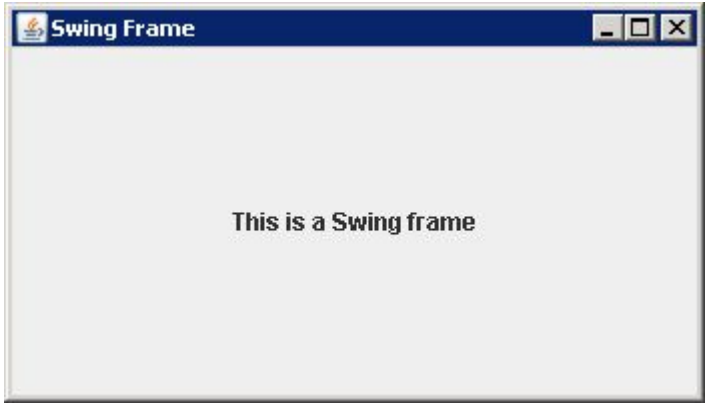
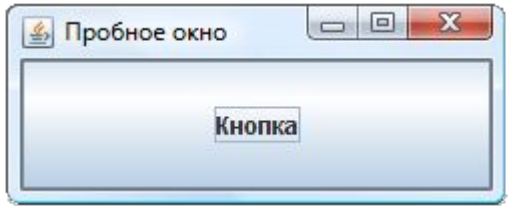
Java

Занятие 15

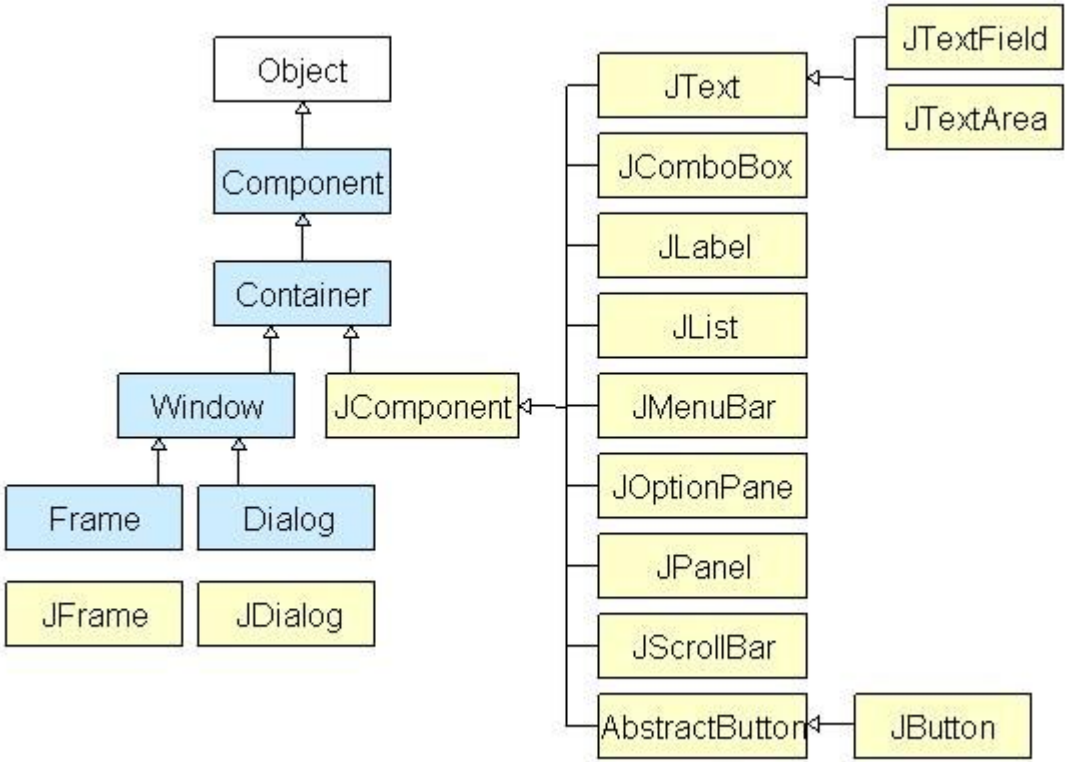
Swing введение

```
import javax.swing.*;

public class MyClass {public static void main (String [] args) {

JFrame myWindow = new JFrame("Пробное окно");

myWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

myWindow.setSize(400, 300);

myWindow.setVisible(true);

}
```





Swing


Object
Component
Container
Window
JComponent
Frame
Dialog
JFrame
JDialog
JText
JComboBox
JLabel
JList
JMenuBar
JOptionPane
JPanel
JScrollBar
AbstractButton
JTextField
JTextArea
JButton

Класс Container

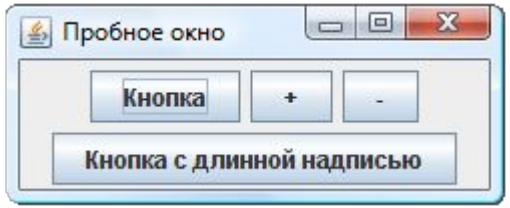
Элементы, которые содержат другие элементы, называются контейнерами. Все они являются потомками класса Container и наследуют от него ряд полезных методов:

add(Component component) — добавляет в контейнер элемент component;

remove(Component component) — удаляет из контейнера элемент component;

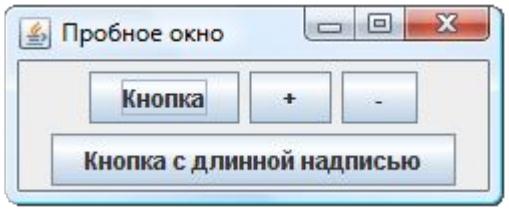
removeAll() — удаляет все элементы контейнера;

getComponentCount() — возвращает число элементов контейнера.

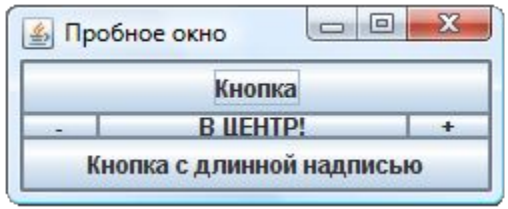


Менеджеры размещения

FlowLayout



BorderLayout



GridLayout

Основные визуальные компоненты Swing

Класс JComponent

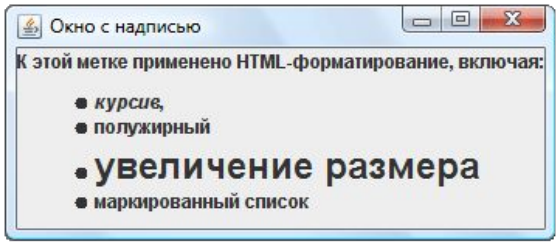
Все визуальные компоненты библиотеки Swing унаследованы от класса JComponent. Сам этот класс является абстрактными и непосредственно не используется, но все визуальные компоненты наследуют его методы. Рассмотрим наиболее полезные из них.

setEnabled(boolean enabled) используется для управления активностью компонента.

setVisible(boolean visible) управляет видимостью компонента.

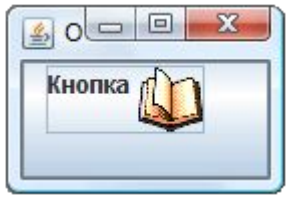
С помощью метода s**etBackground**(Color color) можно изменить цвет заднего фона компонента.

```
JLabel label = new JLabel("<html>К этой метке применено " +"HTML-форматирование, включая: <ul><li> <i>курсив</i>," +"<li><b>полужирный</b> <li><font size = +2> увеличение размера </font>" +"<li>маркированный список </ul>");
```



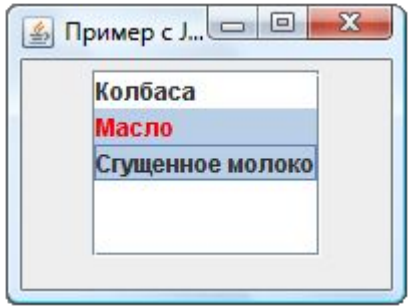
JButton

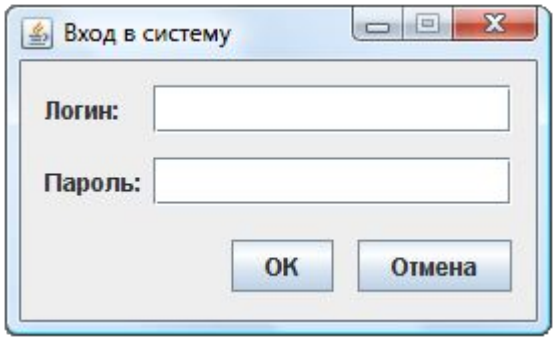
```
SimpleWindow(){
   super("Окно с кнопкой");
   setDefaultCloseOperation(EXIT_ON_CLOSE);
   JButton button = new JButton("Кнопка", new ImageIcon("1.gif"));
   button.setMargin(new Insets(0, 10, 20, 30));
   button.setVerticalTextPosition(SwingConstants.TOP);
   button.setHorizontalTextPosition(SwingConstants.LEFT);
   getContentPane().add(button);
   pack();
}
```



Список JList

Список JList — это один из сложных компонентов, для эффективной работы с которыми необходимо понимание основ библиотеки Swing, в частности, концепции «Модель-Вид».



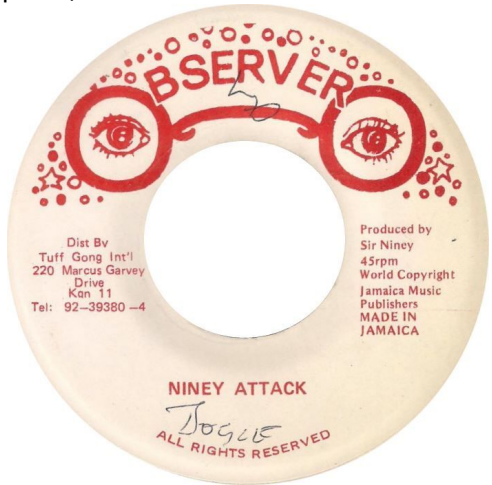


Обработка событий Swing

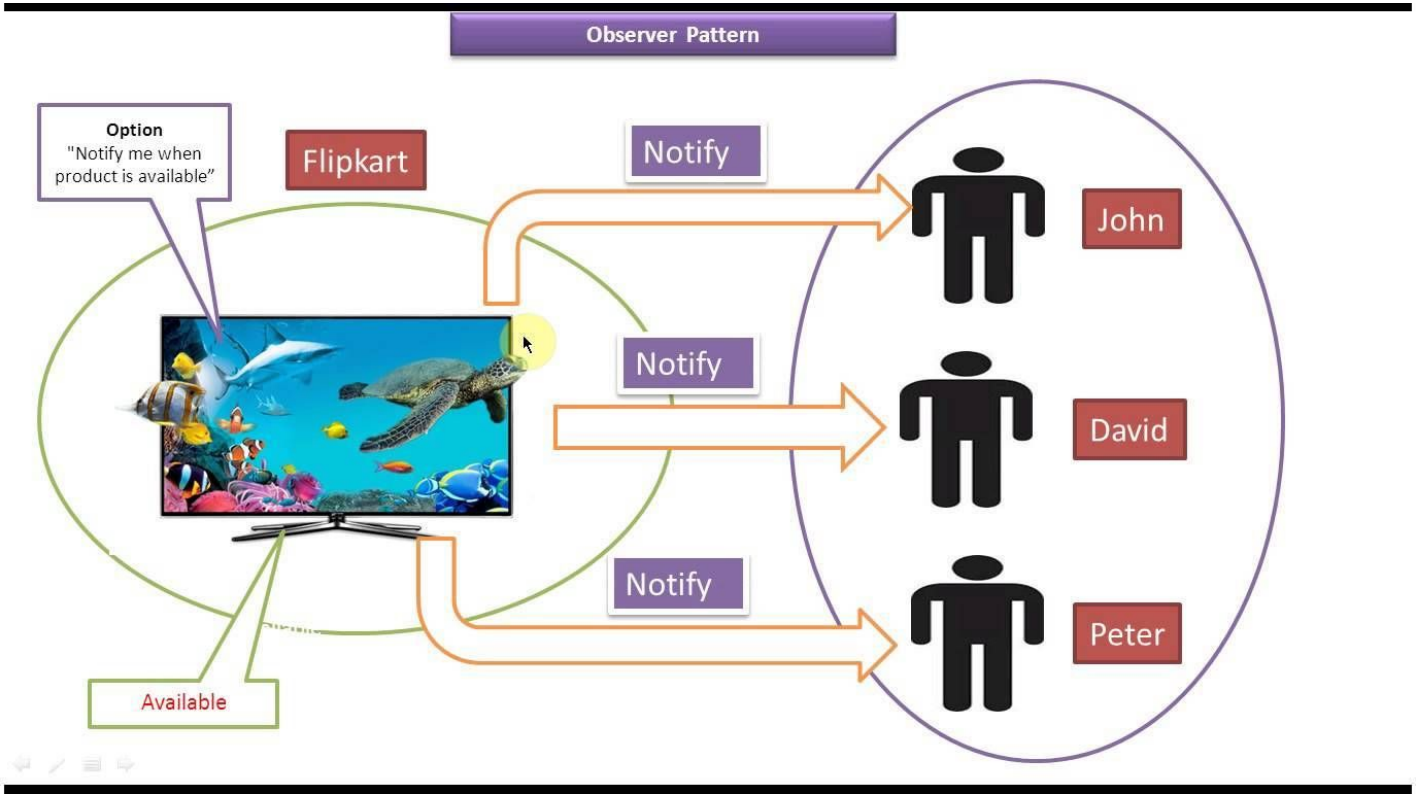
Паттер “наблюдатель”

Паттерн 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.



Обработка событий Swing



Обработка событий от мыши

Интерфейс MouseListener и обработка событий от мыши

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

Создание слушателей с помощью анонимных классов

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

vs

```
ok.addMouseListener(new 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) {}

});
```

Общая структура слушателей

Кроме слушателей MouseListener визуальные компоненты Swing поддерживают целый ряд других слушателей

Каждый слушатель должен реализовывать интерфейс ***Listener, где *** — тип слушателя. Практически каждому из этих интерфейсов (за исключением тех, в которых всего один метод) соответствует пустой класс-заглушка ***Adapter. Каждый метод интерфейса слушателя принимает один параметр типа ***Event, в котором собрана вся информация, относящаяся к событию.

Чтобы привязать слушателя к объекту (который поддерживает соответствующий тип слушателей) используется метод add***Listener(***Listener listener).

Показ кода

WindowListener

Слушатель событий окна WindowListener

Слушатель WindowListener может быть привязан только к окну и оповещается о различных событиях, произошедших с окном:

public void windowOpened(WindowEvent event) — окно открылось.

public void windowClosing(WindowEvent event) — попытка закрытия окна (например, пользователя нажал на крестик). Слово «попытка» означает, что данный метод вызовется до того, как окно будет закрыто и может воспрепятствовать этому (например, вывести диалог типа «Вы уверены?» и отменить закрытие окна, если пользователь выберет «Нет»).

public void windowClosed(WindowEvent event) — окно закрылось.

public void windowIconified(WindowEvent event) — окно свернуто.

public void windowDeiconified(WindowEvent event) — окно развернуто.

public void windowActivated(WindowEvent event) — окно стало активным.

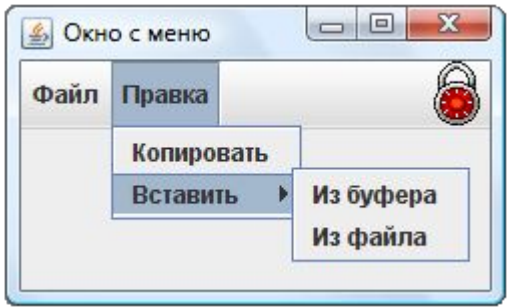
public void windowDeactivated(WindowEvent event) — окно стало неактивным.

Создание 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).

```
JMenuBar menuBar = new JMenuBar();

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



Action и Abstract Action

Основные свойства интерфейса Action (точнее, соответствующие им константы):

NAME — имя действия,

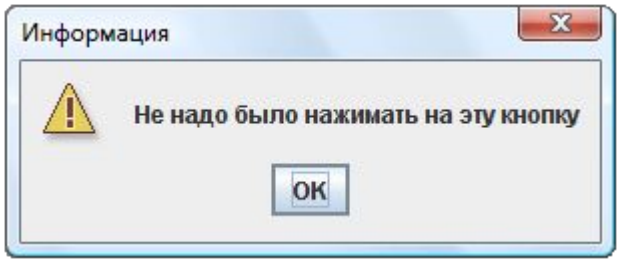
SMALL_ICON — значок, соответствующий действию,

SHORT_DESCRIPTION — краткое описание действия (для всплывающей подсказки).

Метод setEnabled(boolean enabled) позволяет сделать действие активным или неактивным.

На основе созданного действия можно создавать некоторые элементы управления, передавая это действие в качестве единственного параметра конструктора. К таким элементам управления, в частности, относятся элементы меню и кнопки.

Недостаток интерфейса Action — в нем слишком много вспомогательных абстрактных методов (их семь, в том числе setEnabled() и putValue()) и программировать их достаточно утомительно. Поэтому обычно используется реализующий данный интерфейс класс AbstractAction, в котором «не заполнен» единственный метод — actionPerformed(), а его все равно необходимо определить для программирования сути действия.



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() вернут 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.

Метод setDialogTitle(String title) позволяет задать окну заголовок.

{LET'S CODE}