jmiOpen.addActionListener(this);
        jmiClose.addActionListener(this);
        jmiSave.addActionListener(this);
        jmiExit.addActionListener(this);
        jmiRed.addActionListener(this);
        jmiGreen.addActionListener(this);
        jmiBlue.addActionListener(this);
        jmiHigh.addActionListener(this);
        jmiLow.addActionListener(this);
        jmiReset.addActionListener(this);
        jmiAbout.addActionListener(this);
        //ввести метку на панели содержимого
        jfrm.add(jlab);
        //ввести строку меню во фрейм
        jfrm.setJMenuBar(jmb);
        //отобразить фрейм
        jfrm.setVisible(true);
}
```

```
	//обработать события действия от пунктов меню
	public void actionPerformed(ActionEvent ae) {
		//получить команду действия из выбранного меню
		String comStr = ae.getActionCommand();
		//выйти из программы, если пользователь
		//выберет пункт меню Exit
		if(comStr.equals("Exit"))
			System.exit(0);
		//в противном случае отобразить результат
		//выбора из меню
		jlab.setText(comStr + " Selected"); //Выбрано указанное
	}

	public static void main(String args[]) {
		//создать фрейм
		new MenuDemo();
	}
}
```

Рассмотрим подробнее, каким образом в данной программе создаются меню, начиная с конструктора класса MenuDemo. Сначала в этом конструкторе создается фрейм типа JFrame, устанавливается диспетчер компоновки, задаются размеры фрейма и стандартная операция закрытия прикладной программы. Затем создается метка типа JLabel, предназначенная для отображения результатов выбора из меню. Далее создается строка меню, а ссылка на нее присваивается переменной jmb, как показано ниже.

```
//создать строку меню
JMenuBar jmb = new JMenuBar();
```

Затем создается меню File и его пункты, а ссылка на него присваивается переменной jmFile:

```
//создать меню File
JMenu jmFile = new JMenu("File"); //Файл
JMenuItem jmiOpen = new JMenuItem("Open"); //Открыть
JMenuItem jmiClose = new JMenuItem("Close"); //Закрыть
JMenuItem jmiSave = new JMenuItem("Save"); //Сохранить
JMenuItem jmiExit = new JMenuItem("Exit"); //Выход
```

Имена отдельных пунктов Open, Close, Save и Exit меню File будут отображаться на месте метки при их выборе. Далее эти пункты вводятся в меню File следующим образом:

```
jmFile.add(jmiOpen);
jmFile.add(jmiClose);
jmFile.add(jmiSave);
jmFile.addSeparator();
jmFile.add(jmiExit);
```

И наконец, меню File вводится в строку главного меню, как показано ниже.

```
jmb.add(jmFile);
```

По завершении приведенной выше последовательности кода строка меню будет содержать один элемент: меню File, тогда как само меню File — пункты Open, Close, Save и Exit. (Обратите внимание на разделитель, введенный перед пунктом меню Exit. Он визуально отделяет пункт Exit от остальных пунктов меню File).

Аналогичным образом создается и меню Options, но оно состоит из двух подменю, Colors и Priority, и одного пункта Reset. Сначала подменю создаются по отдельности, а затем вводятся в строку меню. И последним в данное меню вводится пункт Reset. После этого меню Options вводится в строку главного меню. Подобным же образом создается и меню Help.

Класс MenuDemo реализует интерфейс ActionListener, а события действия, генерируемые при выборе меню, обрабатываются методом actionPerformed(), определяемым в классе MenuDemo. Следовательно, для пунктов меню в данной программе вводятся приемники действий по ссылке this. В то же время приемники действий не вводятся для пунктов меню Colors и Priority, поскольку они не выбираются, а просто активизируют подменю.

И, наконец, строка меню вводится во фрейм следующим образом:

```
jfrm.setJMenuBar(jmb);
```

Как упоминалось выше, строки меню вводятся не на панели содержимого, а непосредственно во фрейм типа JFrame. В методе actionPerformed() обрабатываются события действия, генерируемые в меню. Для конкретного события в этом методе вызывается метод getActionCommand(), чтобы получить символьную строку с командой действия, связанной с выбором из меню. Ссылка на эту символьную строку сохраняется в переменной comStr. Затем эта строка проверяется на наличие команды действия

```
if(comStr.equals("Exit"))
      System.exit(0);
```

Если символьная строка содержит команду действия «Exit», то программа завершается вызовом метода System.exit(). Этот метод немедленно завершает программу и передает свой аргумент в качестве кода состояния вызывающему процессу, которым обычно является операционная система или браузер. Условно нулевой код состояния означает нормальное завершение программы, а любой другой код состояния — ненормальное ее завершение. Результат выбора всех остальных пунктов меню, кроме Exit, отображается на месте заданной метки. Можете поэкспериментировать с прикладной программой MenuDemo, попробовав ввести еще одно меню или дополнительные пункты в уже имеющиеся меню. Ваша главная задача — уяснить основные принципы создания меню.

Ввод мнемоники и оперативных клавиш в меню

Меню, созданное в предыдущем примере, вполне работоспособно, но его можно усовершенствовать. Меню реальных прикладных программ обычно поддерживают клавиатурные сокращения, предоставляя опытным пользователям возможность быстро выбирать пункты меню. Клавиатурные сокращения принимают две формы: мнемонику и оперативные клавиши. Применительно к меню мнемоника определяет клавишу, нажатием которой выбирается отдельный пункт активного меню. Следовательно, мнемоника позволяет выбрать с клавиатуры пункты того меню, которое уже отображается, а оперативная клавиша — сделать то же самое, не активизируя предварительно меню.

Мнемонику можно указать как для объектов типа JMenuItem (пунктов меню), так и для объектов типа JMenu (самих меню). Мнемоника для объекта типа JMenuItem задается двумя способами. Во-первых, ее можно указать при создании данного объекта с помощью следующего конструктора:

```
JMenuItem(String имя, int мнемоника)
```

В данном случае наименование пункта меню передается в качестве параметра имя, а мнемоника — в качестве параметра мнемоника. И, во-вторых, мнемонику для пунктов меню (объектов типа JMenuItem) или самого меню (объекта типа JMenu) можно задать, вызвав метод setMnemonic(). Этот метод наследуется из класса AbstractButton обоими классами JMenuItem и JMenu. Ниже приведена его общая форма.

```
void setMnemonic(int мнемоника)
```

Здесь параметр мнемоника обозначает задаваемую мнемонику. Этот параметр должен принимать значение одной из констант, определяемых в классе java.awt.event.KeyEvent, в том числе KeyEvent.VK_F или KeyEvent.VK_Z. Мнемоника не зависит от регистра букв, а, следовательно, мнемоники VK_A и VK_a равнозначны. По умолчанию подчеркивается первая буква в наименовании пункта меню, совпадающая с заданной мнемоникой. Если же требуется подчеркнуть другую букву, следует указать ее индекс в качестве аргумента метода setDisplayedMnemonicIndex(), наследуемого классами JMenuItem и JMenu из класса AbstractButton. Ниже приведена его общая форма, где параметр индекс обозначает указанный индекс подчеркиваемой буквы.

```
void setDisplayedMnemonicIndex(int индекс)
```

Оперативная клавиша может быть связана с объектом типа JMenuItem. Она задается с помощью метода setAccelerator(), как показано ниже.

```
void setAccelerator(KeyStroke сочетание_клавиш)
```

Здесь параметр сочетание_клавиш обозначает комбинацию клавиш, нажимаемых для выбора пункта меню. Класс KeyStroke содержит ряд методов для построения разнотипных комбинаций клавиш быстрого выбора пунктов меню. Ниже приведены три общие формы одного из таких методов.

```
static KeyStroke getKeyStroke(char символ)
static KeyStroke getKeyStroke(Character символ, int модификатор)
static KeyStroke getKeyStroke(int символ, int модификатор)
```

Здесь параметр символ обозначает конкретный символ оперативной клавиши. В первой форме данного метода этот символ указывается в виде значения типа char; во второй форме — в виде объекта типа Character; в третьей форме в виде значения константы типа KeyEvent, как пояснялось выше. И в качестве параметра модификатор

следует указать значение одной или нескольких констант, перечисленных ниже и определяемых в классе java.awt.event.InputEvent.

```
InputEvent.ALT_DOWN_MASK
InputEvent.ALT_GRAPH_DOWN_MASK
InputEvent.CTRL_DOWN_MASK
InputEvent.META_DOWN_MASK
InputEvent.SHIFT_DOWN_MASK
```

Так, если в качестве параметра символ передать методу getKeyStroke() константу VK_A, а в качестве параметра модификатор — константу InputEvent.CTRL_DOWN_MASK, то для выбора пункта меню будет задана комбинация оперативных клавиш <Ctrl+A>. В приведенном ниже фрагменте кода в меню File, которое создается в рассмотренном ранее примере прикладной программы MenuDemo, вводится мнемоника и комбинации оперативных клавиш. После ввода этого фрагмента кода в программу MenuDemo меню File можно будет оперативно выбрать, нажав комбинацию клавиш <Alt+F>, а затем воспользоваться мнемоникой O, C, S или E для быстрого выбора соответствующих пунктов этого меню. С другой стороны, эти пункты меню можно выбрать непосредственно, нажимая комбинации клавиш <Ctrl+O>, <Ctrl+C>, <Ctrl+S> или <Ctrl+E> соответственно.

```
//создать меню File с мнемоникой и оперативными клавишами
JMenu jmFile = new JMenu("File");
jmFile.setMnemonic(KeyEvent.VK_F);
JMenuItem jmiOpen = new JMenuItem("Open", KeyEvent.VK_O);
jmiOpen.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O, InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiClose = new JMenuItem("Close", KeyEvent.VK_C);
jmiClose.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C,
                        InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiSave = new JMenuItem("Save", KeyEvent.VK_S);
jmiSave.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
                        InputEvent.CTRL_DOWN_MASK));
JMenuItem jmiExit = new JMenuItem("Exit", KeyEvent. VK_E);
jmiExit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_E,
                        InputEvent.CTRL_DOWN_MASK));
```

Создание всплывающего меню

Всплывающее меню служит весьма распространенной альтернативой или дополнением строки меню. Как правило, всплывающее меню активизируется щелчком правой кнопкой мыши на компоненте. Всплывающие меню поддерживаются в классе JPopupMenu из библиотеки Swing. У этого класса имеются два конструктора, но в примере ниже применяется только следующий конструктор по умолчанию:

```
JPopupMenu()
```

Этот конструктор создает стандартное всплывающее меню. А другой конструктор позволяет указать заголовок всплывающего меню. Порядок отображения этого заголовка зависит от выбранного стиля оформления GUI. В общем, всплывающие меню создаются таким же образом, как и обычные меню. Сначала создается объект типа JPopupMenu, а затем он наполняется пунктами всплывающего меню. Обработка результатов выбора пунктов всплывающего меню выполняется тем же самым способом: приемом событий действия. А отличаются всплывающие меню от обычных главным образом процессом их активации. Процесс активации всплывающего меню разделяется на три стадии.

Регистрация приемника событий от мыши.

Отслеживание в приемнике событий момента запуска всплывающего меню.

Отображение всплывающего меню при получении события запуска всплывающего меню, для чего вызывается метод show().

Рассмотрим каждую из этих стадий подробнее. Всплывающее меню обычно активизируется щелчком правой кнопкой мыши, когда курсор находится на том компоненте, для которого определено всплывающее меню. Следовательно, запуск всплывающего меню происходит после щелчка правой кнопкой мыши на компоненте, снабженном всплывающим меню. Для приема события запуска всплывающего меню сначала реализуется интерфейс MouseListener, а затем регистрируется приемник подобных событий. С этой целью вызывается метод addMouseListener(). Интерфейсе MouseListener определяются приведенные ниже методы.

```
void mouseClicked(MouseEvent событие_от_мыши)
void mouseEntered(MouseEvent событие_от_мыши)
void mouseExited(MouseEvent событие_от_мыши)
```

```
void mousePressed(MouseEvent событие_от_мыши)
void mouseReleased(MouseEvent событие_от_мыши)
```

Из всех перечисленных выше методов для создания всплывающих меню особое значение имеют два метода: mousePressed() и mouseReleased(). В зависимости от выбранного стиля оформления GUI любой из этих методов приема событий от мыши может запустить всплывающее меню. Именно поэтому зачастую проще воспользоваться классом MouseAdapter, чтобы реализовать интерфейс MouseListener и просто переопределить его методы mousePressed() и mouseReleased(). В классе MouseEvent определяется целый ряд методов, но только четыре из них обычно требуются для активации всплывающего меню. Эти методы перечислены ниже.

```
int getX()
int getY()
boolean isPopupTrigger()
Component getComponent()
```

Текущие координаты X, Y положения курсора относительно источника события определяются с помощью методов getX() и getY() соответственно. Полученные с их помощью координаты служат для указания левого верхнего угла всплывающего меню при его отображении. Метод isPopupTrigger() возвращает логическое значение true, если событие от мыши обозначает запуск всплывающего меню, а иначе — логическое значение false. Чтобы получить ссылку на компонент, сгенерировавший событие от мыши, следует вызвать метод getCornponent(). Чтобы отобразить всплывающее меню, следует вызвать метод show(), определяемый в классе JPopupMenu следующим образом:

```
void show(Component вызывающий, int X, int Y)
```

Здесь параметр вызывающий обозначает компонент, относительно которого отображается всплывающее меню; X и Y — координаты X, Y местоположения левого верхнего угла всплывающего меню относительно вызывающего компонента. Чтобы получить вызывающий компонент, проще всего вызвать метод getCornponent() для объекта события, передаваемого обработчику событий от мыши.

Все сказанное выше относительно всплывающих меню можно применить на практике, введя всплывающее меню Edit (Правка) в пример программы MenuDemo, представленный выше. Это меню будет состоять из трех пунктов: Cut (Вырезать), Copy (Скопировать) и Paste (Вставить). Сначала введите в программу MenuDemo приведенную ниже строку кода, где переменная jpu объявляется для хранения ссылки на всплывающее меню.

```
final JPopupMenu jpu;
```

Затем введите следующий фрагмент кода в конструктор класса MenuDemo:

```
//создать всплывающее меню Edit
jpu = new JPopupMenu();
//создать пункты всплывающего меню
JMenuItem jmiCut = new JMenuItem("Cut");
JMenuItem jmiCopy = new JMenuItem("Copy");
JMenuItem jmiPaste = new JMenuItem("Paste");
//ввести пункты во всплывающее меню
jpu.add(jmiCut);
jpu.add(jmiCopy);
jpu.add(jmiPaste);

//ввести приемник событий запуска всплывающего меню
jfrm.addMouseListener(new MouseAdapter() {
      public void mousePressed(MouseEvent me) {
            if(me.isPopupTrigger())
                  jpu.show(me.getComponent(), me.getX(), me.getY() );
      }

      public void mouseReleased(MouseEvent me) {
            if(me.isPopupTrigger())
                  jpu.show(me.getComponent(), me.getX(), me.getY());
      }
});
```

Сначала в приведенном выше фрагменте кода создается экземпляр класса JPopupMenu, сохраняемый в переменной jpu. Затем в нем создаются обычным образом три пункта меню, Cut, Copy и Paste, которыми наполняется объект, хранящийся в переменной jpu. На этом создание всплывающего меню Edit завершается.

Всплывающие меню не вводятся ни в строку меню, ни в какой-нибудь другой объект. Далее создается анонимный внутренний класс для ввода приемника событий от мыши типа MouseListener. Этот класс опирается на класс MouseAdapter, а следовательно, в приемнике событий необходимо переопределить только те методы, которые имеют отношение к всплывающему меню, а именно mousePressed() и mouseReleased(). В классе адаптера предоставляются стандартные реализации других методов из интерфейса MouseListener. Следует также заметить, что приемник событий от мыши вводится во фрейм, хранящийся в переменной jfrm. Это означает, что если щелкнуть правой кнопкой мыши в любом месте панели содержимого, то произойдет запуск всплывающего меню.

В методах mousePressed() и mouseReleased() вызывается метод isPopupTrigger(), чтобы выяснить, связано ли наступившее событие с запуском всплывающего меню. Если это действительно так, то вызывается метод show(), чтобы отобразить всплывающее меню. С целью получить вызывающий компонент используется метод getComponent() для события от мыши. В данном случае вызывающим компонентом будет панель содержимого. А для получения координат левого верхнего угла всплывающего меню вызываются методы getX() и getY(). В итоге левый верхний угол всплывающего меню оказывается прямо под курсором. И, наконец, в данную программу необходимо ввести приведенные ниже приемники действий. Они обрабатывают события действия, наступающие в тот момент, когда пользователь выбирает пункт из всплывающего меню.

```
jmiCut.addActionListener(this);
jmiCopy.addActionListener(this);
jmiPaste.addActionListener(this);
```

После внесения описанных выше дополнений в программу MenuDemo всплывающее меню можно активизировать щелчком правой кнопкой мыши в любом месте панели содержимого.

В отношении рассматриваемого здесь примера всплывающего меню следует также заметить, что компонентом, вызывающим всплывающее меню, в данном случае всегда остается фрейм jfrm, и поэтому его можно передать явным образом, не вызывая метод getComponent(). С этой целью фрейм jfrm следует присвоить переменной экземпляра класса MenuDemo, а не локальной переменной, чтобы сделать его доступным во внутреннем классе. И тогда для отображения всплывающего меню метод show() можно вызвать следующим образом:

```
jpu.show(jfrm, me.getX(), me.getY());
```

И хотя такой прием вполне пригоден в данном примере, преимущество вызова метода getComponent() заключается в том, что всплывающее меню будет автоматически появляться относительно вызывающего компонента. Следовательно, один и тот же код может быть использован для отображения любого всплывающего меню относительно вызывающего его объекта.

Диалоговые окна

До сих пор рассматривались только компоненты пользовательского интерфейса, которые находились в окне фрейма, создаваемого в приложении. Но при разработке приложений возникает потребность в отдельных всплывающих диалоговых окнах, которые должны появляться на экране и обеспечивать обмен данными с пользователем. Как и в большинстве оконных систем, в AWT различаются модальные и немодальные диалоговые окна. Модальные диалоговые окна не дают пользователю возможности одновременно работать с другими окнами приложения. Такие окна требуются в том случае, если пользователь должен ввести данные, от которых зависит дальнейшая работа приложения. Например, при вводе файла пользователь должен сначала указать его имя. И только после того, как пользователь закроет модальное диалоговое окно, приложение начнет ввод файла. Немодальное диалоговое окно дает пользователю возможность одновременно вводить данные, как в этом окне, так и в других окнах приложения. Примером такого окна служит панель инструментов. Она постоянно находится на своем месте, и пользователь может одновременно взаимодействовать как с ней, так и с другими окнами.

Сначала отобразим простейшее модальное диалоговое окно, содержащее единственную строку сообщения. В библиотеке Swing имеется удобный класс JOptionPane, позволяющий выводить на экран модальное диалоговое окно, не прибегая к написанию специального кода для его поддержки.

Диалоговые окна для выбора разных вариантов

В библиотеку Swing входит много готовых простых диалоговых окон, которые позволяют вводить данные отдельными фрагментами. Для этой цели в классе JOptionPane имеются перечисленные ниже статические методы.

showMessageDialog()	Выводит на экран сообщение и ожидает до тех пор, пока пользователь не щелкнет на кнопке OK
showConfirmDialog()	Выводит на экран сообщение и ожидает от пользователя подтверждения (щелчок на

	кнопке OK или Cancel)
showOptionDialog()	Выводит на экран сообщение и предоставляет пользователю возможность выбора среди нескольких вариантов
showInputDialog()	Выводит на экран сообщение и поле, в котором пользователь должен ввести данные

Диалоговые окна

Диалоговое окно может содержать дополнительный компонент для ввода данных. Этим компонентом может быть текстовое поле, в котором пользователь вводит произвольную символьную строку, или комбинированный список, один из элементов которого пользователь должен выбрать. Компоновка подобных диалоговых окон и выбор пиктограмм для стандартных сообщений зависит от визуального стиля оформления GUI. Пиктограмма в левой части диалогового окна выбирается в зависимости от типа сообщения. Существуют пять типов сообщений:

```
ERROR_MESSAGE
INFORMATION_MESSAGE
WARNING_MESSAGE
QUESTION_MESSAGE
PLAIN_MESSAGE
```

Для сообщения типа PLAIN_MESSAGE пиктограмма не предусмотрена. А для каждого типа диалоговых окон существует метод, позволяющий указывать свою собственную пиктограмму. С каждым типом диалоговых окон можно связать определенное сообщение, которое может быть представлено символьной строкой, пиктограммой, компонентом пользовательского интерфейса или любым другим объектом. Ниже поясняется, каким образом отображается объект сообщений.

String	Выводит символьную строку
Icon	Отображает пиктограмму
Component	Отображает компонент
Object[]	Выводит все объекты из массива, отображая их один над другим
Любой другой объект	Вызывает метод toString() и выводит получаемую в итоге символьную строку

Безусловно, на экран чаще всего выводится символьная строка сообщения. В то же время возможность отображать в диалоговом окне объекты типа Component дает немало удобств, поскольку, вызвав метод paintComponent(), можно нарисовать все, что угодно. Внешний вид кнопок, расположенных в нижней части диалогового окна, зависит от его типа, а также от типа вариантов. При вызове метода showMessageDialog() или showInputDialog() выбор ограничивается только стандартным набором экранных кнопок (OK или OK и Cancel). А вызывая метод showConfirmDialog(), можно выбрать один из четырех типов вариантов:

```
DEFAULT_OPTION
YES_NO_OPTION
YES_NO_CANCEL_OPTION
OK_CANCEL_OPTION
```

С помощью метода showOptionDialog() можно указать произвольный набор вариантов, задав массив объектов, соответствующих каждому из них. Элементы этого массива отображаются на экране описанным ниже способом.

String	Создает кнопку, меткой которой служит указанная символьная строка
Icon	Создает кнопку, меткой которой служит указанная пиктограмма
Component	Отображает компонент
Любой другой объект	Вызывает метод toString() и создает кнопку, меткой которой служит получаемая в итоге символьная строка

Статические методы, предназначенные для создания диалоговых окон, возвращают перечисленные ниже значения.

showMessageDialog()	Возвращаемое значение отсутствует
showConfirmDialog()	Целое значение, соответствующее выбранному варианту
showOptionDialog()	Целое значение, соответствующее выбранному варианту
showInputDialog()	Символьная строка, введенная или выбранная пользователем

Методы showConfirmDialog() и showOptionDialog() возвращают целое значение, обозначающее кнопку, на которой щелкнул пользователь. В диалоговом окне для выбора разных вариантов это числовое значение является порядковым номером. Если вместо выбора варианта пользователь закрыл диалоговое окно, возвращается константа CLOSED_OPTION. Ниже приведены константы, используемые в качестве возвращаемых значений.

```
OK_OPTION
CANCEL_OPTION
YES_OPTION
NO_OPTION
CLOSED_OPTION
```

Несмотря на обилие упомянутых выше мнемонических обозначений разных вариантов выбора, создать диалоговое окно данного типа совсем не трудно. Для этого выполните следующие действия.

1. Выберите тип диалогового окна (для вывода сообщения, получения подтверждения, выбора разных вариантов или ввода данных).

2. Выберите пиктограмму (с ошибкой, важной информацией, предупреждением, вопросом, свою собственную) или вообще откажитесь от нее.

3. Выберите сообщение (в виде символьной строки, пиктограммы, пользовательского компонента или массива компонентов).

4. Если вы выбрали диалоговое окно для подтверждения выбора, задайте тип вариантов (по умолчанию Yes/No, No/Cancel или OK/Cancel).

5. Если вы создаете диалоговое окно для выбора разных вариантов, задайте варианты выбора (в виде символьных строк, пиктограмм или собственных компонентов), а также вариант, выбираемый по умолчанию.

6. Если вы создаете диалоговое окно для ввода данных, выберите текстовое поле или комбинированный список.

7. Найдите подходящий метод в классе JOptionPane.

Допустим, на экране требуется отобразить диалоговое окно. В этом окне выводится сообщение, и пользователю предлагается подтвердить или отклонить его. Следовательно, это диалоговое окно для подтверждения выбора. Пиктограмма, отображаемая в этом окне, относится к разряду вопросов, а сообщение выводится символьной строкой. Тип вариантов выбора обозначается константой OK_CANCEL_OPTION. Для создания такого диалогового окна служит следующий фрагмент кода:

```
int selection = JOptionPane.showConfirmDialog(parent,
                  "Message", "Title",
                  JOptionPane.OK_CANCEL_OPTION,
                  JOptionPane.QUESTION_MESSAGE);
if (selection == JOptionPane.OK_OPTION) ...
```

Создание диалоговых окон

Как упоминалось ранее, для применения предопределенных диалоговых окон служит класс JOptionPane. Ниже будет показано, как создать диалоговое окно самостоятельно. Типичное модальное диалоговое окно, содержащее сообщение обычно появляется на экране после того, как пользователь выберет пункт меню About (О программе).

Чтобы реализовать такое окно, необходимо создать подкласс, производный от класса JDialog. По существу, этот процесс ничем не отличается от создания главного окна приложения расширением класса JFrame. А точнее говоря, для этого нужно сделать следующее.

1. Вызовите конструктор суперкласса JDialog в конструкторе вашего диалогового окна.

2. Введите в свое диалоговое окно компоненты пользовательского интерфейса.

3. Введите обработчики событий, наступающих в данном окне.

4. Установите размеры своего диалогового окна.

При вызове конструктора суперкласса следует указать фрейм-владелец, заголовок окна и признак модальности. Фрейм-владелец управляет местом отображения диалогового окна. Вместо владельца можно указать пустое значение null, и тогда диалоговое окно будет принадлежать скрытому фрейму. Признак модальности означает, должны ли блокироваться другие окна приложения до тех пор, пока отображается данное диалоговое окно.

```
public class AboutDialog extends JDialog {
      public AboutDialog(JFrame owner) {
            super(owner, "About DialogTest", true);
            add(new JLabel("<html><hl><i>Сообщение</i></hl></html>"),
            BorderLayout.CENTER);
            JButton ok = new JButton("OK");
            ok.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent event)
                  {
                        setVisible(false);
                  }});
            JPanel panel = new JPanel();
            panel.add(ok);
            add(panel, BorderLayout.SOUTH);
            setSize (250, 150);
      }
}
```

Конструктор диалогового окна вводит в него элементы пользовательского интерфейса, в данном случае метки и кнопку. Кроме того, он вводит обработчик событий от кнопки и задает размеры окна. Чтобы отобразить диалоговое окно, необходимо создать новый объект типа JDialog и вызвать метод setVisible() следующим образом:

```
JDialog dialog = new AboutDialog(this);
dialog.setVisible(true);
```

На самом деле в программе, рассматриваемой здесь в качестве примера, диалоговое окно создается только один раз, а затем оно используется повторно всякий раз, когда пользователь щелкает на кнопке About:

```
if (dialog == null) //в первый раз
      dialog = new AboutDialog(DialogFrame.this);
dialog.setVisible(true);
```

Когда пользователь щелкает на кнопке ОК, диалоговое окно должно закрываться. Такая реакция на действия пользователя определяется в обработчике событий от кнопки ОК, как следует из приведенной ниже строки кода.

```
ok.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent event)
      {
            setVisible(false);
      }
});
```

Когда же пользователь закрывает диалоговое окно, щелкая на кнопке Close, оно исчезает из виду. Как и в классе JFrame, такое поведение можно изменить с помощью метода setDefaultCloseOperation(). В листинге приведен исходный код класса фрейма для примера программы, где демонстрируется применение модального диалогового окна, создаваемою самостоятельно. А в листинге представлен исходный код класса для создания этого диалогового окна.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class DialogTest {
      public static void main(String[] args)
      {
            DialogFrame frame = new DialogFrame();
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
            frame.setVisible(true);
      }
}

public class DialogFrame extends JFrame {
      public DialogFrame() {
            setTitle("DialogTest");
            setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

            //Создание меню File
            JMenuBar menuBar = new JMenuBar();
            setJMenuBar(menuBar);
            JMenu fileMenu = new JMenu("File");
            menuBar.add(fileMenu);

            //добавление пунктов меню About и Exit
            //при выборе пункта About отображается
            //диалоговое окно About
            JMenuItem aboutItem = new JMenuItem("About");
            aboutItem.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent event)
                  {
                        if (dialog == null) //первый раз
                              dialog = new AboutDialog(DialogFrame.this);
                        dialog.setVisible(true);
                  }
            });
            fileMenu.add(aboutItem);

            //при активизации пункта Exit программа завершается
            JMenuItem exitItem = new JMenuItem("Exit");
            exitItem.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent event)
                  {
                        System.exit(0);
                  }
            });
            fileMenu.add(exitItem);
      }

      public static final int DEFAULT_WIDTH = 300;
      public static final int DEFAULT_HEIGHT = 200;
      private AboutDialog dialog;
}

public class AboutDialog extends JDialog {
      public AboutDialog(JFrame owner) {
            super(owner, "About DialogTest", true);

            //метка, содержащая HTML-форматирование, выравнивается
            //по центру
            add(new JLabel("<html><hl><i>Сообщение</i></hl></html>"),
            BorderLayout.CENTER);

            //при активизации кнопки OK диалоговое окно закрывается
            JButton ok = new JButton("OK");
            ok.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent event)
                  {
                        setVisible(false);
                  }
            });

            JPanel panel = new JPanel();
            panel.add(ok);
            add(panel, BorderLayout.SOUTH);
            setSize (250, 150);
      }
}
```