Менеджеры компоновки Layout в Swing

Особенность компоновки GUI форм в Java заключается в том, что необходимо использовать менеджеры Layout. Они определяют размер и расположение компонентов, а так же при изменении размера окна пропорционально масштабируют компоненты формы, эта особенность обусловлена тем, что код Java может запускаться на разных ОС с разными разрешением экрана, поэтому могут возникнуть проблемы при их отображении. Менеджеры компоновки Layout в Swing применяются для компонентов(JFrame,JPanel,JButton и др.).

BorderLayout По умолчанию в Swing используется менеджер BorderLayout.

BorderLayout.NORTH (верх)  
BorderLayout.SOUTH (низ)  
BorderLayout.EAST (справа)  
BorderLayout.WEST (слева)  
BorderLayout.CENTER (заполнить середину до других компонент или до краев)

По умолчанию принимается константа Center

FlowLayout устанавливает компоненты слева направо и переходит на строку вниз.

GridLayout это менеджер, который помещает компоненты в таблицу.

GridBagLayout подобно GridLayout менеджеру устанавливает компоненты в таблицу, но он более гибок, так как предоставляет возможность определять для компонентов разную ширину и высоту колонок и строк таблицы. По существу, GridBagLayout помещает компоненты в ячейки, и затем использует привилегированные размеры компонентов, чтобы определить, насколько большой ячейка должна быть.

BoxLayout позволяет управлять размещением компонентов, отдельно в вертикальном либо горизонтальном направлении помещая их, друг за другом, и управлять пространством между компонентами, используя вставки.

CardLayout менеджер предназначен для использования нескольких менеджеров.

GroupLayout менеджер имеет возможность независимо устанавливать горизонтальное и вертикальное расположение компонентов на форме.

Он использует два типа добавления компонентов параллельный и последовательный объединенный с иерархическим составом.

1. Последовательным добавляет компоненты просто помещая один за другим, точно так же как BoxLayout или FlowLayout вдоль одной оси. Положение каждого компонента определяется относительно предыдущего компонента.

2. Помещает компонентов параллельно относительно друг друга в то же самом месте. Они добавляются к верху формы или выравниваются к основанию вдоль вертикальной оси. Вдоль горизонтальной оси они устанавливаются влево или по центру, если у компонентов разный размер.

SpringLayout очень гибкий менеджер но и очень сложный для ручного кодирования изначально проектировался для использование в средах автоматического проектирования GUI например таких как NetBeans.Особенности его работы заключается в установки отношении между краями компонентов.

NullLayout Если по каким либо причинам вам необходимо самостоятельно расположить компоненты, то можно воспользоваться менеджером NullLayout установив в метод setLayout() значение null.

Нестандартного Layout надо прописывать вручную: должен реализовать методы:  
— public void addLayoutComponent(String name, Component comp)  
— public void removeLayoutComponent(Component comp)  
— public Dimension minimumLayoutSize(Container parent)  
— public Dimension preferredLayoutSize(Container parent)  
— public void layoutContainer(Container parent)

Для установки менеджера компоновки необходимо воспользоваться методом setLayout(), который определен в классе Container.В данной статье рассмотрены стандартные менеджеры компоновки AWT и Swing, если вам нужна более подробная информация по какому-то менеджеру отдельно, то я рекомендую вам обратиться к документации JAVA.

Всего в Java SE (Standart Ediotion) API описано порядка 20 различных layout manager’ов.  2011

Итак, начнем наш обзор менеджеров компоновки.

BorderLayout

По умолчанию в Swing используется менеджер BorderLayout ,в нем определены следующие константы для установки компонентов.

BorderLayout.NORTH (верх)  
BorderLayout.SOUTH (низ)  
BorderLayout.EAST (справа)  
BorderLayout.WEST (слева)  
BorderLayout.CENTER (заполнить середину до других компонент или до краев)

По умолчанию принимается константа Center.  
Пример BorderLayout.

**package** layoutdemo;   
   
**import** java.awt.BorderLayout;   
**import** javax.swing.JButton;   
**import** javax.swing.JFrame;   
**import** javax.swing.JPanel;   
   
**public** **class** BorderLayoutDemo {   
    **public** **static** **void** main(String[] args) {   
        // создаем фрейм и устанавливаем его размер.   
        JFrame jf = **new** JFrame();   
        jf.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
        jf.setSize(400, 300);   
        jf.setVisible(**true**);   
   
        // создаем панель.   
        JPanel p = **new** JPanel();   
        jf.add(p);   
   
        // к панели добавляем менеджер BorderLayout.   
        p.setLayout(**new** BorderLayout());   
   
        // к панели добавляем кнопку и устанавливаем для нее менеджер в верхнее расположение.   
        p.add(**new** JButton("Okay"), BorderLayout.NORTH);   
    }   
}

FlowLayout

FlowLayout менеджер устанавливает компоненты слева направо и при заполнении переходит на строку вниз.

Пример использования FlowLayout.

**package** layoutdemo;   
   
**import** java.awt.FlowLayout;   
**import** javax.swing.JButton;   
**import** javax.swing.JFrame;   
**import** javax.swing.JPanel;   
   
**public** **class** FlowLayoutDemo {   
    **public** **static** **void** main(String[] args) {   
        // создаем окно и устанавливаем его размер.   
        JFrame jf = **new** JFrame();   
        jf.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
        jf.setSize(400, 300);   
        jf.setVisible(**true**);   
   
        // создаем  панель.   
        JPanel p = **new** JPanel();   
        jf.add(p);   
   
        // к панели добавляем менеджер FlowLayout.   
        p.setLayout(**new** FlowLayout());   
   
        // к панели добавляем кнопки.   
        p.add(**new** JButton("start 2"));   
        p.add(**new** JButton("start 2"));   
        p.add(**new** JButton("start 3"));   
        p.add(**new** JButton("start 4"));   
        p.add(**new** JButton("start 5"));   
        p.add(**new** JButton("start 6"));   
        p.add(**new** JButton("Okay"));   
    }   
}

GridLayout

GridLayout это менеджер, который помещает компоненты в таблицу.

Пример.

**package** layoutdemo;   
   
   
**import** java.awt.GridLayout;   
**import** javax.swing.JButton;   
**import** javax.swing.JFrame;   
**import** javax.swing.JPanel;   
   
**public** **class** GridLayoutDemo {   
    **public** **static** **void** main(String[] args) {   
        JFrame jf = **new** JFrame();   
        jf.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
        jf.setSize(400, 300);   
        jf.setVisible(**true**);   
   
        // создаем  панель.   
        JPanel p = **new** JPanel();   
        jf.add(p);   
   
        // к панели добавляем менеджер GridLayout и устанавливаем размеры таблицы 3\*3.   
        p.setLayout(**new** GridLayout(3,3));   
   
        // к панели добавляем кнопку и устанавливаем для нее менеджер в верхнее расположение.   
   
        p.add(**new** JButton("start 2"));   
        p.add(**new** JButton("start 2"));   
        p.add(**new** JButton("start 3"));   
        p.add(**new** JButton("start 4"));   
        p.add(**new** JButton("start 5"));   
        p.add(**new** JButton("start 6"));   
        p.add(**new** JButton("Okay"));   
    }   
}

GridBagLayout

Этот менеджер подобно GridLayout менеджеру устанавливает компоненты в таблицу, но он более гибок, так как предоставляет возможность определять для компонентов разную ширину и высоту колонок и строк таблицы. По существу, GridBagLayout помещает компоненты в ячейки, и затем использует привилегированные размеры компонентов, чтобы определить, насколько большой ячейка должна быть.

**import** java.awt.\*;   
**import** javax.swing.JButton;   
**import** javax.swing.JFrame;   
   
**public** **class** GridBagLayoutDemo {   
    **final** **static** **boolean** shouldFill = **true**;   
    **final** **static** **boolean** shouldWeightX = **true**;   
    **final** **static** **boolean** RIGHT\_TO\_LEFT = **false**;   
   
    **public** **static** **void** addComponentsToPane(Container pane) {   
        **if** (RIGHT\_TO\_LEFT) {   
            pane.setComponentOrientation(ComponentOrientation.RIGHT\_TO\_LEFT);   
        }   
   
        JButton button;   
        pane.setLayout(**new** GridBagLayout());   
        GridBagConstraints c = **new** GridBagConstraints();   
           
        **if** (shouldFill) {   
            // натуральная высота, максимальная ширина   
            c.fill = GridBagConstraints.HORIZONTAL;   
        }   
   
        button = **new** JButton("Button 1");   
   
        **if** (shouldWeightX) {   
            c.weightx = 0.5;   
        }   
   
        c.fill = GridBagConstraints.HORIZONTAL;   
        c.gridx = 0;   
        c.gridy = 0;   
        pane.add(button, c);   
   
        button = **new** JButton("Button 2");   
        c.fill = GridBagConstraints.HORIZONTAL;   
        c.weightx = 0.5;   
        c.gridx = 1;   
        c.gridy = 0;   
        pane.add(button, c);   
           
        button = **new** JButton("Button 3");   
        c.fill = GridBagConstraints.HORIZONTAL;   
        c.weightx = 0.5;   
        c.gridx = 2;   
        c.gridy = 0;   
        pane.add(button, c);   
   
        button = **new** JButton("Long-Named Button 4");   
        c.fill = GridBagConstraints.HORIZONTAL;   
        c.ipady = 40;      // сделать эту кнопку высокой   
        c.weightx = 0.0;   
        c.gridwidth = 3;   
        c.gridx = 0;   
        c.gridy = 1;   
        pane.add(button, c);   
   
        button = **new** JButton("5");   
        c.fill = GridBagConstraints.HORIZONTAL;   
        c.ipady = 0;       // установить первоночальный размер кнопки   
        c.weighty = 1.0;   // установить отступ   
        c.anchor = GridBagConstraints.PAGE\_END; // установить кнопку в конец окна   
        c.insets = **new** Insets(10, 0, 0, 0);  // поставить заглушку   
        c.gridx = 1;       // выравнять компонент по Button 2   
        c.gridwidth = 2;   // установить в 2 колонку   
        c.gridy = 2;       // и 3 столбец   
        pane.add(button, c);   
   
    }   
   
    **private** **static** **void** createAndShowGUI() {   
        // Создание окна   
        JFrame frame = **new** JFrame("GridBagLayoutDemo");   
        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
           
        // Установить панель содержания   
        addComponentsToPane(frame.getContentPane());   
   
        // Показать окно   
        frame.pack();   
        frame.setVisible(**true**);   
    }   
   
    **public** **static** **void** main(String[] args) {   
        javax.swing.SwingUtilities.invokeLater(**new** Runnable() {   
            **public** **void** run() {   
                createAndShowGUI();   
            }   
        });   
    }   
}

BoxLayout

BoxLayout позволяет управлять размещением компонентов, отдельно в вертикальном либо горизонтальном направлении помещая их, друг за другом, и управлять пространством между компонентами, используя вставки.

**package** layoutdemo;   
   
**import** java.awt.Component;   
**import** java.awt.Container;   
**import** javax.swing.BoxLayout;   
**import** javax.swing.JButton;   
**import** javax.swing.JFrame;   
   
**public** **class** BoxLayoutDemo {   
    **public** **static** **void** addComponentsToPane(Container pane) {   
        pane.setLayout(**new** BoxLayout(pane, BoxLayout.Y\_AXIS));   
        addAButton("Button 1", pane);   
        addAButton("Button 2", pane);   
        addAButton("Button 3", pane);   
        addAButton("Long-Named Button 4", pane);   
        addAButton("5", pane);   
    }   
       
    **private** **static** **void** addAButton(String text, Container container) {   
        JButton button = **new** JButton(text);   
        button.setAlignmentX(Component.CENTER\_ALIGNMENT);   
        container.add(button);   
    }   
       
    **private** **static** **void** createAndShowGUI() {   
        // Создание фрейма   
        JFrame frame = **new** JFrame("BoxLayoutDemo");   
        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
        addComponentsToPane(frame.getContentPane());   
   
        frame.pack();   
        frame.setVisible(**true**);   
    }   
   
    **public** **static** **void** main(String[ ] args) {   
        // запустить приложение   
        javax.swing.SwingUtilities.invokeLater(**new** Runnable() {   
            **public** **void** run() {   
                createAndShowGUI();   
            }   
        });   
    }   
}

CardLayout

Этот менеджер предназначен для использования нескольких менеджеров.

**package** layout;   
   
**import** java.awt.\*;   
**import** java.awt.event.\*;   
**import** javax.swing.\*;   
   
**public** **class** CardLayoutDemo **implements** ItemListener {   
    JPanel cards;   
    **final** **static** String BUTTONPANEL = "Card with JButtons";   
    **final** **static** String TEXTPANEL = "Card with JTextField";   
       
    **public** **void** addComponentToPane(Container pane) {   
        // поместить JComboBox в JPanel для наглядности.   
        JPanel comboBoxPane = **new** JPanel();   
        String comboBoxItems[] = { BUTTONPANEL, TEXTPANEL };   
        JComboBox cb = **new** JComboBox(comboBoxItems);   
        cb.setEditable(**false**);   
        cb.addItemListener(**this**);   
        comboBoxPane.add(cb);   
   
        // Создание "cards".   
        JPanel card1 = **new** JPanel();   
        card1.add(**new** JButton("Button 1"));   
        card1.add(**new** JButton("Button 2"));   
        card1.add(**new** JButton("Button 3"));   
   
        JPanel card2 = **new** JPanel();   
        card2.add(**new** JTextField("TextField", 20));   
           
        // Создаем панель   
        cards = **new** JPanel(**new** CardLayout());   
        cards.add(card1, BUTTONPANEL);   
        cards.add(card2, TEXTPANEL);   
   
        pane.add(comboBoxPane, BorderLayout.PAGE\_START);   
        pane.add(cards, BorderLayout.CENTER);   
    }   
       
    **public** **void** itemStateChanged(ItemEvent evt) {   
        CardLayout cl = (CardLayout)(cards.getLayout());   
        cl.show(cards, (String)evt.getItem());   
    }   
       
    **private** **static** **void** createAndShowGUI() {   
        // Создание и настройка окна   
        JFrame frame = **new** JFrame("CardLayoutDemo");   
        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
        CardLayoutDemo demo = **new** CardLayoutDemo();   
        demo.addComponentToPane(frame.getContentPane());   
           
        // Показ окна   
        frame.pack();   
        frame.setVisible(**true**);   
    }   
       
    **public** **static** **void** main(String[] args) {   
        **try** {   
            UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");   
        } **catch** (UnsupportedLookAndFeelException ex) {   
            ex.printStackTrace();   
        } **catch** (IllegalAccessException ex) {   
            ex.printStackTrace();   
        } **catch** (InstantiationException ex) {   
            ex.printStackTrace();   
        } **catch** (ClassNotFoundException ex) {   
            ex.printStackTrace();   
        }   
   
        UIManager.put("swing.boldMetal", Boolean.FALSE);   
   
        javax.swing.SwingUtilities.invokeLater(**new** Runnable() {   
            **public** **void** run() {   
                createAndShowGUI();   
            }   
        });   
    }   
}

GroupLayout

GroupLayout менеджер имеет возможность независимо устанавливать горизонтальное и вертикальное расположение компонентов на форме.

Он использует два типа добавления компонентов параллельный и последовательный объединенный с иерархическим составом.

1. Последовательным добавляет компоненты просто помещая один за другим, точно так же как BoxLayout или FlowLayout вдоль одной оси. Положение каждого компонента определяется относительно предыдущего компонента.

2. Помещает компонентов параллельно относительно друг друга в то же самом месте. Они добавляются к верху формы или выравниваются к основанию вдоль вертикальной оси. Вдоль горизонтальной оси они устанавливаются влево или по центру, если у компонентов разный размер.

Пример GroupLayout

**package** layout;   
   
**import** javax.swing.\*;   
**import** **static** javax.swing.GroupLayout.Alignment.\*;   
   
**public** **class** Find **extends** JFrame {   
    **public** Find() {   
        JLabel label = **new** JLabel("Find What:");   
        JTextField textField = **new** JTextField();   
        JCheckBox caseCheckBox = **new** JCheckBox("Match Case");   
        JCheckBox wrapCheckBox = **new** JCheckBox("Wrap Around");   
        JCheckBox wholeCheckBox = **new** JCheckBox("Whole Words");   
        JCheckBox backCheckBox = **new** JCheckBox("Search Backwards");   
        JButton findButton = **new** JButton("Find");   
        JButton cancelButton = **new** JButton("Cancel");   
           
        caseCheckBox.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));   
        wrapCheckBox.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));   
        wholeCheckBox.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));   
        backCheckBox.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));   
           
        GroupLayout layout = **new** GroupLayout(getContentPane());   
        getContentPane().setLayout(layout);   
        layout.setAutoCreateGaps(**true**);   
        layout.setAutoCreateContainerGaps(**true**);   
   
        layout.setHorizontalGroup(layout.createSequentialGroup()   
                .addComponent(label)   
                .addGroup(layout.createParallelGroup(LEADING)   
                        .addComponent(textField)   
                        .addGroup(layout.createSequentialGroup()   
                        .addGroup(layout.createParallelGroup(LEADING)   
                                .addComponent(caseCheckBox)   
                                .addComponent(wholeCheckBox))   
                        .addGroup(layout.createParallelGroup(LEADING)   
                        .addComponent(wrapCheckBox)   
                        .addComponent(backCheckBox))))   
                .addGroup(layout.createParallelGroup(LEADING)   
                .addComponent(findButton)   
                .addComponent(cancelButton))   
        );   
           
        layout.linkSize(SwingConstants.HORIZONTAL, findButton, cancelButton);   
           
        layout.setVerticalGroup(layout.createSequentialGroup()   
                .addGroup(layout.createParallelGroup(BASELINE)   
                        .addComponent(label)   
                        .addComponent(textField)   
                        .addComponent(findButton))   
                .addGroup(layout.createParallelGroup(LEADING)   
                .addGroup(layout.createSequentialGroup()   
                        .addGroup(layout.createParallelGroup(BASELINE)   
                                .addComponent(caseCheckBox)   
                                .addComponent(wrapCheckBox))   
                        .addGroup(layout.createParallelGroup(BASELINE)   
                        .addComponent(wholeCheckBox)   
                        .addComponent(backCheckBox)))   
                .addComponent(cancelButton))   
        );   
           
        setTitle("Find");   
        pack();   
        setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);   
    }   
    **public** **static** **void** main(String args[]) {   
        java.awt.EventQueue.invokeLater(**new** Runnable() {   
            **public** **void** run() {   
                **try** {   
                    UIManager.setLookAndFeel(   
                            "javax.swing.plaf.metal.MetalLookAndFeel");   
                } **catch** (Exception ex) {   
                    ex.printStackTrace();   
                }   
                **new** Find().setVisible(**true**);   
            }   
        });   
    }   
}

SpringLayout

SpringLayout очень гибкий менеджер но и очень сложный для ручного кодирования изначально проектировался для использование в средах автоматического проектирования GUI например таких как NetBeans.Особенности его работы заключается в установки отношении между краями компонентов.

**package** layout;   
   
**import** javax.swing.SpringLayout;   
**import** javax.swing.Spring;   
**import** javax.swing.JFrame;   
**import** javax.swing.JLabel;   
**import** javax.swing.JTextField;   
**import** java.awt.Container;   
**import** java.awt.Component;   
   
**public** **class** SpringDemo {   
    **private** **static** **void** createAndShowGUI() {   
        // Создаем окно   
        JFrame frame = **new** JFrame("SpringDemo");   
        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
           
        // Устанавливаем менеджер SpringLayout   
        Container contentPane = frame.getContentPane();   
        SpringLayout layout = **new** SpringLayout();   
        contentPane.setLayout(layout);   
           
        // Создание компонентов   
        JLabel label = **new** JLabel("Label: ");   
        JTextField textField = **new** JTextField("Text field", 15);   
        contentPane.add(label);   
        contentPane.add(textField);   
   
        // Делаем ограничения для label   
        SpringLayout.Constraints contentPaneCons =   
                layout.getConstraints(contentPane);   
        contentPaneCons.setX(Spring.sum(Spring.constant(5),   
                contentPaneCons   
                        .getConstraint(SpringLayout.WEST)));   
        contentPaneCons.setY(Spring.sum(Spring.constant(5),   
                contentPaneCons   
                        .getConstraint(SpringLayout.NORTH)));   
           
        // Делаем ограничения для text field   
        SpringLayout.Constraints textFieldCons =   
                layout.getConstraints(textField);   
        textFieldCons.setX(Spring.sum(   
                Spring.constant(5),   
                contentPaneCons.getConstraint(SpringLayout.EAST)));   
        textFieldCons.setY(Spring.constant(5));   
        setContainerSize(contentPane, 5);   
           
        // Делаем окно видемым   
        frame.pack();   
        frame.setVisible(**true**);   
    }   
       
    **public** **static** **void** main(String[] args) {   
        javax.swing.SwingUtilities.invokeLater(**new** Runnable() {   
            **public** **void** run() {   
                createAndShowGUI();   
            }   
        });   
    }   
       
    **public** **static** **void** setContainerSize(Container parent, **int** pad) {   
        SpringLayout layout = (SpringLayout) parent.getLayout();   
        Component[] components = parent.getComponents();   
        Spring maxHeightSpring = Spring.constant(0);   
        SpringLayout.Constraints pCons = layout.getConstraints(parent);   
           
        // устанавливаем контейнеры в правый край   
        // с его rightmost компонентом +дополнения.    
        Component rightmost = components[components.length - 1];   
        SpringLayout.Constraints rCons =   
                layout.getConstraints(rightmost);   
        pCons.setConstraint(   
                SpringLayout.EAST,   
                Spring.sum(Spring.constant(pad),   
                        rCons.getConstraint(SpringLayout.EAST)));   
           
        // устанавливаем контейнеры в нижний край   
        // с его компонентом +дополнения.    
        **for** (**int** i = 0; i < components.length; i++) {   
            SpringLayout.Constraints cons =   
                    layout.getConstraints(components[i]);   
            maxHeightSpring = Spring.max(maxHeightSpring,   
                    cons.getConstraint(   
                            SpringLayout.SOUTH));   
        }   
        pCons.setConstraint(   
                SpringLayout.SOUTH,   
                Spring.sum(Spring.constant(pad),   
                        maxHeightSpring));   
    }   
}

Если по каким либо причинам вам необходимо самостоятельно расположить компоненты, то можно воспользоваться менеджером NullLayout установив в метод setLayout() значение null.

Пример NullLayout.

**package** layout;   
   
**import** java.awt.Container;   
**import** java.awt.Insets;   
**import** java.awt.Dimension;   
**import** javax.swing.JButton;   
**import** javax.swing.JFrame;   
   
**public** **class** AbsoluteLayoutDemo {   
    **public** **static** **void** addComponentsToPane(Container pane) {   
        pane.setLayout(**null**);   
   
        JButton b1 = **new** JButton("one");   
        JButton b2 = **new** JButton("two");   
        JButton b3 = **new** JButton("three");   
   
        pane.add(b1);   
        pane.add(b2);   
        pane.add(b3);   
   
        Insets insets = pane.getInsets();   
        Dimension size = b1.getPreferredSize();   
        b1.setBounds(25 + insets.left, 5 + insets.top,   
                     size.width, size.height);   
        size = b2.getPreferredSize();   
        b2.setBounds(55 + insets.left, 40 + insets.top,   
                     size.width, size.height);   
        size = b3.getPreferredSize();   
        b3.setBounds(150 + insets.left, 15 + insets.top,   
                     size.width + 50, size.height + 20);   
    }   
   
    **private** **static** **void** createAndShowGUI() {   
        JFrame.setDefaultLookAndFeelDecorated(**true**);   
   
        JFrame frame = **new** JFrame("AbsoluteLayoutDemo");   
        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
   
        addComponentsToPane(frame.getContentPane());   
   
        Insets insets = frame.getInsets();   
        frame.setSize(300 + insets.left + insets.right,   
                      125 + insets.top + insets.bottom);   
        frame.setVisible(**true**);   
    }   
   
    **public** **static** **void** main(String[] args) {   
        javax.swing.SwingUtilities.invokeLater(**new** Runnable() {   
            **public** **void** run() {   
                createAndShowGUI();   
            }   
        });   
    }   
}

Если Вам понравилась статья, проголосуйте за нее

|  |
| --- |
|  |

А теперь давайте рассмотрим вариант создания нестандартного layout. Любой layout должен просто реализовать необходимые методы. Вот они:  
— public void addLayoutComponent(String name, Component comp)  
— public void removeLayoutComponent(Component comp)  
— public Dimension minimumLayoutSize(Container parent)  
— public Dimension preferredLayoutSize(Container parent)  
— public void layoutContainer(Container parent)

Наиболее важным является последний метод. Он как раз и занимается тем, что «расставляет» все элементы в том порядке, в котором нам надо.  
Я написал в качестве примера простой layout, который располагает свои элементы вдоль диагонали.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | import java.awt.\*;  import javax.swing.\*;    class DiagLayoutTest extends JFrame  {      public DiagLayoutTest()      {          getContentPane().setLayout(new DiagLayout());          for(int k=0; k<5; k++) {            getContentPane().add(new JButton(""+k));          }          for(int k=0; k<5; k++) {            getContentPane().add(new JLabel(""+k, JLabel.CENTER));          }            setBounds(100,100,600,400);      }        public static void main(String[] args)      {          DiagLayoutTest flt = new DiagLayoutTest();          flt.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);          flt.setVisible(true);      }  } |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63 | // Вот наш собственный layout  import java.awt.Component;  import java.awt.Container;  import java.awt.Dimension;  import java.awt.LayoutManager;  import java.awt.Rectangle;    class DiagLayout implements LayoutManager  {      // Эти методы нас не интересуют и их можно сделать пустыми      public void addLayoutComponent(String name, Component comp)      {}      public void removeLayoutComponent(Component comp)      {}        public Dimension minimumLayoutSize(Container parent)      {          return computeLayoutSize(parent);      }        public Dimension preferredLayoutSize(Container parent)      {          return computeLayoutSize(parent);      }        private Dimension computeLayoutSize(Container parent)      {          int prefWidth = 0;          int prefHeight = 0;          Component[] components = parent.getComponents();          for(int k=0; k<components.length; k++) {              prefWidth += components[k].getWidth();              prefHeight += components[k].getHeight();          }          return new Dimension(prefWidth, prefHeight);      }        // Вот наш фактически самый главный метод. Здесь мы располагаем компоненты      // по диагонали      public void layoutContainer(Container parent)      {          // Получаем список компонентов          Component[] components = parent.getComponents();          int row = 0;          int col = 0;          // Эти две строки можно закрыть комментариями (см. замечание ниже)          int width = parent.getWidth()/components.length;          int height = parent.getHeight()/components.length;          for(int k=0; k<components.length; k++) {              // Вы можете снять комментарии здесь и поставить их двумя строками              // выше и увидите разницу              //int width = (int)(components[k].getPreferredSize().getWidth());              //int height = (int)(components[k].getPreferredSize().getHeight());              // Определяем местоположение компонента и его размеры              Rectangle r = new Rectangle(col, row, width, height);              // Устанавливаем его              components[k].setBounds(r);              // Заготавливаем координаты следующего компонента              col += width;              row += height;          }      }  } |