Теория и технология программирования

Программирование на языке Java

Лекция 8. Введение в графические библиотеки JDK (AWT/Swing)

Глухих Михаил Игоревич, к.т.н., доц.
mailto: glukhikh@mail.ru

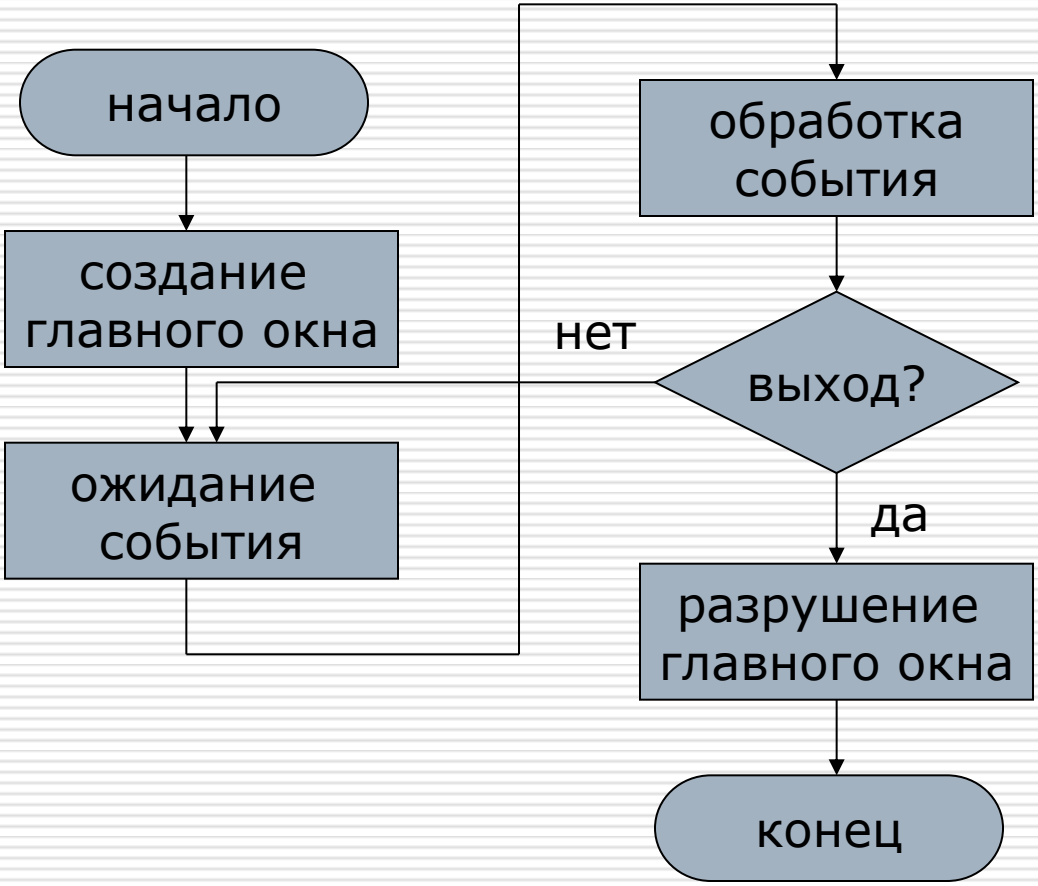
Графический интерфейс пользователя

- GUI-приложения всегда имеют **главное окно**
- Окно может включать в себя
  - разнообразную графическую информацию
  - законченные элементы графического интерфейса - **компоненты**
- Взаимодействие «приложение→пользователь» осуществляется через
  - содержимое окна
  - содержимое компонентов
  - дополнительные окна
- Взаимодействие «пользователь→приложение»
  - осуществляется через механизм **событий**

Механизм взаимодействия «пользователь→приложение»

- Консольное приложение
  - если требуется получить информацию от пользователя, вызывается соответствующая функция, после чего приложение ждет ответа от пользователя через консоль
- GUI-приложение
  - главная **нить** все время ждет новых **событий**, большинство из которых связано с действиями пользователя, произошедшие события обрабатываются
  - длительные действия выполняются в других нитях
  - приложение может не иметь консоли

Схема работы GUI-приложения



События

- Происходят во внешней среде (то есть, в операционной системе)
- Могут быть связаны
  - с периферийными устройствами (мышь, клавиатура)
  - с изменением состояния одного из компонентов приложения
  - с изменением состояния рабочего стола
  - с изменением состояния нитей, таймеров и других составляющих приложения
  - …
- Могут быть обработаны приложением

Поддержка GUI для языка Java

- Через peer-интерфейсы
  - метод на языке Java → функция API
  - графические компоненты в каждой оболочке выглядят по-своему
- Собственные компоненты Java
  - не зависят от конкретного API (реализация через JVM)
  - графические компоненты выглядят в соответствии с выбранным стилем

Библиотеки компонентов

- Компоненты
  - тяжелые (heavyweight) - используют peer-интерфейсы
  - легкие (lightweight) - самостоятельные
- Библиотека JFC (Java Foundation Classes)
  - AWT (Abstract Window Toolkit), ~1995
  - Swing - библиотека легких компонентов, ~1997
  - Java2D - средства рисования
  - DnD - средства Drag and Drop
  - …

Другие популярные GUI-библиотеки для Java

- JavaFX (позиционируется как будущая замена Swing) – 2008 год (1.0), 2014 год (8.0), входит в состав JRE / JDK с версии 8.0 (Java SE 8). **NB: requires Java 8!**
- Qt Jambi (wrapper for Qt) – 2015 год (4.8.7) (по-видимому появилась одновременно с Qt)
- SWT (standard widget toolkit, part of Eclipse IDE) – 1995 год (1.0), 2014 год (4.4)
- JavaServer Faces (user interfaces for **web applications only**, part of Java EE) – 2004 год (1.0), 2015 год (2.2)
- Spring Framework (Java SE / EE) and its MVC (**web applications only**) – 2004 год (1.0), 2015 год (4.2)

Примеры с компонентами AWT/Swing

- Label - текстовая строка
- Button - клавиша
- CheckBox - пункт выбора
- ComboBox - выпадающий список
- ListBox - постоянный список

Примеры с компонентами AWT/Swing

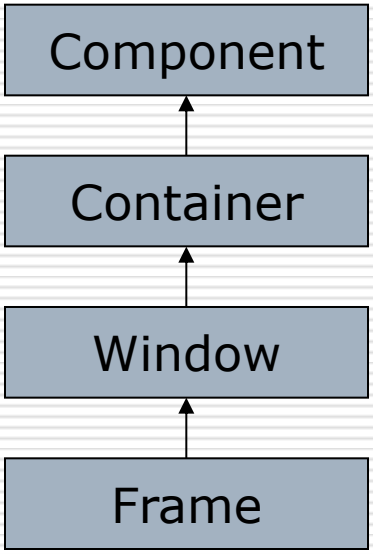
- Обратите внимание на отличия компонентов AWT и Swing – см. примеры

Контейнеры

- Контейнер - это компонент, который может содержать другие компоненты
  - окно
  - панель
  - диалог
  - …

Иерархия

- Контейнер является компонентом (Container extends Component)
- Окно является контейнером (Window extends Container)
- Фрейм является окном (с рамкой) (Frame extends Window)



Простейшее приложение на основе AWT

- Все, что требуется сделать - создать окно *Frame*, задать его размеры *setSize*, отобразить его на экране *setVisible* и создать обработчик закрытия окна *addWindowListener*
- Логика создания нити обработки событий зашита внутрь библиотеки

Простейшее приложение на основе AWT

```
public class MainFrame extends Frame {
    MainFrame(String s) {
        super(s);
        setSize(400, 400);
        setVisible(true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e){
                System.exit(0);
            }
        });
    }
    public static void main(String[] args) {
        new MainFrame("Приложение AWT");
    }
}
```

Простейшее приложение на основе Swing

- Все делается примерно аналогично
- Вместо класса *Frame* (AWT) используется класс *JFrame* (Swing)

Простейшее приложение на основе Swing

```
public class MainFrame extends JFrame {
    MainFrame(String s) {
        super(s);
        setSize(400, 400);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new MainFrame("Приложение Swing");
            }
        });
    }
}
```

Вызов конструктора фрейма

- Создателями AWT/Swing **рекомендуется** вызывать конструктор главного фрейма **в нити обработки сообщений**
- Для этой цели служит метод SwingUtilities.invokeLater(runnable);
- Runnable – интерфейс "запускаемый" с одним методом run

Графический редактор NetBeans

- Позволяет создать контейнер (окно, панель, диалог) и разместить на нем компоненты
- Контейнеру в графическом редакторе (файл *.form) соответствует класс, автоматически генерируемый на языке Java
- В отличие от MFC, никаких скриптов (rc) не создается
- Демонстрация графического редактора

Графический редактор IDEA

- Цели те же
- Также пара файлов -- *.form / *.java
- Два режима – binary class files (default) / Java source code (Settings → GUI Designer)
  - Binary: в Java-файл помещаются только объявления компонентов, код генерируется в бинарном виде при компиляции
  - Source: генерируется также метод *$$$setupUI$$$*, который выполняет необходимые операции по размещению компонентов (требует наличия в проекте библиотеки rt_forms, можно взять из Maven)
- Автоматическое создание метода main()

Принципы отображения графики

- Отображение содержимого окна (контейнера) осуществляется в методе *paint*
- Вывод графических примитивов осуществляется через классы управления графическим контекстом (Graphics, Graphics2D, аналоги QPainter в Qt или CDC в MFC)

Метод paint

- В результате действий пользователя окно может полностью или частично скрываться и затем открываться снова
- Поэтому окно должно иметь возможность отобразить свое содержимое в любой момент работы приложения

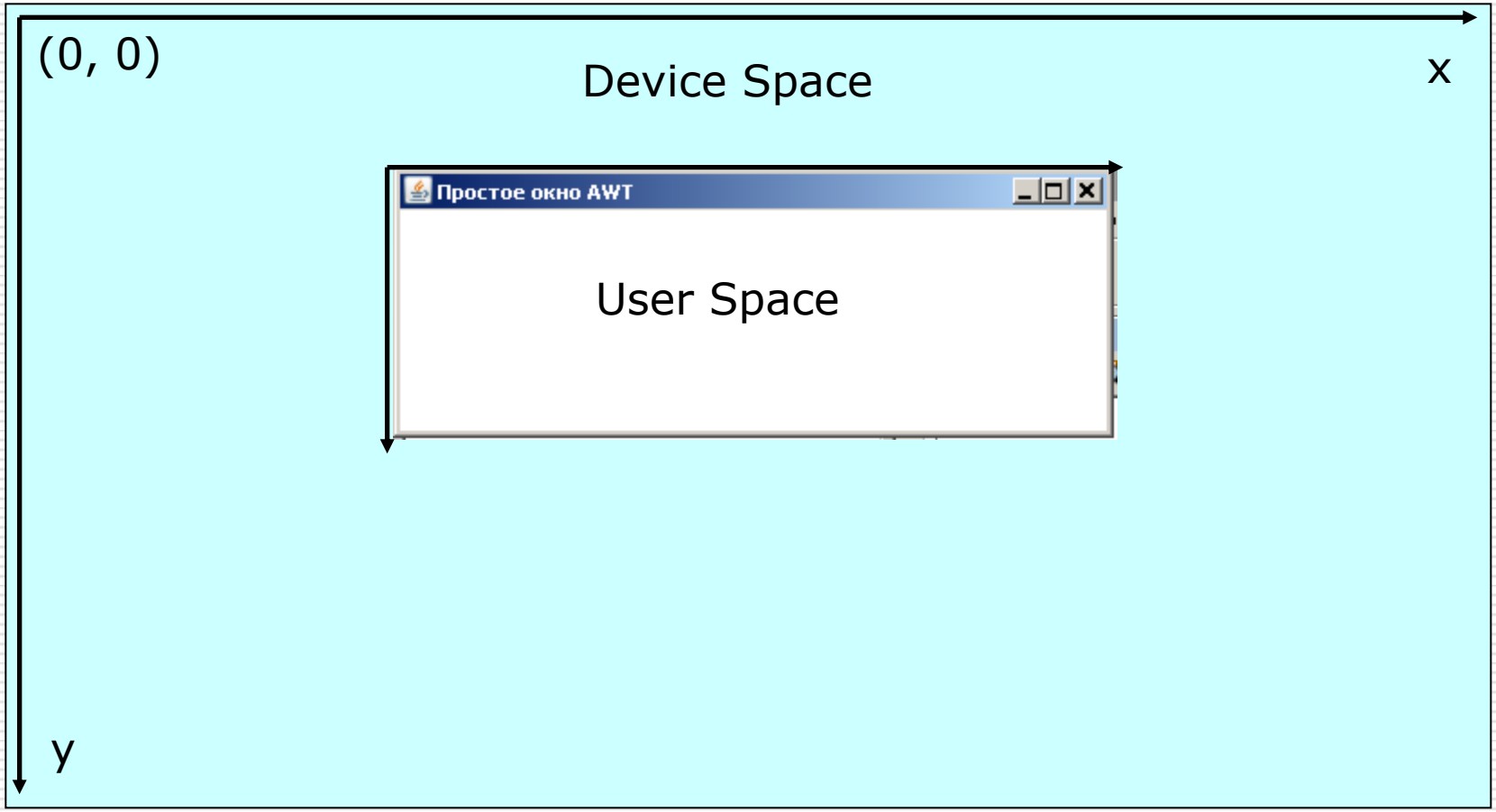
Метод paint

- В тот момент, когда графическая оболочка считает, что окно должно быть перерисовано, вызывается его метод *paint*
- Программа может сама вызвать перерисовку своего окна целиком или частично, вызвав метод *repaint*
  - repaint(); // или
  - repaint(x, y, width, height);

Метод paint

- В этом методе окно всегда должно быть перерисовано **полностью**
- Фактически, отрисовка осуществляется не на экране, а в памяти ЭВМ
- После этого область, которую необходимо перерисовать, копируется из памяти на экран

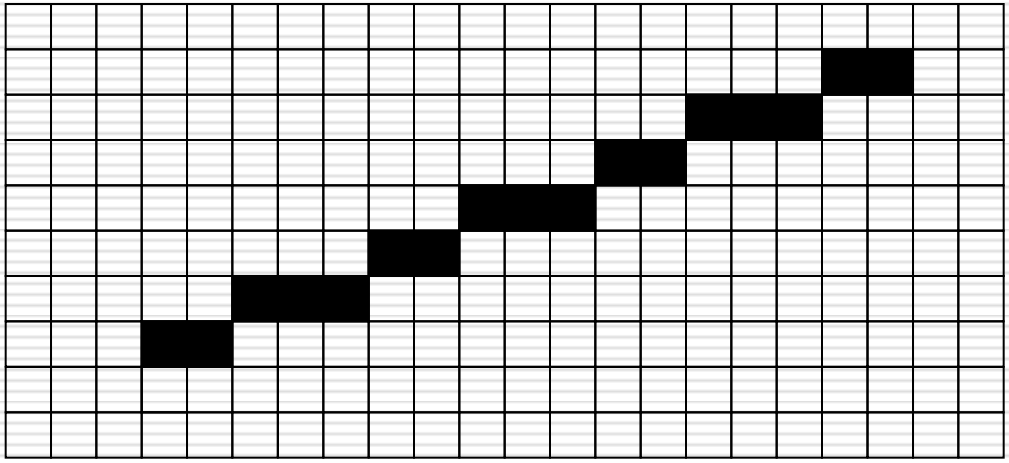
Координатная сетка



Пиксели

- Пиксел (pixel) - атомарный элемент графического изображения, квадрат с маленькими сторонами
- Каждый пиксел может иметь свой цвет
- Пикселы имеют целочисленные координаты:
  $0<=x<X_{max}$, $0<=y<Y_{max}$
- Все элементы изображения складываются из пикселей

Пример - построение линии



Цвета

- Цвет пикселя задается интенсивностями красного, зеленого и синего цвета, в диапазоне от 0 до 255
- int r=0, g=255, b=255;
- Color color = new Color(r, g, b);

Прозрачность (alpha)

- Для цвета может быть задана прозрачность (alpha) в диапазоне от 0 до 255. 255 - непрозрачный цвет, 0 - полностью прозрачный цвет (не будет виден)
- Color color = new Color(r, g, b, alpha);
- По умолчанию цвета непрозрачны

Выбор цвета

- Текущий цвет, используемый при рисовании, выбирается отдельным методом *setColor* класса *Graphics*
- Кроме этого, может быть выбран цвет фона окна, для этого используется метод *setBackground* класса *Component*

Основные примитивы

- drawLine(x, y, width, height) - рисование линии
- drawRect(x, y, width, height) - рисование прямоугольника
- drawOval(x, y, width, height) - рисование овала
- drawArc(x, y, width, height, start, length) - рисование дуги

Основные примитивы

- fillRect - закрашенный прямоугольник
- fillRoundRect - закрашенный закругленный прямоугольник
- fillOval - закрашенный овал

Вывод текста

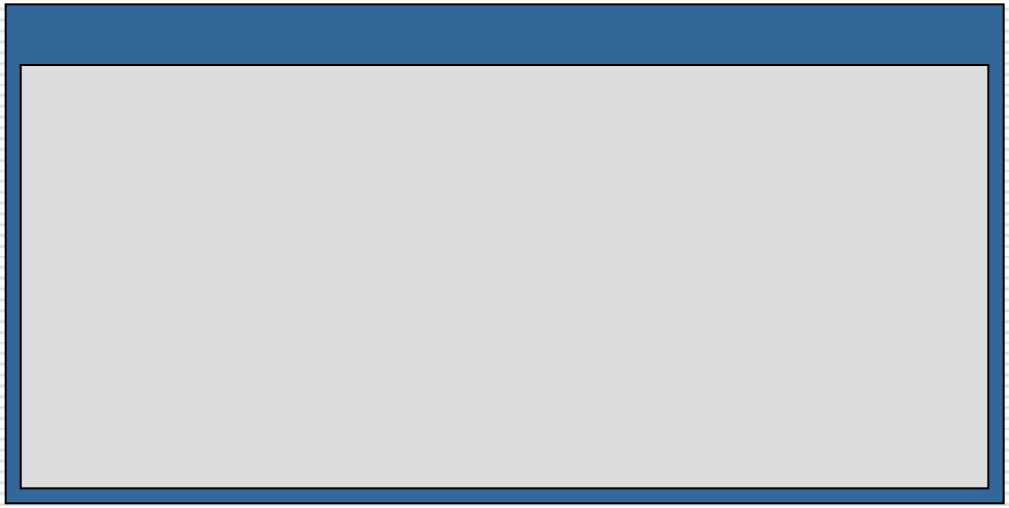
- Необходимо создать шрифт
  Font font = new Font (
  "Serif", Font.ITALIC, 24);
- Затем его выбрать *setFont(font)*
- Затем вызвать один из методов отображения текста, например, *drawString*(string, x, y);

Пример с примитивами AWT и Swing

- См.

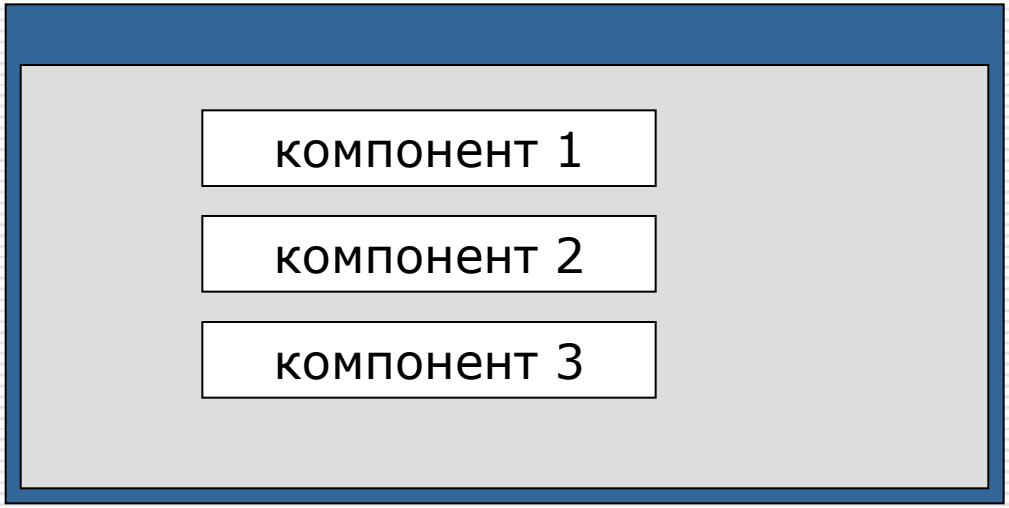
Устройство фрейма Swing

- Фрейм Swing, в отличие от фрейма AWT, содержит внутри себя еще один контейнер - так называемый contentPane



Устройство фрейма Swing

- Все компоненты, которые добавляются во фрейм Swing, на самом деле добавляются в его contentPane

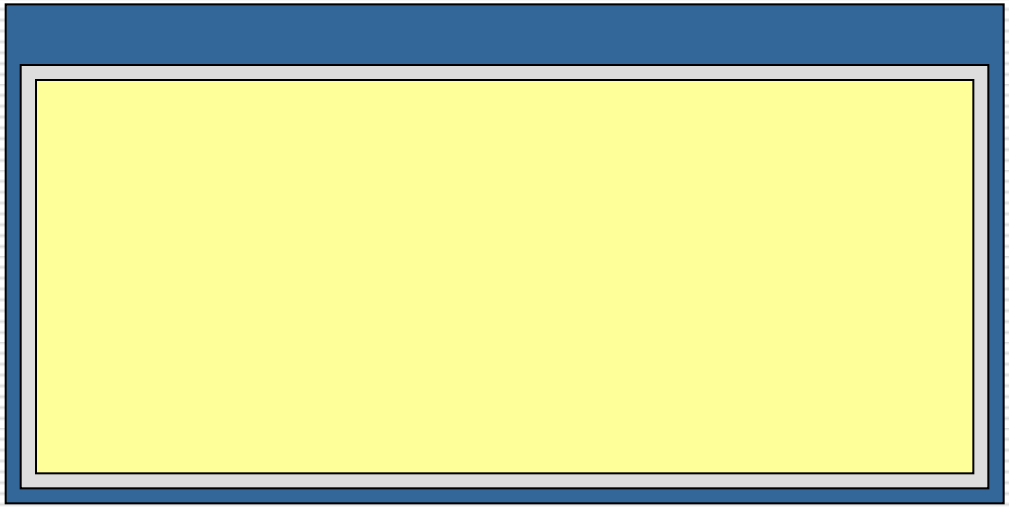


Перерисовка фрейма Swing

- Если метод *paint* фрейма Swing переопределен, то перерисовка осуществляется на основном фрейме
- Попытка изменить фон вызовом метода *setBackground* также изменяет фон основного фрейма
- В некоторых случаях содержимое *contentPane* затирает собой на экране содержимое основного фрейма

Перерисовка фрейма Swing

- Метод paint для контейнера contentPane переопределить мы не можем
- Наиболее разумно добавить во фрейм еще один контейнер - JPanel



Добавление панели во фрейм

```
MainFrame(String s) {
    super(s);
    setSize(600, 400);
    JPanel panel = new MainPanel();
    panel.setBackground(Color.CYAN);
    // or this.setContentPane(panel)
    this.add(panel);
    setVisible(true);
    setDefaultCloseOperation(
        JFrame.EXIT_ON_CLOSE);
}
```

Расширение класса «панель»

```
public class MainPanel extends JPanel {
    public MainPanel() {
        super();
    }
    public void paint(Graphics g) {
        super.paint(g);
        g.setColor(Color.BLACK);
        g.drawLine(0, 0, getWidth()-1,
                   getHeight()-1);
        g.drawLine(getWidth()-1, 0, 0,
                   getHeight()-1);
    }
}
```

Таким образом

- Всю перерисовку во фреймах Swing рекомендуется делать на панели
- Если проект включает как отрисованную часть, так и часть с компонентами, рекомендуется размещать их на отдельных панелях

Дополнения к предыдущей лекции

- Примеры с апплетами

Апплет посложнее

- Панель, по которой можно перемещать круглые фишки
- См. пример

HTML-файл

```
<html>
 <title>Страница со сложным апплетом</title>
 <body>
  <p align=right>
   Здесь находится сложный апплет
  </p>
  <applet code="boardapplet.BoardApplet.class"
    width="800" height="300" align=right
    archive="BoardApplet.jar">
  </applet>
 </body>
</html>
```

Протоколирование

```
static private String logStr = "";
static private int counter = 0;
private int id = 0;

private void log(String msg) {
    logStr = logStr + msg + id + " ";
    if (statusLabel != null)
        statusLabel.setText(logStr);
}
```

Функция init

```
public void init() {
    counter++; // Увеличение счетчика апплетов
    id=counter; // Установка собственного номера
    log("init");
    boardPanel = new BoardPanel();
    statusLabel = new JLabel();
    statusLabel.setText(logStr);
    statusLabel.setBorder(
        new LineBorder(Color.BLACK, 1));
    add(boardPanel, BorderLayout.CENTER);
    add(statusLabel, BorderLayout.SOUTH);
    log("^" + boardPanel.getWidth() + "x" +
            boardPanel.getHeight() + "^");
}
```

Функция start

```
public void start() {
    // ...
    // Почему не в init()?
    for (int i=0; i<4; i++)
        boardPanel.addChip();
    repaint();
}
```

Выбор мышью

```
private void onMousePressed(MouseEvent e) {
    if (e.getButton()!=MouseEvent.BUTTON1)
        return;
    for (Chip chip: chips) {
        if (chip.contains(e.getPoint())) {
            chosenChip = chip;
            return;
        }
    }
    chosenChip = null;
}
```

Перетаскивание мышью

```
private void onMouseDragged(MouseEvent e) {
    if (chosenChip==null) return;
    Point p=e.getPoint();
    Point oldCenter = chosenChip.getCenter(); // Старое
    if (!chosenChip.moveTo(p)) return; // Было ли движение
    if (!chosenChip.inBorder(getBounds())) { // Проверка
        chosenChip.moveTo(oldCenter);
        return;
    }
    for (Chip chip: chips) { // Проверка пересечений
        if (chosenChip!=chip && chip.crossed(chosenChip)) {
            chosenChip.moveTo(oldCenter);
            return;
    }  }
    repaint();
}
```

Инициализация всплывающего меню

```
private void initMenu() {
    popupMenu = new JPopupMenu();
    addChipMenu =
      new JMenuItem("Добавить фишку");
    addChipMenu.addActionListener(…);
    popupMenu.add(addChipMenu);
    clearChipsMenu =
      new JMenuItem("Очистить поле");
    clearChipsMenu.
      addActionListener(…);
    popupMenu.add(clearChipsMenu);
}
```

Активация всплывающего меню

```
@Override
public void processMouseEvent
        (MouseEvent e) {
    if (e.isPopupTrigger())
        popupMenu.show(
                e.getComponent(),
                e.getX(), e.getY());
    else
        super.processMouseEvent(e);
}
```

Итоги

- Рассмотрено
  - Принципы построения GUI
  - Принципы AWT / Swing
  - Графические редакторы форм
  - Простые приложения
  - Пример с апплетом
- Далее
  - Таймеры, графика, мышь, клавиатура...