Графический интерфейс на Java Swing

В Java есть 3 основных пакета для создания графических интерфейсов (Graphics User Interface).

Самая ранняя из них называется AWT(Abstract Windows Toolkit). Библиотека Swing разработана на базе AWT и заменяет большинство ее компонентов своими, спроектированными более тщательно и удобно. Третья, самая новая библиотека, называется SWT.

Рассмотрим основные элементы библиотеки Swing и создадим простой интерфейс (GUI) в качестве примера.

**Окно JFrame**

Каждая GUI-программа запускается в окне и по ходу работы может открывать несколько дополнительных окон. Для группировки компонент интерфейса используются контейнеры (Container). Для создания основного контейнера для приложения чаще всего используется контейнер JFrame (есть еще JWindows и JApplet).

Наследуя JFrame можно получить доступ ко множеству методов, например:

setSize(int width, int height) — устанавливает размеры окна. Если не задать размеры, окно будет иметь нулевую высоту независимо от того, что в нем находится и пользователю после запуска придется растягивать окно вручную. Размеры окна включают не только «рабочую» область, но и границы и строку заголовка.

Вместо setSize() можно использовать метод pack(), который автоматически подбирает требуемые размеры.

setDefaultCloseOperation(int operation) — позволяет указать действие, которое необходимо выполнить, когда пользователь закрывает окно нажатием на крестик. Обычно в программе есть одно или несколько окон при закрытии которых программа прекращает работу. Для того, чтобы запрограммировать это поведение, следует в качестве параметра operation передать константу EXIT\_ON\_CLOSE, описанную в классе JFrame.

setVisible(boolean visible) — когда окно создается, оно по умолчанию невидимо.

setBounds(x, y, w, h) - указывает координаты верхней левой вершины окна, а также его ширину и высоту.

setResizable(bool) - указывает, можно ли изменять размер окна.

setTitle(str) - устанавливает название окна.

Пример

import javax.swing.\*;

В файле SimpleWindow.java:

public class SimpleWindow extends JFrame {

SimpleWindow(){

super("Пробное окно");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setSize(250, 100);

}

}

В файле Program.java:

public class Program {

public static void main (String [] args) {

JFrame myWindow = new SimpleWindow();

myWindow.setVisible(true);

}

}

Задание 1.

Используя код из примера 1, опробовать работу методов setBounds(x, y, w, h), setResizable(bool), setTitle(str).

Напрямую в окне элементы управления обычно не размещают. Для этого служит панель содержимого, занимающая все пространство окна. Обратиться к этой панели можно методом getContentPane() класса JFrame.

С помощью метода add(Component component) можно добавить на нее любой элемент управления.

Задание 2.

Добавьте кнопку в панель содержимого окна командами:

JButton newButton = **new** JButton(“OK”);

getContentPane().add(newButton);

Для размещения произвольного числа элементов управления используется класс Jpanel.

Панель JPanel — это элемент управления, представляющий собой прямоугольное пространство, на котором можно размещать другие элементы. Элементы добавляются и удаляются методами, унаследованными от класса Container.

У каждой панели есть так называемый менеджер размещения(LayoutManager), который определяет стратегию взаимного расположения элементов, добавляемых на панель.

FlowLayout - используется для последовательного отображения элементов. Если элемент не помещается в конкретную строку, он отображается в следующей.

GridLayout - отображения элементов в виде таблицы с одинаковыми размерами ячеек.

BorderLayout - используется при отображении не более 5 элементов. Эти элементы располагаются по краям фрейма и в ценрте: North, South, East, West, Center.

BoxLayout - отображает элементы в виде рядка или колонки.

GridBagLayout - позволяет назначать месторасположение и размер каждого виджета.

**Менеджер FlowLayout**

Класс FlowLayout реализует простой стиль размещения, при котором компоненты располагаются, начиная с левого верхнего угла, слева направо и сверху вниз. Если в данную строку не помещается очередной компонент, он располагается в левой позиции новой строки. Справа, слева, сверху и снизу компоненты отделяются друг от друга небольшими промежутками. Ширину этого промежутка можно задать в конструкторе FlowLayout. Каждая строка с компонентами выравнивается по левому или правому краю, либо центрируется в зависимости от того, какая из констант:

*FlowLayout.LEFT, FlowLayout.RIGHT* или *FlowLayout.CENTER* была передана конструктору. Режим выравнивания по умолчанию - CENTER, используемая по умолчанию ширина промежутка - 5 пикселей.

***Конструкторы***:

FlowLayout();

FlowLayout(int align);

FlowLayout(int align, int hgap, int vgap);

Пример 2.

JPanel panel = **new** JPanel();

panel.setLayout(**new** FlowLayout());

panel.add(**new** JButton("Кнопка"));

panel.add(**new** JButton("+"));

panel.add(**new** JButton("-"));

panel.add(**new** JButton("Кнопка с длинной надписью"));

setContentPane(panel);setSize(250, 100);

Задание 3. Используя пример 3, опробовать работу разных конструкторов менеджера FlowLayout.

**Менеджер BorderLayout**

Менеджер размещения BorderLayout разделяет панель на пять областей: центральную, верхнюю, нижнюю, правую и левую. В каждую из этих областей можно добавить ровно по одному компоненту, причем компонент будет занимать всю отведенную для него область. Компоненты, добавленные в верхнюю и нижнюю области, будут растянуты по ширине, добавленные в правую и левую — по высоте, а компонент, добавленный в центр, будет растянут так, чтобы полностью заполнить оставшееся пространство панели.

При добавлении элемента на панель с менеджером размещения BorderLayout, необходимо указывать в методе add(), какая из областей имеется в виду. Для этого используются константы, определенные в классе BorderLayout: NORTH, SOUTH, EAST, WEST и CENTER.

Пример 3

SimpleWindow(){

super("Пробное окно");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

getContentPane().add(new JButton("Кнопка"), BorderLayout.NORTH);

getContentPane().add(new JButton("+"), BorderLayout.EAST);

getContentPane().add(new JButton("-"), BorderLayout.WEST);

getContentPane().add(new JButton("Кнопка с длинной надписью"), BorderLayout.SOUTH);

getContentPane().add(new JButton("В ЦЕНТР!"));

setSize(250, 100);

}

В центре может быть размещенна другая панель с произвольным количеством и порядком расположения элементов.

**Задание 4.**

Разместите в центре панель из задания 3.

**Менеджер GridLayout**

Класс GridLayout размещает компоненты в простой равномерной сетке. Конструктор этого класса позволяет задавать количество строк и столбцов. Например, setLayout(new GridLayout(5,4)) – указывает желаемое количество строк (5) и столбцов (4). А setLayout(new GridLayout(5,4,3,3)) – указывает желаемое количество строк (5) и столбцов (4) и расстояние между компонентами по горизонтали (3) и вертикали (3).

Пример 4.

SimpleWindow(){super("Пробное окно");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JPanel panel = **new** JPanel();

panel.setLayout(**new** GridLayout(2,3,5,10));

panel.add(**new** JButton("Кнопка"));

panel.add(**new** JButton("+"));

panel.add(**new** JButton("-"));

panel.add(**new** JButton("Кнопка с длинной надписью"));

panel.add(**new** JButton("еще кнопка"));

setContentPane(panel);setSize(250, 100);}

**Задание 5.**

Выполните код из примера 4 с разными значениями параметров конструктора GridLayout.

**Менеджер BoxLayout**

Менеджер BoxLayout размещает элементы на панели в строку или в столбец.

Обычно для работы с этим менеджером используют вспомогательный класс Box, представляющий собой панель, для которой уже настроено блочное размещение. Создается такая панель не конструктором, а одним из двух статических методов, определенных в классе Box: createHorizontalBox() и createVerticalBox().

Элементы, добавленные на панель с блочным размещением, выстраиваются один за другим. Расстояние между элементами по умолчанию нулевое. Однако вместо компонента можно добавить невидимую «распорку», единственная задача которой — раздвигать соседние элементы, обеспечивая между ними заданное расстояние. Горизонтальная распорка создается статическим методом createHorizontalStrut(int width), а вертикальная — методомcreateVerticalStrut(int height). Оба метода определены в классе Box, а целочисленный параметр в каждом из них определяет размер распорки.

Кроме того, на такую панель можно добавить еще один специальный элемент — своеобразную «пружину». Если размер панели будет больше, чем необходимо для оптимального размещения всех элементов, те из них, которые способны растягиваться, будут стараться заполнить дополнительное пространство собой. Если же разместить среди элементов одну или несколько «пружин», дополнительное свободное пространство будет распределяться и в эти промежутки между элементами. Горизонтальная и вертикальная пружины создаются соответственно методами createHorizontalGlue() иcreateVerticalGlue().

Пример 5.

SimpleWindow(){super("Пробное окно");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

Box box = Box.createVerticalBox();

box.add(**new** JButton("Кнопка"));

box.add(Box.createVerticalStrut(10));

box.add(**new** JButton("+"));

box.add(Box.createVerticalGlue());

box.add(**new** JButton("-"));

box.add(Box.createVerticalStrut(10));

box.add(**new** JButton("Кнопка с длинной надписью"));

setContentPane(box);setSize(250, 100);}

**Задание 6.**

Выполните код из примера 5 с разными вариантами расположения и значений распорок и пружин.

**Ручное размещение элементов**

Если в качестве менеджера размещения панели установить **null**, элементы не будут расставляться автоматически. Координаты каждого элемента необходимо в этом случае указать явно, при этом они никак не зависят от размеров панели и от координат других элементов. По умолчанию координаты равны нулю (т.е. элемент расположен в левом верхнем углу панели). Размер элемента также необходимо задавать явно (в противном случае его ширина и высота будут равны нулю и элемент отображаться не будет).

Координаты элемента можно задать одним из следующих методов:

setLocation(int x, int y),

setLocation(Point point)

Эти методы работают аналогично, устанавливая левый верхний угол элемента в точку с заданными координатами. Разница в способе задания точки. Можно представить точку двумя целыми числами, а можно объектом класса Point. Класс Point по сути представляет собой ту же пару чисел, его конструктор имеет вид Point(int x, int y). Получить доступ к отдельной координате можно методами getX() и getY().

Размер элемента задается одним из двух методов:

setSize(int width, int height),

setSize(Dimension size)

Пример 6.

Создадим панель, с которой не будет связано никакого менеджера размещения и вручную разместим на ней две кнопки:

SimpleWindow(){

super("Пробное окно");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JPanel panel = **new** JPanel();

panel.setLayout(**null**);

JButton button = **new** JButton("Кнопка");

button.setSize(80, 30);

button.setLocation(20,20);

panel.add(button);

button = **new** JButton("Кнопка с длинной надписью");

button.setSize(120, 40);

button.setLocation(70,50);

panel.add(button);

setContentPane(panel);setSize(250, 150);}

Задание 7.

Выполните код примера 6.

**Основные визуальные компоненты Swing**

**Класс JComponent**

Все визуальные компоненты библиотеки Swing унаследованы от класса JComponent. Сам этот класс является абстрактными и непосредственно не используется, но все визуальные компоненты наследуют его методы. Рассмотрим наиболее полезные из них.

setEnabled(boolean enabled) используется для управления активностью компонента. При вызове этого метода с параметром **false** компонент переходит в неактивное состояние. Метод isEnabled() возвращает **true**, если элемент активен и **false** в противном случае.

setVisible(boolean visible) управляет видимостью компонента. Метод isVisible() возвращает **false**, если элемент невидим и true в противном случае.

С помощью метода setBackground(Color color) можно изменить цвет фона компонента.

**Метка JLabel**

Метка представляет собой обычный текст, который выводится в заданном месте окна и используется для вывода вспомогательной текстовой информации: подписи к другим элементам, инструкции и предупреждения для пользователя. В Swing помимо текста можно использовать значок.

Конструкторы:

JLabel(String text) — создает метку с надписью text

JLabel(Icon image) — создает метку со значком image

JLabel(String text, Icon image, int align) — создает метку с надписью text и значком image. Третий параметр задает выравнивание текста вместе со значком. В качестве него может быть использована одна из констант, описанных в интерфейсе SwingConstants: LEFT, RIGHT, CENTER.

Пример 7.

SimpleWindow(){

**super**("Окно с надписью");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JLabel label = **new** JLabel("Метка со значком и с надписью", **new** ImageIcon("1.gif"), SwingConstants.RIGHT);

getContentPane().add(label);

pack();}

Основные методы класса JLabel:

getText() — возвращает текущий текст надписи метки;

setText(String text) — задает новый текст надписи.

Существует и ряд других методов управления внешним видом кнопки.

Задание 8.

Выполните код примера 7, используя разные конструкторы метки.

**Кнопка JButton**

Кнопка создается одним из конструкторов:

JButton(),

JButton(String text),

JButton(Icon icon),

JButton(String text, Icon icon), параметры которых говорят сами за себя.

Все методы класса JLabel, описанные в предыдущем разделе, присутствуют и в классе JButton. С помощью этих методов можно изменять значок и текст надписи на кнопке, а также управлять их взаимным расположением друг относительно друга и относительно края кнопки (с учетом отступов). Существует и ряд методов, изменяющих её внешний вид при действиях с кнопкой.

Пример 8.

Для примера создадим кнопку со значком и с надписью, изменим ее отступы и расположение текста относительно значка (текст будет выровнен влево и вверх относительно значка).

SimpleWindow(){

**super**("Окно с кнопкой");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JButton button = **new** JButton("Кнопка", **new** ImageIcon("1.gif"));

button.setMargin(**new** Insets(0, 10, 20, 30));

button.setVerticalTextPosition(SwingConstants.TOP);

button.setHorizontalTextPosition(SwingConstants.LEFT);

getContentPane().add(button);

pack();}

Задание 9.

Выполните код примера 8, с различными параметрами методов кнопки.

**Компоненты JToggleButton, JCheckBox, JRadioButton**

Компонент JToggleButton представляет собой кнопку, которая может находиться в двух состояниях: нажатом и отпущенном.

Основной конструктор — JToggleButton(String text, Icon icon, boolean selected) создает кнопку с заданными надписью, значком и текущим состоянием. Кнопку можно перевести в требуемое состояние программным путем, вызвав метод setSelected(boolean selected). Метод isSelected() возвращает **true**, если кнопка выбрана (т.е. находится в нажатом состоянии) и **false** в противном случает.

От класса JToggleButton унаследован класс JCheckBox — флажок. Этот класс имеет точно такой же набор конструкторов и методов, т.е. не расширяет функциональность предка. Единственное различие между ними — во внешнем виде: JCheckBox выглядит не как кнопка, а как небольшой квадратик, в котором можно поставить или убрать галочку.

Аналогичным образом ведет себя класс JRadioButton — переключатель или радиокнопка, внешне выглядящая как пустой кружок, когда она не выделена и кружок с точкой в выделенном состоянии.

Несмотря на то, что классы JCheckBox и JRadioButton ведут себя абсолютно одинаково (и аналогично их общему предкуJToggleButton), их принято использовать в различных ситуациях. В частности, JRadioButton предполагает выбор единственной альтернативы из нескольких возможных: несколько таких объектов объединяются в одну группу (чаще всего эта группа визуально обозначается рамкой) и при выборе одного из элементов группы предыдущий выбранный элемент переходит в состояние «не выбран».

Для того, чтобы получить такое поведение, используется специальный контейнер ButtonGroup — взаимоисключающая группа (создается конструктором без параметров). Если добавить в один такой контейнер несколько элементовJRadioButton, то выбранным всегда будет только один из них.

Рассмотри пример, в котором создаются две кнопки выбора, два флажка и два переключателя. Кнопки выбора и переключатели объединены в группы ButtonGroup. Для того, чтобы обвести каждую пару элементов рамкой, необходимо расположить каждую пару элементов на отдельной панели.

Пример 9.

import javax.swing.\*;

import javax.swing.border.TitledBorder;

import java.awt.\*;

SimpleWindow(){super("Пример с кнопками выбора, флажками и переключателями");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

ImageIcon icon = new ImageIcon("1.gif");

Box mainBox = Box.createVerticalBox();

Box box1 = Box.createVerticalBox();

JToggleButton tButton1 = new JToggleButton("Кнопка выбора 1");

JToggleButton tButton2 = new JToggleButton("Кнопка выбора 2", icon);

tButton2.setSelectedIcon(new ImageIcon("2.gif"));

box1.add(tButton1);box1.add(tButton2);

box1.setBorder(new TitledBorder("Кнопки выбора"));

Box box2 = Box.createVerticalBox();

JCheckBox check1 = new JCheckBox("Флажок 1");

JCheckBox check2 = new JCheckBox("Флажок 2");

box2.add(check1);

box2.add(check2); // добавили флажки на панель box2

box2.setBorder(new TitledBorder("Флажки"));

Box box3 = Box.createVerticalBox();

JRadioButton rButton1 = new JRadioButton("Переключатель 1");

JRadioButton rButton2 = new JRadioButton("Переключатель 2");

ButtonGroup bg = new ButtonGroup();

bg.add(rButton1);

bg.add(rButton2); // сделали радиокнопки взаимоисключающими

box3.add(rButton1);

box3.add(rButton2); // добавили радиокнопки на панель box3

box3.setBorder(new TitledBorder("Переключатели"));

mainBox.add(box1);

mainBox.add(box2);

mainBox.add(box3);

setContentPane(mainBox);

pack();}

Задание 10.

Выполните код примера 9, с различными параметрами методов флажка и переключателей.

Примечание. Файлы 1.gif и 2.gif должны находиться : в директории вашего проекта.

**Текстовое поле JTextField**

*Текстовое поле* — простой и часто используемый компонент, предназначенный для ввода небольших по объему (записываемых в одну строку) текстовых данных. Для создания текстового поля чаще всего используются конструкторы:

JTextField(int columns) — создает пустое текстовое поле, ширина которого достаточна для размещения columns символов. При этом пользователь может вводить в текстовое поле строку какой угодно длины: она просто будет прокручиваться.

JTextField(String text) — создает текстовое поле с начальным текстом text.

JTextField(String text, int columns) — устанавливает и ширину и начальный текст.

Занести текст в поле можно методом setText(String text). Метод getText() возвращает содержимое текстового поля целиком.

**Поле для ввода пароля JPasswordField**

JPasswordField является прямым потомком JTextField, поэтому для него справедливо все сказанное выше. Отличие заключается в том, что весь введенный в него текст скрыт от посторонних глаз: он заменяется звездочками или другим символом, установить который позволяет метод setEchoChar(char echo), а получить — getEchoChar().

Чаще всего JPasswordField применяется для ввода пароля. Метод getText() позволяет получить этот пароль, но пользоваться им не рекомендуется (злоумышленник может проанализировать содержимое оперативной памяти и перехватить этот пароль). Вместо него следует использовать метод getPassword(), возвращающий массив символовchar[].

**Область для ввода текста JTextArea**

JTextArea также является потомком JTextField и наследует все его методы. В отличие от текстового поля область для ввода текста позволяет ввести не одну строку, а несколько. При создании JTextArea чаще всего используют конструктор JTextArea(int rows, int columns), устанавливающий высоту (количество строк) и ширину (количество символов) компонента.

В связи с этим JTextArea предлагает несколько дополнительных функций. Во-первых, это способность переносить слова на соседнюю строку целиком, которой управляет метод setWrapStyleWord(boolean wrapStyle). Если вызвать этот метод с параметром true, то слова не будут разрываться в том месте, где они «натыкаются» на границу компонента, а будут целиком перенесены на новую строку. Во-вторых, это способность переносить текст (то есть длинные строки будут укладываться в несколько строк вместо одной, уходящей за границы компонента. Этой способностью управляет метод setLineWrap(boolean lineWrap). Методы isWrapStyleWord() и isLineWrap() возвращают текущее состояние данных способностей (true — активирована и false — деактивирована).

Для работы со своим содержимым JTextArea дополнительно предлагает два удобных метода. Метод append(String text) добавляет строку text в конец уже имеющегося текста, а метод insert(String text, int position) вставляет ее в позицию position.

Пример 10.

SimpleWindow(){**super**("Пример текстовых компонентов");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JTextField textField = **new** JTextField("Текстовое поле", 20);

textField.setCaretColor(Color.RED);

textField.setHorizontalAlignment(JTextField.RIGHT);

JPasswordField passwordField = **new** JPasswordField(20);

passwordField.setEchoChar('$');

passwordField.setText("пароль");

JTextArea textArea = **new** JTextArea(5, 20);

textArea.setLineWrap(true);textArea.setWrapStyleWord(true);

**for** (**int** i = 0; i <= 20; i++)

textArea.append("Область для ввода текстового содержимого ");

getContentPane().add(textField, BorderLayout.NORTH);

getContentPane().add(textArea);

getContentPane().add(passwordField, BorderLayout.SOUTH);pack();}

Задание 11.

Выполните код примера 10, с различными параметрами методов текстовых полей.

**Панель прокрутки JScrollPane**

Наблюдая за поведением компонента JTextArea в предыдущем примере, легко можно обнаружить проблемы, которые возникают, когда тексту становится «тесно» в рамках отведенного места. В зависимости от используемого менеджера расположения текст либо обрезается, уходя за границы компонента, либо раздвигает эти границы (но в любом случае остается ограничен размером окна). В таких случаях типично использование полос прокрутки, но в Swing полосы прокрутки сами собой не появляются.

Добавить к компоненту полосы прокрутки на самом деле очень просто. Для этого служит компонентJScrollPane — панель прокрутки. Чаще всего она просто «надевается» на требуемый объект посредством собственного конструктора, принимающего этот объект в качестве параметра. Например, чтобы текстовая область textArea из предыдущего примера обрела полосы прокрутки, необходимо заменить команду

getContentPane().add(textArea);

на команду

getContentPane().add(**new** JScrollPane(textArea));

**Задание 12.**

Вставьте полосу прокрутки в textArea из предыдущего примера.

**Инструментальная панель JToolBar**

Большинство программных продуктов предоставляют удобные инструментальные панели, расположенные вдоль границ окна программы и содержащие кнопки, выпадающие списки и другие элементы управления, обычно соответствующие командам меню. В Swing для инструментальных панелей разработан визуальный компонент JToolBar.

Пример 11.

Создадим окно с менеджером расположенияBorderLayout, разместим по центру область для ввода текста JTextArea, а к верхней границе прикрепим инструментальную панель с тремя кнопками и одним разделителем:

SimpleWindow(){super("Пример использования JToolBar");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JTextArea textArea = new JTextArea(5, 20);

getContentPane().add(textArea);

JToolBar toolBar = new JToolBar("Инструментальная панель");

toolBar.add(new JButton("Кнопка 1"));

toolBar.add(new JButton("Кнопка 2"));

toolBar.addSeparator();

toolBar.add(new JButton("Кнопка 3"));

getContentPane().add(toolBar, BorderLayout.NORTH);pack();}

**Задание 13.**

Запустите пример 11 и поэкспериментируйте с инструментальной панелью. Попробуйте отсоединить ее от верхней границы окна и прикрепить к какой-либо другой. Отсоедините ее от границ окна так, чтобы панель стала самостоятельным окном. При этом панель всегда отображается над родительским окном, даже если именно оно, а не панель является активным. Если закрыть самостоятельную панель кнопкой с крестиком, она вернется в свое окно, в то место, где она была закреплена последний раз.

Конструктор JToolBar(String title) создает горизонтальную панель с заданным заголовком. Горизонтальная панель предназначена для прикрепления к верхней либо нижней границе родительской панели (имеющей расположение BorderLayout). Для создания вертикальной панели используется конструктор JToolBar(String title, int orientation), где параметр orientation задается константой VERTICAL из интерфейса SwingConstants.

add(Component component) — добавляет на инструментальную панель новый элемент управления. Взаимосвязанные группы элементов управления принято разделять с помощью линии или пустого пространства. Метод addSeparator()добавляет такой разделитель.

**Выпадающий список JComboBox**

Выпадающий список — весьма распространенный элемент управления. Он содержит множество вариантов, из которых пользователь может выбрать один и только один, либо (если выпадающий список это позволяет) ввести свой собственный.

Создать выпадающий список можно конструктором по умолчанию JComboBox(), после чего добавлять в него элементы методом addItem(Object item), добавляющим новый элемент в конец списка, или методом insertItemAt(Object item, int index), позволяющим уточнить позицию, в которую требуется вставить элемент. Однако проще использовать конструктор, в котором сразу указываются все элементы выпадающего списка JComboBox(Object[] elements)

Метод getItemAt(int index) позволяет обратиться к произвольному элементу.

Метод removeAllItems() удаляет из JComboBox все элементы, а методremoveItem(Object item) — конкретный элемент (при условии, что он содержался в списке).

Метод getSelectedIndex() позволяет получить индекс выбранного пользователем элемента (элементы нумеруются начиная с нуля), а метод getSelectedItem()возвращает сам выбранный объект. Сделать конкретный элемент выбранным можно и программно, воспользовавшись методом setSelectedIndex(int index) или setSelectedItem(Object item).

Пример 12.

Рассмотрим пример, в котором создается выпадающий список из 3 элементов и выбирается 2-й.

SimpleWindow(){super("Пример использования JComboBox");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

String[] elements = new String[] {"Вася", "Петя","<html><font size = +1 color = yellow>Иван</font>"};

JComboBox combo = new JComboBox(elements);

combo.setSelectedIndex(1);

JPanel panel = new JPanel();

panel.add(combo);

setContentPane(panel);setSize(200,200);}

**Задание 14.**

Опробуйте код примера 12. Добавьте в список новые элементы

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JSlider slider = new JSlider(JSlider.HORIZONTAL, 50, 150, 70);

slider.setMajorTickSpacing(20);slider.setMinorTickSpacing(5);

slider.setPaintTicks(true);slider.setPaintLabels(true);

slider.setSnapToTicks(true);JPanel panel = new JPanel();

panel.add(slider);setContentPane(panel);

pack();}

**Окно входа в систему**

В качестве реального примера законченного окна, иллюстрирующего положения этой и предыдущей глав, рассмотрим окно для входа в систему Это окно содержит два текстовых поля для ввода логина и пароля, подписи к этим полям и кнопки «OK» и «Отмена».

При этом:

1. Кнопки «ОК» и «Отмена» принято прижимать к правому краю окна, поэтому в начале третьей горизонтальной панели необходимо добавить «пружину».
2. Должно выдерживаться аккуратное расстояние между элементами. В частности, для стиля Java разработаны следующие рекомендации. Тесно связанные элементы (такие как текстовое поле и подпись к нему) должны отстоять друг от друга на 6 пикселов. Логически сгруппированные элементы — на 12 пикселов (в нашем случае это две верхние панели и пара кнопок). Все остальные элементы должны находиться на расстоянии 17 пикселов друг от друга. Не следует забывать и про рамку окна.
3. Элементы должны быть аккуратно выровнены. В нашем случае надписи, которые мы собираемся поместить перед текстовыми полями, наверняка будут разной длины и из-за этого поля будут сдвинуты относительно друг друга, что не желательно. Поэтому следует принудительно задать у надписей одинаковую ширину.
4. При увеличении размеров окна текстовые поля будут неэстетично изменять свою высоту. Можно ее зафиксировать, а можно просто запретить окну изменять свои размеры после упаковки командой setResizable(false).

Итоговый код окна приведен в примере 13.

Пример 13.

import javax.swing.\*;

import javax.swing.border.\*;

import java.awt.\*;

public class LoginWindow extends JFrame {

JTextField loginField;

JPasswordField passwordField;

LoginWindow(){super("Вход в систему");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

Box box1 = Box.createHorizontalBox();

JLabel loginLabel = new JLabel("Логин:");

loginField = new JTextField(15);

box1.add(loginLabel);

box1.add(Box.createHorizontalStrut(6));

box1.add(loginField);

Box box2 = Box.createHorizontalBox();

JLabel passwordLabel = new JLabel("Пароль:");

passwordField = new JPasswordField(15);

box2.add(passwordLabel);

box2.add(Box.createHorizontalStrut(6));

box2.add(passwordField);

Box box3 = Box.createHorizontalBox();

JButton ok = new JButton("OK");

JButton cancel = new JButton("Отмена");

box3.add(Box.createHorizontalGlue());

box3.add(ok);box3.add(Box.createHorizontalStrut(12));

box3.add(cancel);

loginLabel.setPreferredSize(passwordLabel.getPreferredSize());

Box mainBox = Box.createVerticalBox();

mainBox.setBorder(new EmptyBorder(12,12,12,12));

mainBox.add(box1);mainBox.add(Box.createVerticalStrut(12));

mainBox.add(box2);

mainBox.add(Box.createVerticalStrut(17));

mainBox.add(box3);

setContentPane(mainBox);

pack();

setResizable(false);}}

**Задание 15.**

Опробуйте код примера 13. Проведите эксперименты с элементами разметки

**Обработка событий в Swing**

Графический интерфейс пользователя включает в себя не только расположение в окне необходимых элементов управления, но и описание реакции на пользовательские события: выбор пользователем команд меню, нажатие кнопок, ввод нового символа в текстовое поле и др.

Таким образом, при разработке программы необходимо:

1. Выявить события, в ответ на которые потребуется реакция программы.
2. Написать код, реализующий эту реакцию (так называемый обработчик событий).
3. Связать обработчик события с соответствующим событием.

Первый пункт зависит от логики работы приложения и находится всецело на усмотрении программиста. Обработчик события — это, как правило, обычный метод и ничего особенного в его написании нет.

**Шаблон проектирования «наблюдатель»**

В библиотеке Swing для привязки обработчика события к вызывающему его компоненту используется шабло н(паттерн) проектирования «наблюдатель».

Шаблоны проектирования — это стандартные приемы объектно-ориентированного программирования.

Шаблон проектирования «наблюдатель» применяется, когда один объект должен оповещать другие о произошедших с ним изменениях или внешних воздействиях. Такой объект называется наблюдаемым, а объекты, которые следует оповестить — наблюдателями.

Для того, чтобы подобное взаимодействие было возможным, объект-наблюдатель должен иметь метод (или несколько методов) с заранее определенной сигнатурой (именем и параметрами). Когда с наблюдаемым объектом произойдет ожидаемое событие, он вызовет соответствующий метод у своего наблюдателя. В этом методе и будет запрограммирована реакция на событие.

Для того, чтобы наблюдаемый объект мог вызвать метод наблюдателя, он должен знать о том, что такой наблюдатель существует. Поэтому у наблюдаемого предварительно должен быть вызван специальный метод, регистрирующий его наблюдателя.

Заметим, что в данной схеме один наблюдатель может быть зарегистрирован у нескольких объектов (т.е. одинаково реагировать на изменения в каждом из них), а у одного объекта может быть несколько наблюдателей (т.е. при возникновении события выполняется несколько независимых методов-обработчиков). Это весьма увеличивает гибкость программирования.

**Механизм обработки событий библиотеки Swing**

В контексте графического интерфейса пользователя наблюдаемыми объектами являются элементы управления: кнопки, флажки, меню и т.д. Они могут сообщить своим наблюдателям об определенных событиях, как элементарных (наведение мышкой, нажатие клавиши на клавиатуре), так и о высокоуровневых (изменение текста в текстовом поле, выбор нового элемента в выпадающем списке и т.д.).

Наблюдателями должны являться объекты классов, поддерживающих специальные интерфейсы. Такие классы в терминологии Swing называются *слушателями*.

**Интерфейс MouseListener и обработка событий от мыши**

Рассмотрим технологию написания слушателей на примере слушателей событий мыши.

События от мыши — один из самых популярных типов событий. Практически любой элемент управления способен сообщить о том, что на него навели мышь, щелкнули по нему и т.д. Об этом будут оповещен все зарегистрированные слушатели событий от мыши.

Так например, кнопка для входа в систему из примера 13 должна реагировать на щелчок по ней, проверяя имя и пароль, введенные пользователем.

Слушатель событий от мыши должен реализовать интерфейс MouseListener. В этом интерфейсе перечислены следующие методы:

public void mouseClicked(MouseEvent event) — выполнен щелчок мышкой на наблюдаемом объекте

public void mouseEntered(MouseEvent event) — курсор мыши вошел в область наблюдаемого объекта

public void mouseExited(MouseEvent event) — курсор мыши вышел из области наблюдаемого объекта

public void mousePressed(MouseEvent event) — кнопка мыши нажата в момент, когда курсор находится над наблюдаемым объектом

public void mouseReleased(MouseEvent event) — кнопка мыши отпущена в момент, когда курсор находится над наблюдаемым объектом

Чтобы обработать нажатие на кнопку, требуется описать класс, реализующий интерфейс MouseListener, причем метод mouseClicked() должен содержать обработчик события. Далее необходимо создать объект этого класса и зарегистрировать его как слушателя интересующей нас кнопки.

Для регистрации слушателя используется метод addMouseListener(MouseListener listener).

Пример 14.

Опишем класс слушателя в пределах класса окна SimpleWindow, после конструктора. Обработчик события будет проверять, ввел ли пользователь логин «Иван» (пароль проверять не будем) и выводить сообщение об успехе или неуспехе входа в систему (работа стандартных диалоговых окон обсуждается далее):

**class** MouseL **implements** MouseListener {

**public void** mouseClicked(MouseEvent event) {

**if** (loginField.getText().equals("Иван"))JOptionPane.showMessageDialog(**null**, "Вход выполнен");

**else** JOptionPane.showMessageDialog(**null**, "Вход НЕ выполнен");}

**public void** mouseEntered(MouseEvent event) {}

**public void** mouseExited(MouseEvent event) {}

**public void** mousePressed(MouseEvent event) {}

**public void** mouseReleased(MouseEvent event) {}

}

Задание 16.

Добавьте класс слушателя из примера 14, создайте его объект и прикрепите его к кнопке. Для этого в код конструктора SimpleWindow() необходимо добавить команду:

ok.addMouseListener(**new** MouseL());

Это можно сделать сразу после команды:

JButton ok = **new** JButton("OK");

**Класс MouseAdapter**

Программа стала выглядеть загроможденной главным образом из-за того, что помимо полезного для нас метода mouseClicked() нам пришлось определять пустые реализации всех остальных, не нужных методов. В принципе, этого можно избежать.

Класс MouseAdapter реализует интерфейс MouseListener, определяя пустые реализации для каждого из его методов. Можно унаследовать своего слушателя от этого класса и переопределить те методы, которые нам нужны.

Пример15.

Описание слушателя из примера 14 можно сделать более компактным:

ok.addMouseListener(**new** MouseAdapter() {

**public void** mouseClicked(MouseEvent event)

{**if** (loginField.getText().equals("Иван"))

JOptionPane.showMessageDialog(**null**, "Вход выполнен");

**else** JOptionPane.showMessageDialog(**null**, "Вход НЕ выполнен");} });

Кроме слушателей MouseListener визуальные компоненты Swing поддерживают целый ряд других слушателей:

* Слушатель фокуса FocusListener
* Слушатель клавиатуры KeyListener
* Слушатель изменения состояния ChangeListener
* Слушатель событий окна WindowListener
* Слушатель событий компонента СomponentListener
* Слушатель выбора элемента ItemListener

**Универсальный слушатель ActionListener**

Среди многочисленных событий, на которые реагирует каждый элемент управления (и о которых он оповещает соответствующих слушателей, если они к нему присоединены), есть одно основное, вытекающее из самой сути компонента и обрабатываемое значительно чаще, чем другие. Например, для кнопки это щелчок на ней, а для выпадающего списка — выбор нового элемента.

Для отслеживания и обработки такого события может быть использован особый слушатель ActionListener, имеющий один метод:

public void actionPerformed(ActionEvent event).

Пример 16.

Обработка нажатия на кнопку ok из примера 14 легко переписывается для ActionListener:

ok.addActionListener(**new** ActionListener()

{ **public void** actionPerformed(ActionEvent event)

{**if** (loginField.getText().equals("Иван"))

JOptionPane.showMessageDialog(**null**, "Вход выполнен");

**else** JOptionPane.showMessageDialog(**null**, "Вход НЕ выполнен");} });

**Работа с меню в библиотеке Swing**

Неотъемлемой частью современных оконных программ является меню, представляющее собой удобно сгруппированный набор команд. Меню бывает двух типов: главное и контекстное. *Главное меню* располагается вдоль верхней границы окна и содержит команды, относящиеся ко всему приложению (точнее, все команды, которые можно выполнить, работая с данным окном). *Контекстное меню* вызывается нажатием правой кнопки мышки на конкретном объекте и содержит команды, которые могут быть применены именно к этому объекту.

**Создание главного меню**

Главное меню окна представлено в Swing классом JMenuBar. По сути своей этот класс представляет собой панель с менеджером расположения BoxLayout (по горизонтали), в которую можно добавлять не только элементы меню, но и что угодно: хоть выпадающие списки, хоть панели с закладками. Однако для удобства пользования программой предпочтительнее использовать «традиционные» возможности меню.

Главное меню должно быть присоединено к окну методом setJMenuBar(JMenuBar menuBar).

Элементами главного меню являются обычные меню — выпадающие прямоугольные блоки команд — объекты классаJMenu. Конструктор JMenu(String title) принимает один параметр: название меню, которое будет отображаться в строке главного меню.**\***

Меню, в свою очередь, состоит из *пунктов меню*, представленных классом JMenuItem. По логике работы пункты меню аналогичны кнопке JButton, то есть, при нажатии на него пользователем выполняется какое-то действие.

Создать элемент меню можно пустым конструктором JMenuItem либо одним из конструкторов, в которые передается текст и/или значок элемента меню: JMenuItem(String text), JMenuItem(Icon icon),JMenuItem(String text, Icon icon). Конечно, в любой момент текст и значок можно сменить методами setText(String text) и setIcon(Icon icon) соответственно.

Элемент добавляется в меню методом add(JMenuItem item) класса JMenu. Чтобы отделить группы взаимосвязанных элементов меню, можно добавить между ними разделитель методом addSeparator() класса JMenu.

Пример 17.

Создадим главное меню окна, состоящее из двух подменю: «Файл» и «Правка», причем в меню «Правка» поместим выпадающее подменю. Кроме того, воспользуемся знаниями о менеджере расположения главного меню, чтобы добавить с правого края значок (наподобие того, как это сделано в браузере Internet Explorer).

SimpleWindow() {

**super**("Окно с меню");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

JMenuBar menuBar = **new** JMenuBar();

JMenu fileMenu = **new** JMenu("Файл");

fileMenu.add(**new** JMenuItem("Новый"));

fileMenu.add(**new** JMenuItem("Открыть", **new** ImageIcon("1.gif")));

fileMenu.add(**new** JMenuItem("Сохранить"));

fileMenu.addSeparator();

fileMenu.add(**new** JMenuItem("Выйти"));

JMenu editMenu = **new** JMenu("Правка");

editMenu.add(**new** JMenuItem("Копировать"));

JMenu pasteMenu = **new** JMenu("Вставить");

pasteMenu.add(**new** JMenuItem("Из буфера"));

pasteMenu.add(**new** JMenuItem("Из файла"));

editMenu.add(pasteMenu);

menuBar.add(fileMenu);

menuBar.add(editMenu);

menuBar.add(Box.createHorizontalGlue());

menuBar.add(**new** JLabel(**new** ImageIcon("2.gif")));

setJMenuBar(menuBar);

setSize(250,150);}

Задание 17.

Опробовать код из примера 17.

**Обработка команд меню**

Как было сказано, каждый элемент меню по сути представляет собой кнопку, поэтому реагирует на те же события и позволяет присоединять к себе тех же слушателей, что и кнопка. Поэтому мы можем назначить реакцию на команды меню точно также, как делали это выше.

Задание 18.

Например, чтобы при выборе в меню элемента «Выйти» программа прекращала свою работу, замените в примере команду

fileMenu.add(**new** JMenuItem("Выйти"));

на последовательность команд:

JMenuItem exit = **new** JMenuItem("Выйти");

exit.addActionListener(**new** ActionListener() {

**public void** actionPerformed(ActionEvent event) {System.exit(0);}});

fileMenu.add(exit);

*Примечание. Командой System.exit(0) прекращается работа текущего приложения.*

**Создание контекстного меню**

Контекстное (или всплывающее) меню реализуется в классе JPopupMenu, очень похожему на класс JMenu. Отличительным методом этого класса является метод show(Component comp, int x, int y), отображающий меню в точке с заданными координатами относительно границ заданного компонента.

**Стандартные диалоговые окна**

**Класс JOptionPane**

Неотъемлемой частью большинства программ являются небольшие диалоговые окна: для вывода пользователю сообщения (например, сообщения об ошибке), для вопроса, ответ на который важен для выполнения текущего действия (например, просьба подтвердить или отменить запрашиваемую операцию(см. пример 14). Swing предоставляет в распоряжение программиста несколько готовых классов для работы с ними.

Чаще всего используется класс JOptionPane, содержащий несколько статических методов, отображающих стандартные диалоги.

Метод showMessageDialog() выводит на экран диалоговое окно, информирующее пользователя. Оно содержит надпись, значок и кнопку ОК. Существует несколько разновидностей этого метода с разными наборами параметров.

Другое часто используемое диалоговое окно — окно вопроса. В этом окне несколько кнопок, одну из которых пользователь должен нажать. В программу, вызывающую это диалоговое окно, возвращается информация о выборе пользователя, на основе которой и строится дальнейший ход работы программы.

Данное окно отображается методом showConfirmDialog(Component component, Object content). Параметры этого метода идентичны по смыслу параметрам showMessageDialog(), но в диалоговом окне появится не одна кнопка, а три: «Yes», «No» и «Cancel». Более того, метод возвращает значение, которое можно сравнить с константами YES\_OPTION, NO\_OPTION и CANCEL\_OPTION.

Пример 18.

**public class** SimpleWindow extends JFrame {

JButton button;

SimpleWindow(){

**super**("Предупреждающий диалог");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

button = **new** JButton("Выход");

button.addActionListener(**new** ActionListener() {

public void actionPerformed(ActionEvent event) {**if** (JOptionPane.showConfirmDialog(button, "Вы уверены, что хотите выйти?") == JOptionPane.YES\_OPTION) System.exit(0);} });

getContentPane().setLayout(**new** FlowLayout());

getContentPane().add(button);

setSize(200,150);}}

**Класс JFileChooser**

Swing содержит готовое окно для выбора файла (полезное, например, для программирования пункта меню Файл --> Открыть). Объект класса JFileChooser создается простым конструктором без параметров, после чего может выводиться на экран методом showOpenDialog(). Этот метод возвращает результат действий пользователя по выбору файла, который сравнивается с одной из следующих констант:

APPROVE\_OPTION — выбор файла прошел успешно. Теперь можно методом getFile() получить выбранный файл.

CANCEL\_OPTION — пользователь отменил выбор файла, щелкнув на кнопке Cancel.

ERROR\_OPTION — при выборе файла произошла ошибка, либо пользователь закрыл диалоговое окно крестиком.

Метод showSaveDialog() отображает то же самое окно, но теперь оно работает в режиме сохранения. Пользователь выбирает директорию для сохранения файла и может ввести его имя. Метод возвращает результат того же типа, что и showOpenDialog(). Если выбор пути для сохранения прошел успешно, вызов метода getFile() вернет путь, куда пользователь желает сохранить файл.

Следует иметь в виду, что сам класс JFileChooser ничего не открывает и не сохраняет. Он только возвращает путь к выбранному пользователем файлу. А открыть его или сохранить файл в заданном месте должна уже сама программа.