6. Графический интерфейс

AWT, Swing, JavaFX

- AWT — Abstract Window Toolkit
  - библиотека, зависимая от графической подсистемы ОС
  - разный вид на разных платформах
- Swing
  - надстройка над AWT в виде легковесных Java-компонентов
  - изменяемый вид компонентов
- JavaFX
  - новая графическая библиотека с улучшенной поддержкой анимации и визуальных эффектов, возможностью задания интерфейса с помощью XML и стилей с помощью CSS

Создание графических приложений

1) Создание главного окна
2) Создание остальных компонентов интерфейса
3) Размещение компонентов интерфейса в главном окне
4) Обеспечение реакции компонентов на события

Класс java.awt.Component

- Компонент — отображаемый и взаимодействующий с пользователем элемент GUI
  - java.awt.Component - абстрактный класс — элемент GUI
  - задает размер, цвет, область отображения
  - порождает основные события

Методы класса Component

- Цвет текста и цвет фона

```
Color getForeground()   void setForeground(Color)
Color getBackground()   void setBackground(Color)
```

- Класс Color
  - Константы Color.BLACK, Color.WHITE, Color.RED ...
  - Конструкторы Color(r, g, b [,a]), Color(int [,boolean])
    - r,g,b,[a] — int (0-255), float (0.0-1.0), int (0x[AA]RRGGBB)
  - Методы
    - getRed(), getGreen(), getBlue(), getAlpha()
    - brighter(), darker()

Методы класса Component

- Шрифт

  `Font getFont()` `void setFont(Font)`

- Класс Font
  - физические (Arial, Times) и логические (Dialog, DialogInput, Serif, SansSerif, Monospaced)
  - Константы:
    - Font.DIALOG, Font.MONOSPACED, Font.SERIF, Font.SANS_SERIF
    - Font.PLAIN, Font.BOLD, Font.ITALIC
  - Конструктор Font(String name, int style, int size)
  - Методы
    - String getFontName(), int getStyle(), int getSize()

Методы класса Component

- Положение и размеры

```
void setBounds(Rectangle)     Rectangle getBounds()
void setLocation(Point)       Point getLocation()
void setSize(Dimension)       Dimension getSize()
```

- Класс Point (int x, int y)
  - getX(), getY(), setLocation(x,y)
- Класс Dimension (int height, int width)
  - getHeight(), getWidth(), setSize(h, w)
- Класс Rectangle (int x, int y, int height, int width)
  - getX(), getY(), getHeight(), getWIdth(), getLocation(), getSize()
  - setLocation(x,y), setSize(h,w), setBounds(x,y,h,w)

Методы класса Component

- Видимость и активность

```
boolean isVisible()     void setVisible(boolean)

boolean isEnabled()     void setEnabled(boolean)
```

- Компоненты изначально видимы, кроме основных окон, им надо явно вызывать метод setVisible(true)
- Компоненты изначально активны (воспринимают действия пользователя и порождают события)

Методы класса Component

- Дополнительное рисование

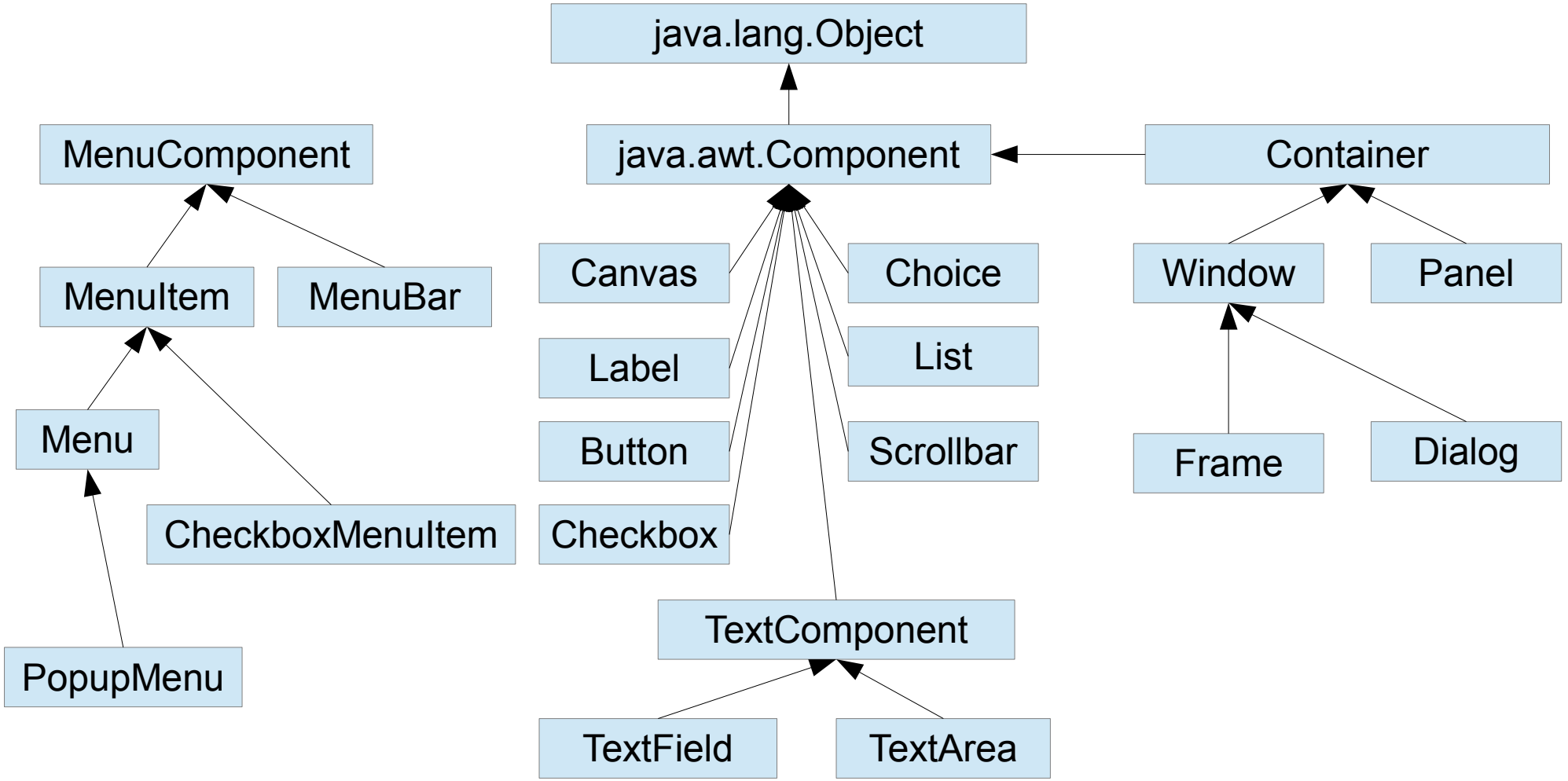
  `void paint(Graphics)  void update(Graphics)  void repaint()`

- Graphics — графический контекст компонента
- Метод paint должен содержать весь код отрисовки
- 2 варианта вызова paint — системный и программный
- Системный — первое отображение, изменение размера, необходимость перерисовки
  - JVM вызывает paint(Graphics)
- Программный — изменение состояния компонента
  - в программе вызывается repaint()
  - регистрируется событие отрисовки
  - JVM вызывает update(Graphics)

Класс java.awt.Container

- Контейнер — компонент, на котором располагаются другие компоненты
- extends java.awt.Component
- Компонент может находиться только в одном контейнере
- Методы:
  - add(Component)
  - setLayout(LayoutManager)
  - validate()

Основные компоненты и контейнеры AWT

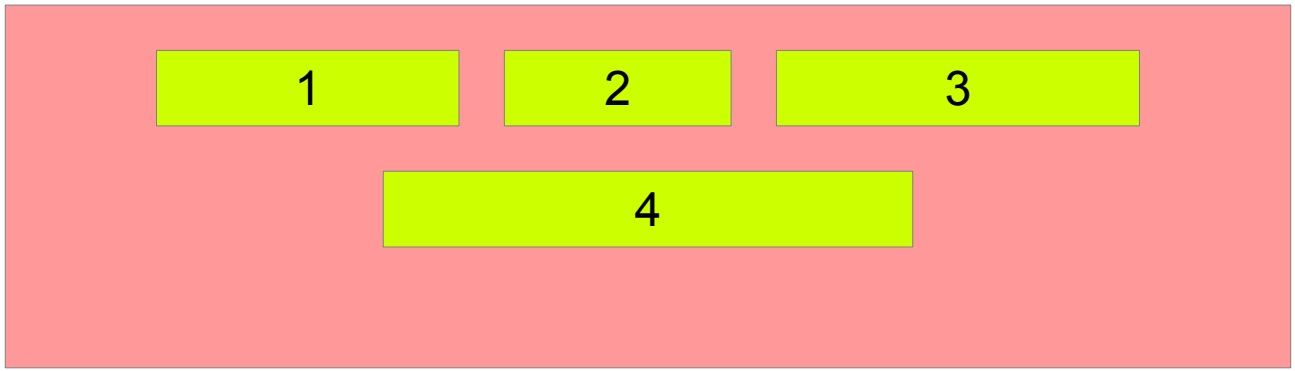


Менеджер компоновки

- Интерфейс LayoutManager
  - Container.setLayout(LayoutManager)
  - Container.add(Component)
- Интерфейс LayoutManager2
  - Container.setLayout(LayoutManager2, Object constraints)
  - Container.add(Component, Object constraint)
- Расстановка элементов
  - Container.validate()
  - Container.invalidate()
- Управление размером компонентов
  - Component.getPreferredSize()
  - Component.getMinimumSize()
  - Component.getMaximumSize()

FlowLayout

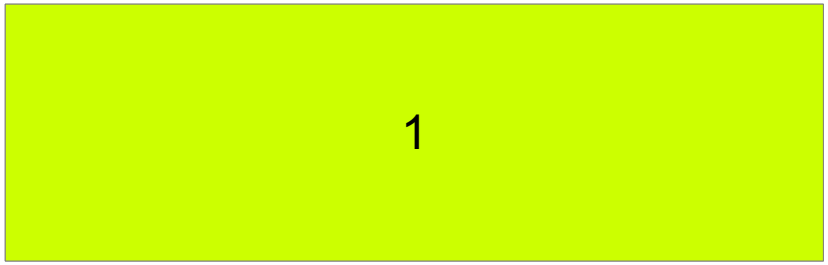
- Заполнение контейнера слева направо (или справа налево) построчно
- Компоненты сохраняют свой размер preferredSize
- Управление размещением:
  - `setHgap(int), setVgap(int) // 5`
  - `setAlignment(LEFT, RIGHT, CENTER)  // CENTER`



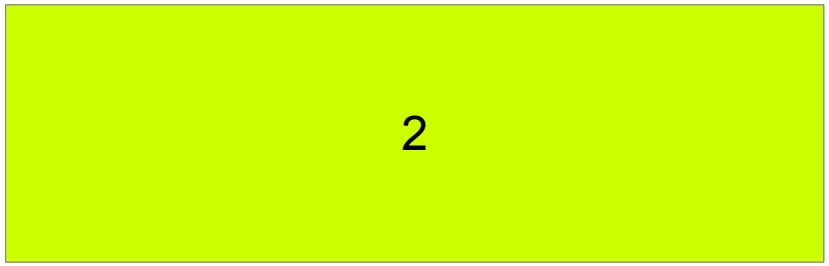
CardLayout

- Несколько компонентов отображаются в одном месте

```
CardLayout cl = new CardLayout();
p.setLayout(cl);
p.add(new Button("1"), "Card1");
p.add(new Button("2"), "Card2");
cl.show(p,"Card1");
```

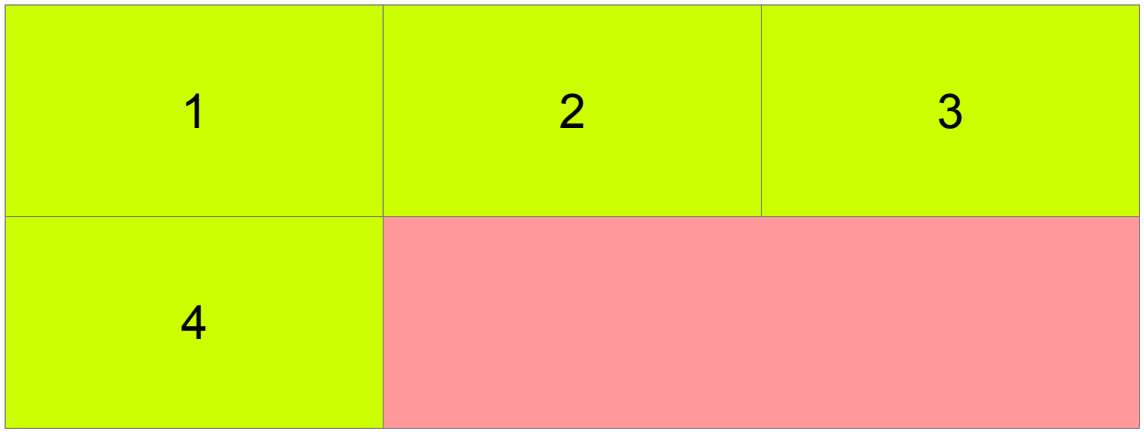



```
cl.show(p,"Card2");
```



GridLayout

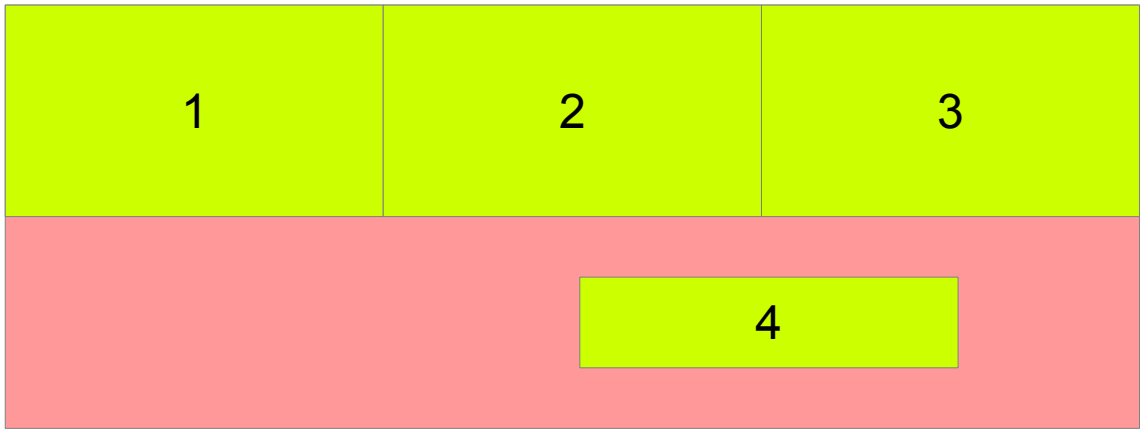
- Контейнер делится одинаковые ячейки по строкам и столбцам
- Все компоненты будут одного размера
- GridLayout(int rows, int cols)
- Управление размещением:
  - `setHgap(int), setVgap(int) // 0`
  - `setRows(int), setColumns(int) // 1, 0`



GridBagLayout

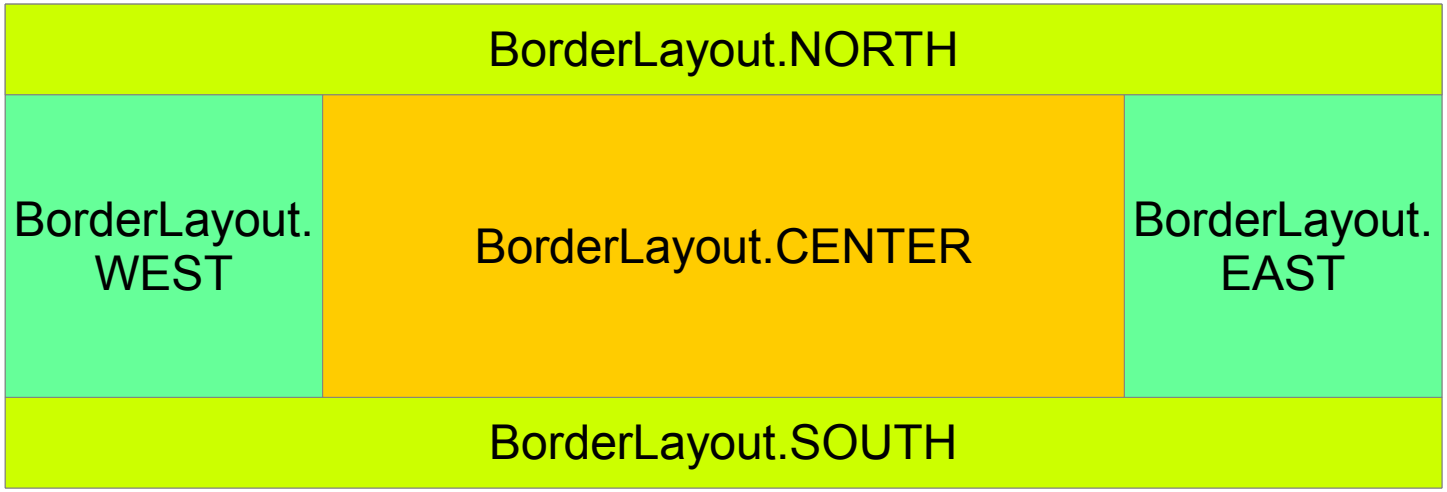
- Контейнер делится на ячейки по строкам и столбцам

```
p.setLayout(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();
c.gridX = 0; c.gridY = 0; c.fill = GridBagConstraints.BOTH;
p.add(new Button("1"), c);
c.gridX = GridBagConstraints.RELATIVE;
p.add(new Button("2"), c); p.add(bew Button("3"), c);
c.gridY = 1; c.gridX = 1; c.gridWidth = 2;
c.fill = GridBagConstraints.NONE; p.add(new Button("4"), c);
```



BorderLayout

- Компоненты располагаются в 5 областях
- p.add(new Button("1"), BorderLayout.NORTH);



Обработка событий

- Источник события — любой компонент
- Событие — потомок класса AWTEvent
- Обработчик события — реализует интерфейс XListener и соответствующие методы, в которых располагается код обработки события. Методу передается объект события

```
class A implements ActionListener {
    Button b = new Button("OK");
    Label l = new Label("Button pressed");
    l.setVisible(false);
    b.addActionListener(this);
.....
public void actionPerformed(ActionEvent e) {
    l.setVisible("true");
}
```

Обработка событий анонимным классом

```
class A {
    Button b = new Button("OK");
    final Label l = new Label("Button pressed");
    l.setVisible(false);
    b.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        l.setVisible("true");
      }
    });
```

Обработка событий

- Компонент регистрирует слушателей в списке
- При наступлении события у всех слушателей из списка вызывается метод, соответствующий событию
- В объекте события хранится информация об источнике события (getSource), времени его наступления (getWhen) и другая информация, зависящая от типа события (например, координаты клика мышки)
- Низкоуровневые события — KeyEvent, MouseEvent, MouseMotionEvent
- Семантические события — ActionEvent, ItemEvent

MouseListener, MouseMotionListener, MouseEvent

- MouseListener
  - mousePressed(MouseEvent)
  - mouseReleased(MouseEvent)
  - mouseClicked(MouseEvent)
  - mouseEntered(MouseEvent)
  - mouseExited(MouseEvent)
- MouseMotionListener
  - mouseDragged(MouseEvent)
  - mouseMoved(MouseEvent)
- MouseEvent
  - getPoint()
  - getLocationOnScreen()
  - getButton()
  - getClickCount()

Классы-адаптеры

- ```
  class X implements MouseListener {
      public void mousePressed(MouseEvent e) {
          // обработка нажатия кнопки мыши
      }
  // обработка других событий не требуется
      public void mouseClicked(MouseEvent e) { }
      public void mouseReleased(MouseEvent e) { }
      ....
  }
  ```
- ```
  class Y extends MouseAdapter {
      public void mousePressed(MouseEvent e) { ... }
  }
  ```

KeyListener, KeyEvent

- KeyListener
  - keyPressed(KeyEvent)
  - keyReleased(KeyEvent)
  - keyTyped(KeyEvent)
- KeyEvent
  - getKeyChar() // для keyTyped()
  - getKeyCode() // для keyPressed, keyReleased
  - getModifiers() // Shift, Alt, Ctrl, Meta ...
  - getKeyLocation() // Standard, Left, Right, Numpad, Unknown

WindowListener, WindowEvent

- WindowListener
  - windowOpened(WindowEvent)
  - windowClosing(WindowEvent)
  - windowClosed(WindowEvent)
  - windowActivated(WindowEvent)
  - windowDeactivated(WindowEvent)
  - windowIconified(WindowEvent)
  - windowDeiconified(WindowEvent)
- WindowEvent
  - getNewState()
  - getOldState()
  - getOppositeWindow()

ActionListener, ActionEvent

- ActionListener
  - actionPerformed(ActionEvent)
- ActionEvent
  - нажата кнопка
  - двойной клик в списке
  - выбор пункта меню
  - клавиша Enter в текстовом поле

AdjustmentListener, AdjustmentEvent

- AdjustmentListener
  - adjustmentValueChanged(AdjustmentEvent)
- AdjustmentEvent
  - int getValue()
  - boolean getValueIsAdjusting()

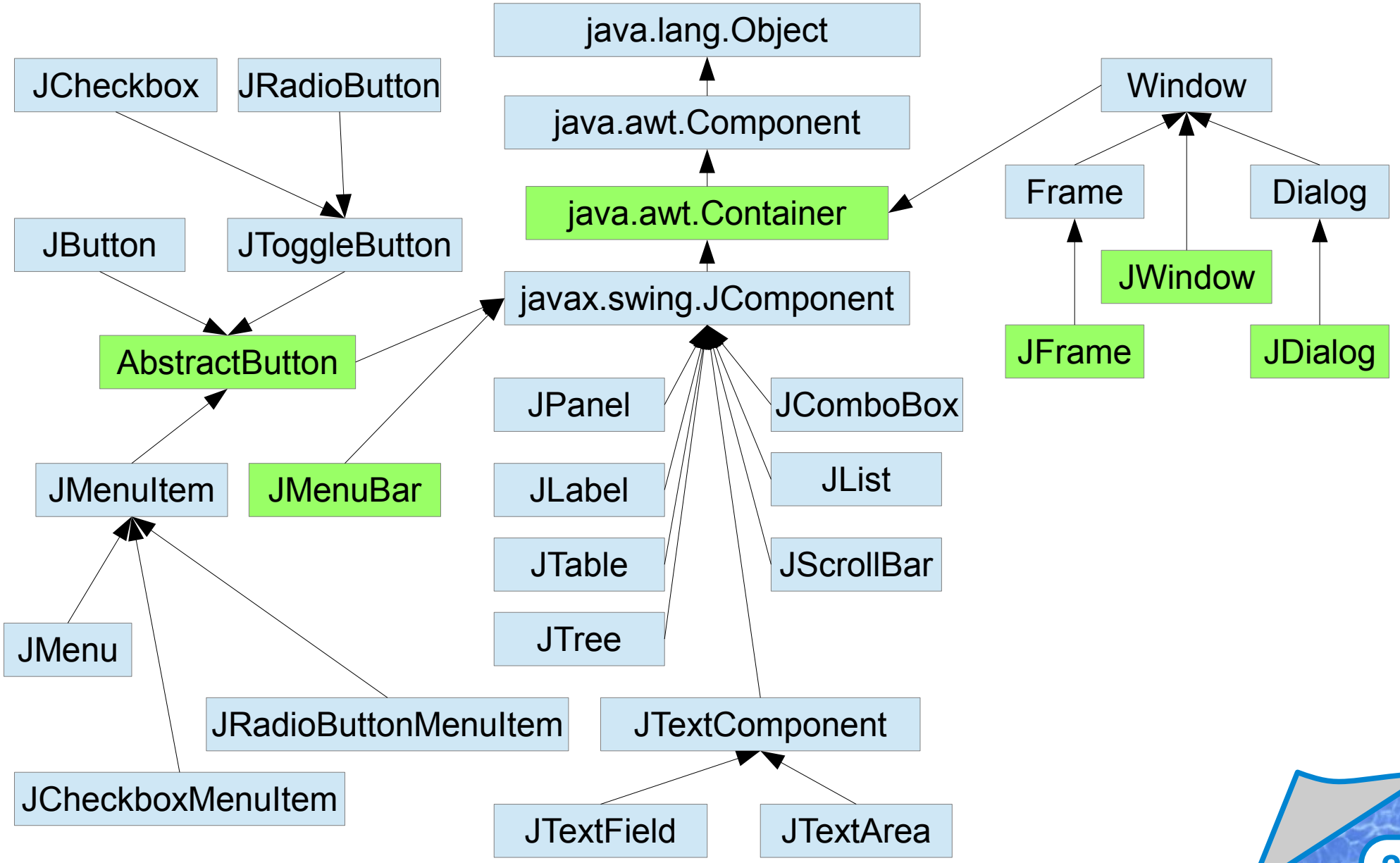
ItemListener, ItemEvent

- ItemListener
  - itemStateChanged(ItemEvent)
- ItemEvent
  - Object getItem()
  - int getStateChange() // selected-deselected
  - установка-сброс флажка
  - установка-сброс пункта меню
  - выбор элемента списка

TextListener, TextEvent

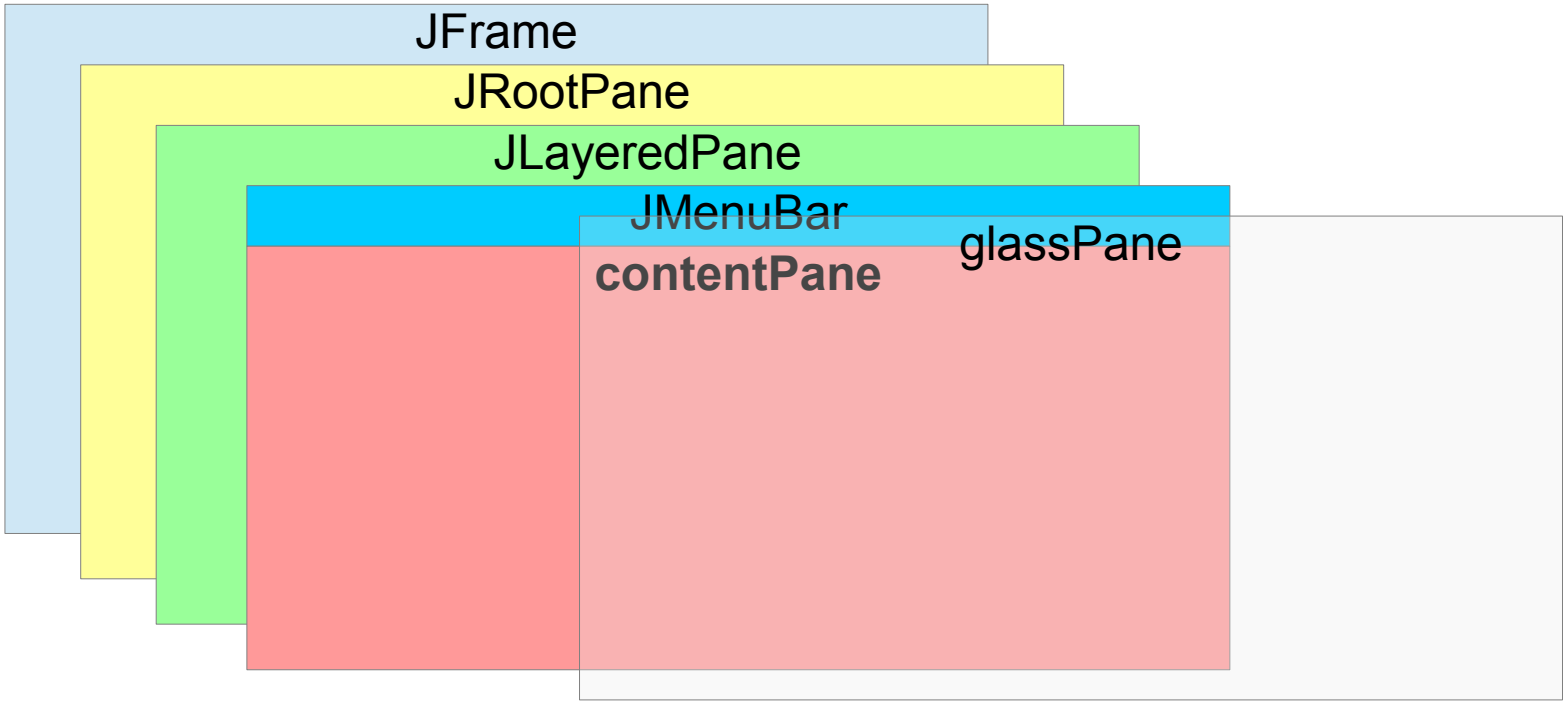
- TextListener
  - textValueChanged(TextEvent)
- TextEvent
  - изменился текст в текстовом компоненте

Основные компоненты Swing



JFrame

- Не является легковесным компонентом — это окно ОС
- Содержит набор панелей для размещения компонентов
- При создании — невидимый
- JFrame.add() = JFrame.getContentPane.add()



Организация приложения Swing

```
public class Main {
  public static void main(String... args) {
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        gui();
      }
    });
  }
  private void gui() {
    JFrame f = new JFrame();
    ...
    f.setVisible(true);
  }
}
```

JFrame

```
JFrame f = new JFrame();
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.add(new JLabel("Hello!"), BorderLayout.CENTER);
f.setJMenuBar(new JMenuBar());
f.pack(); // установка размеров фрейма
f.setVisible(true);
```

JComponent

- extends java.awt.Container — может содержать картинку
- всплывающие подсказки — setToolTipText()
- построение основано на шаблоне MVC
- встроенная двойная буферизация при отрисовке
- реализация метода paint

```
void paint(Graphics g) {
   paintComponent(g);
   paintBorder(g);
   paintChildren(g);
}
```

- Для отрисовки нужно переопределить paintComponent(g)
- Необходимо вызывать super.paintComponent(g);

MVC и Swing

- MVC — Model, View, Controller
- Модель отвечает за поведение
- Представление — отвечает за отображение
- Контроллер — связывает модель и представление и управляет ими
- Реализация Swing — Model + UI Delegate
- UI Delegate = View + Controller
- Модель может быть визуальной или моделью данных
- Одну модель данных можно назначить разным компонентам
- В случае большого числа событий можно использовать ChangeEvent — изменение в модели.

Модели и делегаты

- Модели — интерфейсы: ButtonModel, ListModel, ...
- Реализации моделей по умолчанию — классы, например: DefaultListModel, DefaultTableModel
- Для сложных моделей дополнительно имеются классы абстрактных моделей, предназначенные для облегчения написания своих классов на их основе. Например, AbstractTableModel, AbstractTreeModel
- Делегаты — потомки класса javax.swing.plaf.ComponentUI, например, ButtonUI, ListUI
- Для управления делегатами предназначен класс javax.swing.UIManager

```
UIManager.setLookAndFeel(
      UIManager.getSystemLookAndFeelClassName());
SwingUtilities.updateComponentTreeUI(frame);
frame.pack();
```

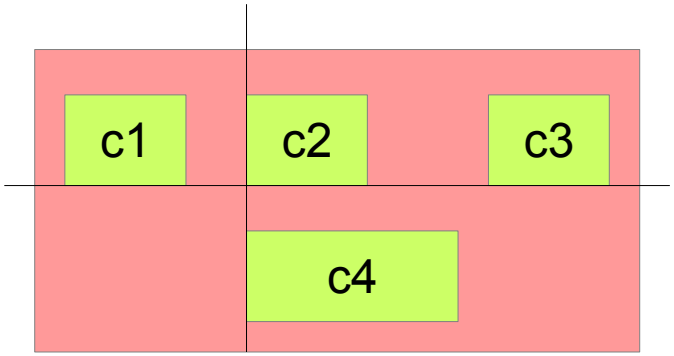
Менеджеры компоновки Swing

- BoxLayout
  - Компоненты располагаются в один ряд вертикально или горизонтально
- Класс Box — контейнер с BoxLayout
- Box.createHorizontalBox()
- Box.createVerticalBox()
- createRigidArea(Dimension)
- createHorizontalGlue()
- createVerticalGlue()
- Filler(minSize, prefSize, maxSize)

Менеджеры компоновки Swing

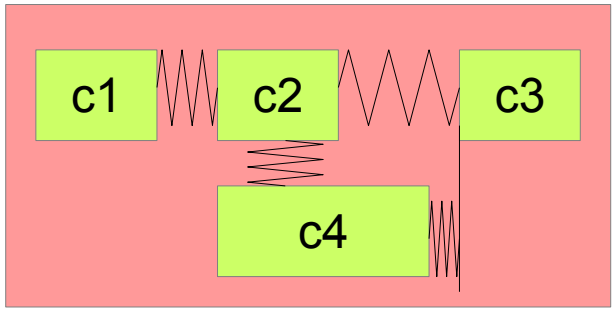
- GroupLayout
  - Все компоненты описываются дважды — горизонтальное расположение и вертикальное расположение
  - Все компоненты являются участниками групп — последовательных и параллельных

```
GroupLayout gl = new GroupLayout(p);
gl.setVerticalGroup(
   gl.createSequentialGroup()
   .addGroup(gl.createParallelGroup(GroupLayout.Alignment.BASELINE)
               .addComponent(c1)
               .addComponent(c2)
               .addComponent(c3))
   .addComponent(c4)
);
gl.setHorizontalGroup(
   gl.createSequentialGroup()
     .addComponent(c1)
     .addGroup(gl.createParallelGroup(GroupLayout.Alignment.LEADING)
                 .addCompoment(c2)
                 .addComponent(c4))
     .addComponent(c3)
);
```



Менеджеры компоновки Swing

- SpringLayout
  - Все компоненты соединены пружинами (Spring), которые имеют минимальную, максимальную и предпочтительную длину
  - Между краями соседних компонентов устанавливаются соответствующие пружины, в итоге получается компоновка, в определенных пределах растягивающаяся и сжимающаяся
  - Обычно используется автоматическими расстановщиками



JLabel

- Метка изначально прозрачная. Если нужна непрозрачная метка, то вызывается метод setOpaque(true)

JTextField, JTextArea

- Конструктор принимает строку
- Могут быть редактируемыми и нет
- События — ActionEvent, DocumentEvent

JButton, JCheckBox, JRadioButton

- Конструктор принимает строку
- Для JCheckBox и JRadioButton — еще состояние (boolean)
- JRadioButton используется в группе ButtonGroup
- События —
  - ActionEvent для JButton, JRadioButton
  - ItemEvent для JCheckBox (позволяет отследить select-deselect)
- Модель — DefaultButtonModel — элемент с двумя состояниями
- Почти так же обрабатываются JMenuItem, JMenu, JCheckBoxMenuItem, JRadioButtonMenuItem

JList

- Конструктор принимает массив или вектор объектов
- Основное событие — ListSelectionEvent
- Модель DefaultListModel — модель данных (вектор), DefaultListSelectionModel — модель вариантов выбора (одиночный, интервальный, множественный)

JComboBox

- Может быть редактируемым и нередактируемым
- Конструктор принимает массив или вектор объектов
- Основное событие — ActionEvent, иногда ItemEvent
- Модель DefaultComboBoxModel реализует 3 интерфейса — ListModel, ComboBoxModel и MutableComboBoxModel.
- По сравнению с ListModel — ComboBoxModel вводит понятие выбранный элемент (отображаемый)

JSpinner

- Составной компонент — 2 кнопки и редактор значений
- Конструктор принимает модель SpinnerModel
- Основное событие — ChangeEvent
- 3 готовых модели — SpinnerListModel, SpinnerDateModel, SpinnerNumberModel + AbstractSpinnerModel
- 3 готовых редактора — JSpinner.ListEditor, JSpinner.DateEditor, JSpinner.NumberEditor

JSlider

- Составной компонент — 2 кнопки и редактор значений
- Конструктор принимает min и max значения
- Основное событие — ChangeEvent
- Модель — DefaultBoundedRangeModel — еще используется для JProgressBar

JPanel

- Универсальный контейнер
- По умолчанию — FlowLayout