

Программирование на Java

8. Отрисовка

Глухих Михаил Игоревич
mailto: glukhikh@mail.ru

За занавесом

► Дополнительное чтение:

- <http://www.oracle.com/technetwork/java/painting-140037.html>
- <http://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html>
- <https://docs.oracle.com/javase/tutorial/extra/fullscreen/doublebuf.html>
- <http://docs.oracle.com/javafx/2/scenegraph/jfxpub-scenegraph.htm>

Выдержки (weight)

- ▶ heavyweight component: всегда имеет привязанное к нему окно ОС
 - Любой наследник Component, но не JComponent

Выдержки (weight)

- ▶ heavyweight component: всегда имеет привязанное к нему окно ОС
 - Любой наследник Component, но не JComponent
- ▶ lightweight component: переиспользует окно ОС ближайшего «тяжелого» родителя
 - Любой наследник JComponent
 - NB: JFrame НЕ наследник JComponent (!)

Выдержки (painting)

- ▶ System-triggered
- ▶ Application-triggered (repaint)

Выдержки (painting)

- ▶ System-triggered
- ▶ Application-triggered (repaint)
- ▶ Lightweight
 - Container.paint calls paint of its children
 - Container.paintComponents (!)
 - NB: don't forget to call super.paint() (!)

Выдержки (Swing Containers)

- ▶ NB: all heavy-weight (!)
 - Container
 - Window
 - JWindow
 - Frame
 - JFrame
 - Dialog
 - JDialog
 - Panel
 - Applet
 - JApplet

Выдержки (Swing JFrame)

- ▶ JFrame состоит из (NB: неточно!)
 - JMenuBar
 - Content Pane

Выдержки (Swing JFrame)

- ▶ JFrame состоит из
 - JRootPane (! extends JComponent !)
 - Такая же панель входит в JWindow / JDialog / JApplet

Выдержки (Swing JFrame)

- ▶ JFrame состоит из
 - JRootPane (! extends JComponent !) состоит из
 - glassPane (перехватывает движения мыши)
 - JLayeredPane состоит из
 - JMenuBar (верхняя часть)
 - Content Pane (нижняя часть)

Выдержки (painting in Swing)

- ▶ Двойная буферизация
 - Paint to image, then copy to screen

Выдержки (painting in Swing)

- ▶ Двойная буферизация
 - Paint to image, then copy to screen
- ▶ `JComponent.paint`
 - `paintComponent` (NB: to override!)
 - `paintBorder`
 - `paintChildren`

Выдержки (JavaFX)

- ▶ Отрисовка с помощью scene graph (tree)
 - Включает в себя узлы (Node, Parent) для отрисовки
 - Графические примитивы тоже являются узлами

Движущиеся изображения

- ▶ Один из способов создания – использование таймера (Timer)
- ▶ Таймер – объект, генерирующий события периодически, с равным интервалом
- ▶ Необходимо при возникновении события таймера изменить изображение на фрейме

Последовательность действий

► Создать «слушатель» – ActionListener

```
timerListener = new ActionListener() {  
    public void actionPerformed(  
       (ActionEvent e) {  
        // Какие-то изменения  
        repaint();  
    }  
};
```

Последовательность действий

► Или короче

```
timerListener = e -> {  
    // Какие-то действия  
    repaint();  
};
```


Последовательность действий

- ▶ Создать и запустить таймер

```
timer = new Timer(20, timerListener);  
timer.start();
```

- ▶ Используется таймер `javax.swing.Timer`
- ▶ NB: существует ещё `java.util.Timer`,
`sun.misc.Timer`, ...

Пример

- ▶ Необходимо написать программу, создающую окно с изображением летящего шарика
- ▶ При столкновении с границами окна шарик должен отражаться от них

Проектирование

- ▶ Фрейм, в нём *панель*
- ▶ Панель
 - *таймер*
 - *слушатель*
 - *шарик*
 - **Отрисовка шариков**
- ▶ Шарик
 - *Положение, Размеры, Скорость*
 - *Цвет*
 - **передвижение**

Класс «фрейм»

```
public class MainFrame extends JFrame {  
    MainFrame(String s) {  
        super(s);  
        setSize(300, 200);  
        this.setContentPane(new MainPanel());  
        setVisible(true);  
        setDefaultCloseOperation(  
            WindowConstants.EXIT_ON_CLOSE);  
    }  
    public static void main(String[] args) {  
        SwingUtilities.invokeLater(  
            () -> MainFrame("Заставка с шариком"));  
    }  
}
```

Класс «панель», конструктор

```
class MainPanel extends JPanel {  
    public MainPanel() {  
        setBackground(Color.BLACK);  
        ball = new Ball(50, 150,  
                        1, 2, 10, Color.RED);  
        ActionListener timerListener = e -> {  
            ball.step(getWidth(), getHeight());  
            repaint();  
        };  
        Timer timer = new Timer(  
            20, timerListener);  
        timer.start();  
    }  
}
```

Класс «панель», шарик, перерисовка

```
class MainPanel extends JPanel {  
    private Ball ball;  
    public void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        paintBall(g, ball);  
    }  
    private void paintBall(Graphics g,  
                             Ball b) {  
        g.setColor(b.getColor());  
        int radius = b.getRadius();  
        g.fillOval(b.getX() - radius,  
                   b.getY() - radius,  
                   2 * radius, 2 * radius);  
    }  
}
```

Класс «шарик», Котлин

```
class Ball(var x: Int, var y: Int,  
           private var dx: Int,  
           private var dy: Int,  
           val radius: Int,  
           val color: Color) { ... }
```

Класс «шарик», шаг

- ▶ См. `part3.painting.ball.Ball.step`

Класс Graphics2D

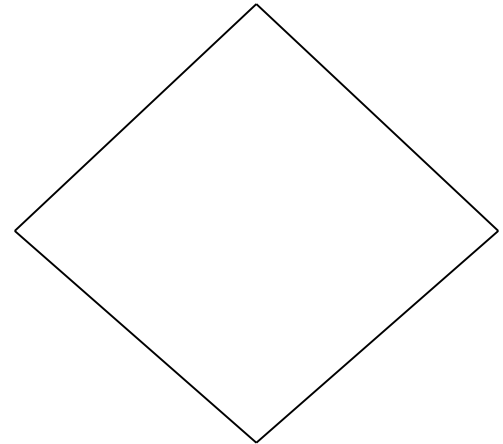
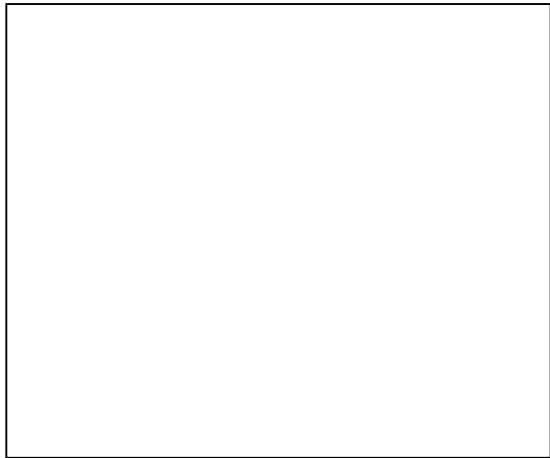
- ▶ Введен в Java 2D (1.2?) и является расширением (extends) класса Graphics
- ▶ Новые возможности
 - преобразование координат

Преобразование координат (аффинное)

- ▶ Осуществляется преобразование вида $(x, y) \rightarrow (ax + cy + e, bx + dy + f)$
- ▶ Используется класс `AffineTransform`
- ▶ Возможные преобразования:
 - общий вид: `new AffineTransform(a,b,c,d,e,f);`
 - поворот: `getRotateInstance(angle,x,y);`
 - расширение/сжатие: `getScaleInstance(sx, sy);`
 - сдвиг: `getTranslateInstance(tx, ty);`
 - конкатенация: `concatenate(at);`
- ▶ Выбор: `setTransform(at);`

Преобразование координат

- ▶ Поворот на 45 градусов со сжатием



Пример (part3.painting.g2d)

```
public void paintComponent(Graphics g) {  
    super.paintComponent(g);  
    Graphics2D g2d = (Graphics2D)g;  
    AffineTransform at =  
        AffineTransform.getRotateInstance(  
            Math.PI/6.0, 100, 100);  
    at.concatenate(  
        AffineTransform.getScaleInstance(1.0, 0.5));  
    g2d.setTransform(at);  
    g2d.setColor(new Color(0, 128, 0));  
    g2d.setFont(new Font("Serif", Font.ITALIC, 24));  
    g2d.drawString("Графические примитивы", 100, 50);  
    g2d.drawRect(75, 25, 300, 50);  
}
```

Класс Graphics2D

- ▶ Новые возможности
 - преобразование координат
 - введены способы вычерчивания (перья) и способы заливки (кисти)

Перья

- ▶ Задают способ вычерчивания линий
- ▶ Используется интерфейс `Stroke` и его реализация – класс `BasicStroke`
- ▶ Для выбора пера используется метод `setStroke(stroke);`

Свойства перьев

- ▶ толщина (width)
- ▶ конец линии (cap) – закругленный, квадратный
- ▶ сопряжение линий (join) – дуга, промежуточный отрезок, стык
- ▶ длина штрихов и промежутков (dash)

Примеры перьев

- ▶ **Перо толщиной 10**

```
g2d.setStroke(new BasicStroke(10));
```

- ▶ **Перо толщиной 10 с закруглением на концах и отрезками на стыках**

```
g2d.setStroke(new BasicStroke(10,  
    BasicStroke.CAP_ROUND,  
    BasicStroke.JOIN_BEVEL));
```

- ▶ **Штрихпунктирное перо толщиной 5**

```
g2d.setStroke(new BasicStroke(  
    5, BasicStroke.CAP_BUTT,  
    BasicStroke.JOIN_MITER, 1,  
    new float[]{10, 5, 20, 5, 10, 5}, 0));
```

- ▶ **И так далее**

Кисти

- ▶ Задают способ заливки фигур
- ▶ Используется интерфейс Paint
 - реализация Color (сплошная заливка)
 - реализация GradientPaint (градиентная заливка)
 - реализация TexturePaint (текстурная заливка)
- ▶ Для выбора кисти используется метод `setPaint(p)`

Градиентная заливка

```
new GradientPaint(  
    100, 100, Color.RED,  
    300, 300, Color.BLUE);
```

- ▶ В точке (100, 100) красный цвет
- ▶ В точке (300, 300) синий цвет
- ▶ Между ними – смешанный цвет

Класс Graphics2D

- ▶ Новые возможности
 - преобразование координат
 - введены способы вычерчивания (перья) и способы заливки (кисти)
 - работа с изображениями

Работа с готовыми изображениями

▶ Чтение иконы из файла

```
ImageIcon icon =  
    new ImageIcon("shield.gif");  
Image image = icon.getImage();
```

▶ Вывод рисунка в окно

```
g2d.drawImage(  
    icon.getImage(), x, y, this);
```

Класс Graphics2D

- ▶ Новые возможности
 - преобразование координат
 - введены способы вычерчивания (перья) и способы заливки (кисти)
 - работа с изображениями
 - введен интерфейс для рисования различных фигур

Фигуры

- ▶ Для рисования фигур используется интерфейс Shape, методы draw(shape) и fill(shape)
- ▶ Преимущество фигур над стандартными примитивами состоит в том, что их можно подготовить заранее и потом нарисовать путем вызова одного метода (draw или fill)
- ▶ Существующие готовые фигуры

```
Rectangle2D r2d = new Rectangle2D.Double(  
    0.0, 0.0, 50.0, 100.0);
```

```
Line2D l2d = new Line2D.Float(  
    0.0F, 0.0F, 100.0F, 50.0F);
```

Ellipse2D – по аналогии

Arc2D – по аналогии

Создание своих фигур

- ▶ Один из существующих способов – использование класса `GeneralPath`
- ▶ Например (стрелка вверх)

```
GeneralPath shape = new GeneralPath();  
shape.moveTo(-1.0, 0.0);  
shape.lineTo(-1.0, -90.0);  
shape.lineTo(-10.0, -80.0);  
shape.lineTo(0.0, -100.0);  
shape.lineTo(10.0, -80.0);  
shape.lineTo(1.0, -90.0);  
shape.lineTo(1.0, 0.0);  
shape.closePath();
```

Создание своих изображений

- ▶ Один из существующих способов – использование класса `BufferedImage`

- ▶ Например (эллипс с каймой)

```
BufferedImage image = new BufferedImage(
    100, 100, BufferedImage.TYPE_INT_ARGB);
Graphics2D g2d = image.createGraphics();
g2d.setBackground(new Color(0, 0, 0, 0));
Ellipse2D e2d = new Ellipse2D.Double(
    0.0, 0.0, 100.0, 100.0);
g2d.setColor(Color.RED);
g2d.fill(e2d);
g2d.setColor(Color.BLUE);
g2d.draw(e2d);
```

- ▶ Созданное изображение может быть нарисовано (`drawImage`)

Демонстрация основных методов Graphics2D

- ▶ См. `part3.painting.g2d`
- ▶ Также см. `part3.simple.primitives.java`

Более реальный пример: часы

- ▶ Необходимо написать программу, изображающую на экране окно с часами (циферблат и стрелки)
- ▶ Время на часах должно изменяться в соответствии с системным временем

Проектирование

- ▶ Фрейм
 - *панель*
- ▶ Панель
 - *таймер, слушатель*
 - *циферблат, стрелки*
 - **отрисовка**
- ▶ Циферблат
 - *изображение*
- ▶ Стрелка
 - *фигура стрелки, цвет*

Класс «панель», поля

```
public class ClockPanel extends JPanel {  
    static private final double BASE_SIZE = 1000.0;  
    private final ClockFace face = new ClockFace(BASE_SIZE,  
        Color.LIGHT_GRAY, Color.BLACK);  
    private final ClockHand hourHand = new ClockHand(  
        0.6*BASE_SIZE, Color.RED);  
    private final ClockHand minuteHand = new ClockHand(  
        0.8*BASE_SIZE, Color.BLUE);  
    private final ClockHand secondHand = new ClockHand(  
        BASE_SIZE, Color.GREEN);  
}
```

Класс «панель», конструктор, время

```
public ClockPanel() {  
    super();  
    setBackground(Color.DARK_GRAY);  
    Timer timer = new Timer(1000, e -> repaint());  
    timer.start();  
}  
@Override  
public void paintComponent(Graphics g) {  
    super.paintComponent(g);  
    final Calendar calendar = Calendar.getInstance();  
    final int hour = calendar.get(Calendar.HOUR);  
    final int minute = calendar.get(Calendar.MINUTE);  
    final int second = calendar.get(Calendar.SECOND);  
    final double hourAngle = hour * Math.PI / 6;  
    final double minuteAngle = minute * Math.PI / 30;  
    final double secondAngle = second * Math.PI / 30;  
}
```

Класс «панель», отрисовка

```
final int width = getWidth();
final int height = getHeight();
final int size = width < height ? width : height;
Graphics2D g2d = (Graphics2D) g;
// Циферблат
g2d.setTransform(AffineTransform.getScaleInstance(
    size/BASE_SIZE, size/BASE_SIZE));
g2d.drawImage(face.getImage(), 0, 0, null);
// Базовое преобразование для стрелок
final AffineTransform base = new AffineTransform();
base.translate(0.5 * size, 0.5 * size);
base.scale(0.5 * size / BASE_SIZE,
    0.5 * size / BASE_SIZE);
// Рисование стрелок
hourHand.paint(g2d, base, hourAngle);
minuteHand.paint(g2d, base, minuteAngle);
secondHand.paint(g2d, base, secondAngle);
```

Класс «циферблат», конструктор

```
public class ClockFace {  
    private final BufferedImage image;  
    public ClockFace(final double size,  
                     final Color back, final Color marks) {  
        image = new BufferedImage((int)size, (int)size,  
                                   BufferedImage.TYPE_INT_ARGB);  
        Graphics2D g2d = image.createGraphics();  
        g2d.setBackground(new Color(0, 0, 0, 255));  
        g2d.setColor(back);  
        Ellipse2D ellipse = new Ellipse2D.Double(  
            0.0, 0.0, size, size);  
        g2d.fill(ellipse);  
        // ...  
    }  
}
```

Класс «циферблат», конструктор

```
public class ClockFace {  
    private final BufferedImage image;  
    public ClockFace(final double size,  
                     final Color back, final Color marks) {  
        // ...  
        g2d.setColor(marks);  
        g2d.setStroke(new BasicStroke(5));  
        Line2D line = new Line2D.Double(  
            0.5 * size, 0.1 * size, 0.5 * size, 0.05 * size);  
        for (int i = 0; i < 12; i++) {  
            final AffineTransform at =  
                AffineTransform.rotateInstance(  
                    (i+1) * Math.PI / 6,  
                    0.5 * size, 0.5 * size);  
            g2d.setTransform(at);  
            g2d.draw(line);  
        }  
    }  
}
```


Класс «стрелка», конструктор

```
public class ClockHand {  
    private final GeneralPath shape;  
    private final Color color;  
    public ClockHand(final double length,  
                     final Color color) {  
        this.color = color;  
        shape = new GeneralPath();  
        shape.moveTo(-0.01 * length, 0.0);  
        shape.lineTo(-0.01 * length, -0.9 * length);  
        shape.lineTo(-0.1 * length, -0.8 * length);  
        shape.lineTo(0.0, -1.0 * length);  
        shape.lineTo(0.1 * length, -0.8 * length);  
        shape.lineTo(0.01 * length, -0.9 * length);  
        shape.lineTo(0.01 * length, 0.0);  
        shape.closePath();  
    }  
}
```

Рисование стрелки

```
public class ClockHand {  
    public void paint(final Graphics2D g2d,  
        final AffineTransform base,  
        final double angle) {  
        g2d.setColor(color);  
        final AffineTransform at = new  
            AffineTransform(base);  
        at.rotate(angle);  
        g2d.setTransform(at);  
        g2d.fill(shape);  
    }  
}
```

Демонстрация работы

- ▶ См. пример `part3.painting.clock`

Демонстрация работы

- ▶ См. пример `part3.painting.clock`
- ▶ Включение / выключение анти-алиасинга:
 - `g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);`

ИТОГИ

- ▶ Рассмотрены
 - Основные принципы отрисовки
 - Таймеры
 - Graphics2D
- ▶ Далее
 - События
 - Компоненты
 - Менеджеры размещения
 - Редактор форм