

Теория и технология программирования

Программирование на языке Java

Лекция 10. Компоненты и события

Глухих Михаил Игоревич, к.т.н., доц.

[mailto: glukhikh@mail.ru](mailto:glukhikh@mail.ru)

Общие для компонентов события

- ❑ Действия с компонентом в целом
(изменения размера и положения)
ComponentEvent
- ❑ Действия с фокусом
(выбор этого или другого компонента)
FocusEvent
- ❑ Действия с мышью
MouseEvent
- ❑ Действия с клавиатурой
KeyEvent

Общая схема обработки событий

- Для каждого события есть соответствующий ему интерфейс-слушатель, например, *ComponentListener*
- В интерфейсе определено несколько методов-обработчиков, вызывающихся при выполнении соответствующих действий - например, *componentMoved* для изменения размера компонента
- Для каждого интерфейса-слушателя есть примитивная реализация – класс вида *ComponentAdapter*, в котором все методы-обработчики пустые

Общая схема обработки событий

- ❑ У всех обработчиков есть событие-аргумент (например, *ComponentEvent*), позволяющее уточнить детали происходящего (например, каким стал размер компонента)
- ❑ Для обработки событий определяется класс, реализующий соответствующий интерфейс (часто класс является вложенным)
- ❑ Затем вызывается метод *addZZZListener* (например, *addComponentListener*) исходного компонента

Обработка событий мыши

- Интерфейс `MouseListener`
 - `mouseClicked` – щелчок (нажали и отпустили)
 - `mousePressed` – нажатие (момент нажатия)
 - `mouseReleased` – отпускание (момент отпускания)
 - `mouseEntered` - появление курсора мыши в компоненте
 - `mouseExited` - исчезновение курсора мыши из компонента
- Интерфейс `MouseMotionListener`
 - `mouseMoved` - движение мыши
 - `mouseDragged` - перетаскивание мышью
- Интерфейс `MouseWheelListener`
 - `mouseWheelMoved` - передвинуто колесо

Обработка событий мыши

□ Методы MouseEvent:

- `getX()`, `getY()` - координаты
- `getButton()` - к какой клавише относится (`BUTTON1`, `BUTTON2`, `BUTTON3`)
- `getClickCount()` - кратность щелчка
- `getModifiers()` - различные модификаторы (какие еще клавиши были нажаты и т.д.)

Пример

- Изменим нашу заставку с шариком так, чтобы нажатием левой клавиши мыши можно было остановить движение шарика, а отпусканием - восстановить движение шарика
- Какие возможности для этого следует добавить в сам шарик?

Метод проверки положения

```
public boolean inside(int px, int py) {  
    return (px-x) * (px-x) + (py-y) * (py-y) <  
        radius*radius;  
}
```


Добавим слушателя

```
private boolean hold; // надо ли «держат» шарик
// ...
MouseListener = new MouseListener() {
    // ...
    public void mousePressed(MouseEvent e) {
        if (e.getButton() == MouseEvent.BUTTON1)
            hold = ball.inside(e.getX(), e.getY());
    }
    public void mouseReleased(MouseEvent e) {
        if (e.getButton() == MouseEvent.BUTTON1)
            hold = false;
    }
};
this.addMouseListener(mouseListener);
```

Демонстрация

□ См. пример

Обработка событий клавиатуры

- Происходит в компоненте, владеющем фокусом
- Интерфейс `KeyListener`
 - `keyPressed` - нажатие клавиши
 - `keyReleased` - отпущание клавиши
 - `keyTyped` - ввод очередного символа

Обработка событий клавиатуры

□ Методы KeyEvent

- getKeyChar - возвращает соответствующий символ или константу CHAR_UNDEFINED
- getKeyCode - возвращает код нажатой клавиши, например, VK_DOWN
- getModifiers - различные модификаторы (нажата ли ALT, SHIFT)

Пример

- Изменим нашу заставку так, чтобы нажатием на клавиши ВВЕРХ, ВНИЗ, ВПРАВО или ВЛЕВО можно было управлять движением шарика (например, изменять скорость на единицу в соответствующую сторону)
- Какие возможности следует добавить?

Метод изменения скорости

```
public void touch(int ddx, int ddy) {  
    dx += ddx;  
    if (dx > 20)  
        dx = 20;  
    if (dx < -20)  
        dx = -20;  
    dy += ddy;  
    if (dy > 20)  
        dy = 20;  
    if (dy < -20)  
        dy = -20;  
}
```

Метод изменения скорости (для панели)

```
public void touchBall(int dx, int dy) {  
    controlledBall.touch(dx, dy);  
}
```

// Метод необходим, так как

// управлять нам придется из фрейма

- ❑ События клавиатуры приходят в главный фрейм, пока нет курсора ввода
- ❑ Если курсор ввода есть, события приходят в компонент, владеющий курсором ввода

Добавим слушателя

```
// ...
KeyListener = new KeyListener() {
    public void keyPressed(KeyEvent e) {
        switch (e.getKeyCode()) {
            case KeyEvent.VK_DOWN:
                panel.touchBall(0, 1);
                break;
            case KeyEvent.VK_UP:
                panel.touchBall(0, -1);
                break;
            // ...
        }
    }
}
// ...
this.addKeyListener(keyListener);
```


Демонстрация

□ См. пример

События, связанные с определенными компонентами

- ❑ Добавление/удаление компонента - ContainerEvent-контейнеры
- ❑ Открытие, закрытие, разворачивание, сворачивание окон - WindowEvent - окна, фреймы, диалоги
- ❑ Сигналы к выполнению действий - ActionEvent - клавиши, команды меню
- ❑ Изменение введенного текста - TextEvent - компоненты, содержащие текст
- ❑ Изменение выбранного пункта - ItemEvent - компоненты, содержащие перечень пунктов
- ❑ Сдвиг - AdjustmentEvent - полосы прокрутки

Размещение компонентов

- При проектировании графического интерфейса целесообразно
 - отделять компоненты от рисованной части приложения
 - отделять группы компонентов друг от друга
- Для этого целесообразно создать в основном фрейме **несколько** панелей

Размещение компонентов

- Проблема - как задать размеры и местоположение компонентов?

Размеры и местоположение компонентов

- Способ 1 - ручное задание
 - setSize - размеры
 - setLocation - положение левого верхнего угла
 - setBounds - и то и другое сразу

Размеры и местоположение компонентов

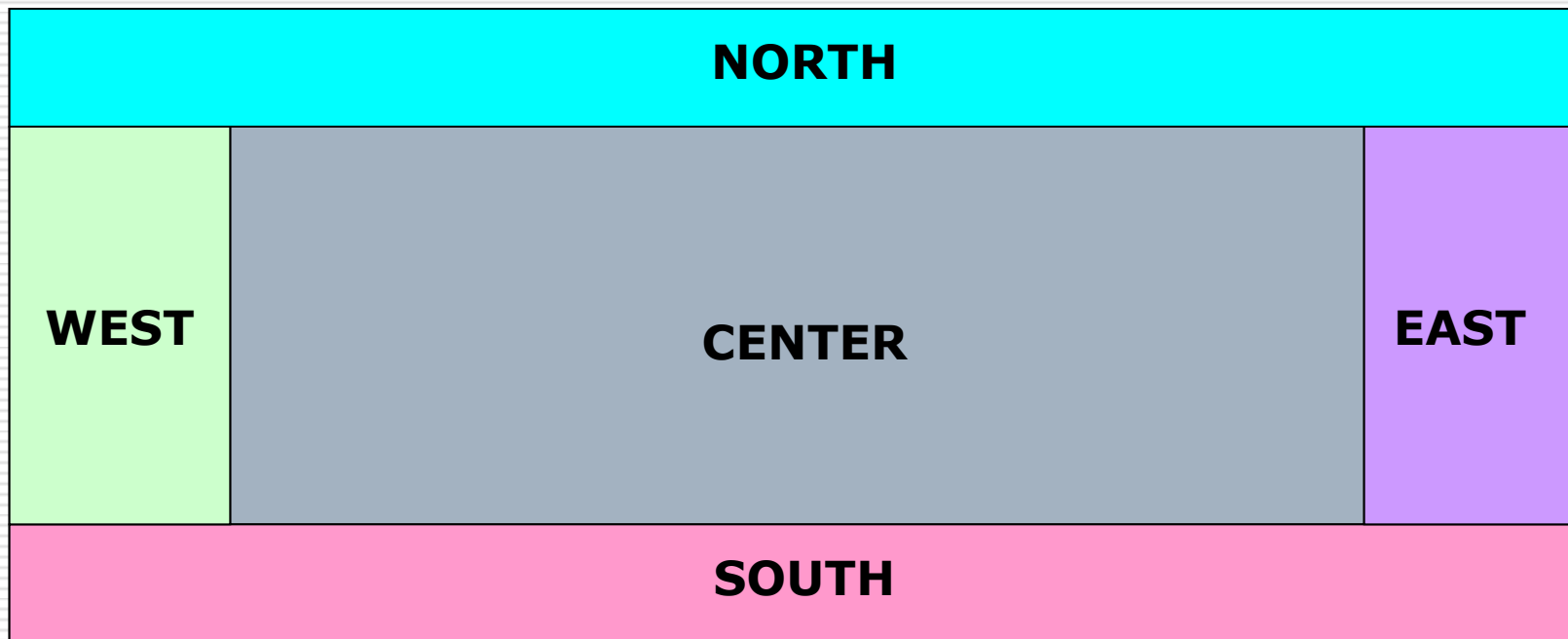
- Что делать, если размеры контейнера изменились?
 - можно вызвать `setResizable(false)`
 - или же использовать способ 2
- Способ 2 - автоматизированный выбор размеров
 - при этом используются менеджеры размещения *LayoutManager*
 - а также **предпочтительный** размер - выбирается с помощью `setPreferredSize`

Менеджеры размещения

- ❑ Выбираются вызовом `setLayout(manager)`
- ❑ `setLayout(null)` - убрать менеджер размещения, положение задавать самостоятельно
- ❑ `setLayout(new FlowLayout())` - заполнять контейнер компонентами слева направо и сверху вниз, используя предпочтительные размеры

Разделение на области

- Используется менеджер BorderLayout



Пример

- Создадим положение с тремя панелями:
 - заголовочная панель (с большой надписью посередине)
 - основная панель (для рисунков)
 - панель опций (для настроек)

Конструктор

```
// ...
this.setLayout(new BorderLayout());
JPanel mainPanel = new JPanel();
mainPanel.setBackground(Color.CYAN);
mainPanel.setBorder(new LineBorder(Color.BLUE, 2));
this.add(mainPanel, BorderLayout.CENTER);
JPanel headPanel = new JPanel();
headPanel.setPreferredSize(new Dimension(200, 30));
headPanel.setBorder(new LineBorder(Color.ORANGE, 2));
headPanel.setBackground(Color.YELLOW);
this.add(headPanel, BorderLayout.NORTH);
JPanel optionPanel = new JPanel();
optionPanel.setPreferredSize(new Dimension(200, 300));
optionPanel.setBorder(new LineBorder(Color.DARK_GRAY, 2));
this.add(optionPanel, BorderLayout.EAST);
// ...
```

Простые компоненты - метка

- Неизменяемая надпись
- Описывается классами JLabel, Label
- Основные методы
 - JLabel(String) - конструктор
 - setText(String) - смена текста
 - setIcon(Icon) - смена иконы
 - setHorizontalAlignment, setVerticalAlignment - смена выравнивания

Пример

```
private void initHeadPanel() {
    headPanel = new JPanel();
    headPanel.setPreferredSize(new Dimension(200, 30));
    headPanel.setBorder(new LineBorder(Color.ORANGE, 2));
    headPanel.setBackground(Color.YELLOW);
    JLabel headLabel = new JLabel("ПРОСТЫЕ КОМПОНЕНТЫ");
    ImageIcon icon = new ImageIcon("headIcon.gif");
    headLabel.setHorizontalAlignment(SwingConstants.CENTER);
    headLabel.setIcon(icon);
    headLabel.setBorder(new LineBorder(Color.BLACK, 1));
    headPanel.add(headLabel);
    this.add(headPanel, BorderLayout.NORTH);
}
```

// Обратите внимание, что метка находится посередине

Простые компоненты - клавиша

- ❑ Выполняется определенное действие при нажатии
- ❑ Описывается классами JButton, Button
- ❑ Основные методы
 - JButton(String) - конструктор
 - addActionListener - добавить обработчик
 - setText(String) - поменять надпись
 - setIcon(Icon) - поменять икону
 - setEnabled(boolean) - включить/выключить

Пример

```
private void initOptionPanel() {
    optionPanel = new JPanel();
    optionPanel.setPreferredSize(new Dimension(200, 300));
    optionPanel.setBorder(new
        LineBorder(Color.DARK_GRAY, 2));
    JButton exitButton = new JButton("Выход");
    exitButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    });
    optionPanel.add(exitButton);
    this.add(optionPanel, BorderLayout.EAST);
}
// Клавиша появилась сверху, по центру
```

Простые компоненты - пункт выбора

- ❑ Включает надпись и возможность отметки
- ❑ Описывается классами `JCheckBox`, `Checkbox`
- ❑ Основные методы
 - `JCheckBox(String)` - конструктор
 - `addItemListener` - добавить обработчик смена состояния
 - `isSelected` - узнать выбрана или нет
 - `setSelected(boolean)` - установить состояние

Пример

```
private void initOptionPanel() {
    // ...
    drawingBox = new JCheckBox("Показать рисунок");
    drawingBox.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            mainPanel.setDrawing(drawingBox.isSelected());
        }
    });
    optionPanel.add(drawingBox);
    // ...
}
// Пункт выбора появился над клавишей, по центру
```


Пример

```
public class MainPanel extends JPanel {
    private boolean drawingOn = false;
    public void setDrawing(boolean on) {
        drawingOn = on;
        repaint();
    }
    public void paint(Graphics g) {
        super.paint(g);
        if (drawingOn) {
            g.setColor(Color.BLUE);
            g.drawRect(50, 50, 100, 100);
        }
    }
}
```

Простые компоненты - группа выбора

- ❑ Включает несколько пунктов, из которых выбирается один
- ❑ Описывается классами `ButtonGroup` и `JRadioButton`
- ❑ Основные методы
 - `ButtonGroup.add` - добавление пункта в группу
 - `JRadioButton(String)` - конструктор
 - `JRadioButton.addActionListener` - добавление обработчика
 - `JRadioButton.setActionCommand` - установка строки, соответствующей команде

Группа выбора - традиционное применение

- Создать панель, в которой будет находиться группа
- Создать все пункты и обработчики
- Создать группу
- Добавить пункты как в панель, так и в группу

Пример

```
private void initRadioPanel() {
    radioPanel = new JPanel();
    radioPanel.setBorder(new
        LineBorder(Color.BLACK, 1));
    radioPanel.setPreferredSize(new
        Dimension(125, 100));
    radioPanel.setLayout(new
        FlowLayout(FlowLayout.LEFT));
    JLabel head = new JLabel("Выберите");
    radioPanel.add(head);
    square = new JRadioButton("квадрат");
    triangle = new JRadioButton("треугольник");
```

Пример

```
ActionListener listener = new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        String command = e.getActionCommand();  
        if (command.equals("Square")) {  
            mainPanel.setDrawingKind(0);  
        } else {  
            mainPanel.setDrawingKind(1);  
        }  
    }  
};
```

Пример

```
square.setEnabled(false);
square.setSelected(true);
square.addActionListener(listener);
square.setActionCommand("Square");
triangle.setEnabled(false);
triangle.setSelected(false);
triangle.addActionListener(listener);
triangle.setActionCommand("Triangle");
radioPanel.add(square);
radioPanel.add(triangle);
bg = new ButtonGroup();
bg.add(square);
bg.add(triangle);
optionPanel.add(radioPanel);
}
```

Создание диалога

- Диалоги общего вида наследуются от класса *JDialog*
- Диалог, за несколькими исключениями, похож на фрейм
 - у диалога всегда есть родительское окно (фрейм, диалог)
 - диалог может быть модальным или нет

Создание диалога в графическом редакторе

- ❑ New → Other →
Swing GUI Forms → JDialogForm
- ❑ При создании с помощью редактора для
данного класса появляются два режима -
Source и Design
- ❑ Графический редактор позволяет задать
начальное положение компонентов
диалога, значения их свойств
(properties), методы обработки событий

Пример

- Диалог "Начало игры"

Итоги

- Рассмотрено
 - Обработка событий мыши и клавиатуры
 - Менеджеры размещения (layouts)
 - Компоненты
- Далее
 - Пример сложного GUI-приложения