Karlsruher Institut für Technologie

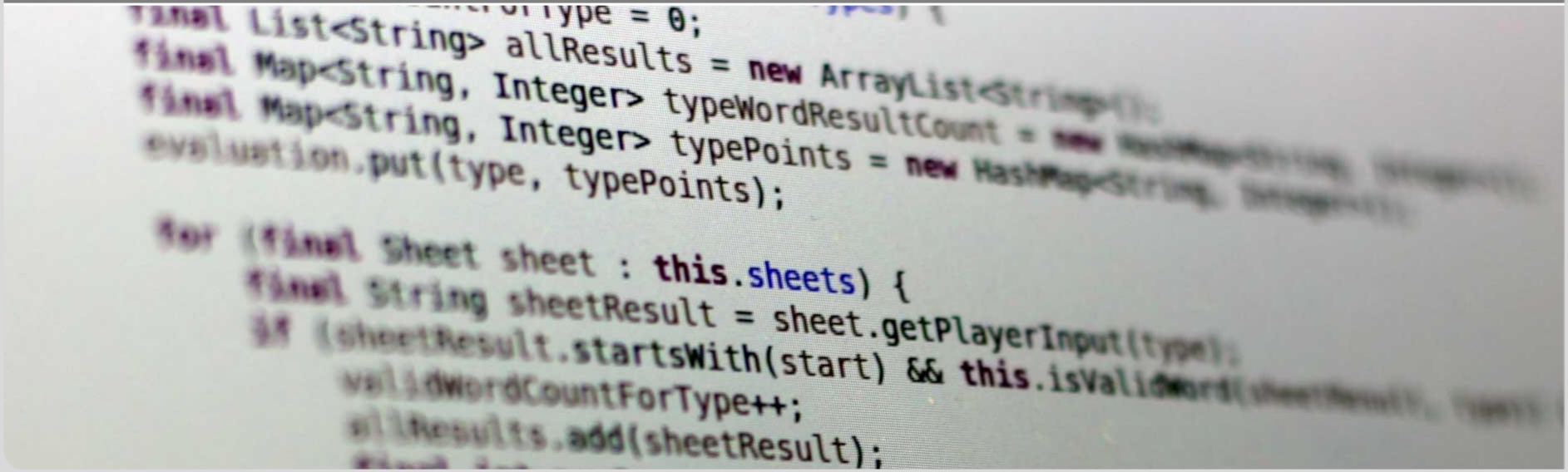
DHBW
Duale Hochschule
Baden-Württemberg

Programmieren II

Zeichnen in Swing-Komponenten

Heusch 18
Ratz 16.1

Institut für Automation und angewandte Informatik



Zeichnen in Swing-Komponenten (1)

- Eine Swing-Komponente kann nicht nur eine grafisch vorgefertigte Komponente sein (z.B. ein Button).
  Sie kann auch selbst grafisch gestaltet werden.
- In diesem Kapitel wird behandelt, wie man in einer Swing-Komponente eigene grafische Elemente zeichnen kann,
  indem man die Methode, die für das Zeichnen der Komponente zuständig ist, überschreibt.

- Für die Darstellung der einzelnen Swing-Komponenten ist der *Repaint-Manager* zuständig.
- Dieser veranlasst, dass für das erstmalige Erscheinen einer Komponente sowie bei Änderungen die *Darstellung* der Komponente aktiviert bzw. aktualisiert wird.

Zeichnen in Swing-Komponenten (2)

- Hierzu ruft der Repaint-Manager die von `Component` geerbte Instanzmethode
  `void` **`repaint`**`()`
  der Komponente auf.
- Die Methode `repaint()` bewirkt einen Aufruf der Methode
  `void` **`paint`**`(Graphics g)`
  der Komponente.
- Die `paint`-Methode ist die *Zeichenmethode* für die Darstellung der jeweiligen Komponente.

Zeichnen in Swing-Komponenten (3)

- Die `paint`-Methode wiederrum verteilt den Aufruf intern weiter auf die Methoden:
  - `protected void paintComponent(Graphics g)`
    zeichnet den eigentlichen Inhalt der Komponente
  - `protected void paintBorder(Graphics g)`
    zeichnet einen (optionalen) Rahmen
  - `protected void paintChildren(Graphics g)`
    zeichnet alle Unterkomponenten (nur bei Containern)
- Wenn die grafische Darstellung (der Inhalt) einer Komponente festgelegt oder verändert werden soll, muss die (geerbte) `paintComponent`-Methode der Komponente überschrieben werden.

Zeichnen in Swing-Komponenten (4)

- Der Parameter `Graphics g` der verschiedenen `paintXXX`-Methoden liefert einen Bezug zum grafischen Kontext), in den gezeichnet werden soll (Grafik-Fenster auf dem Ausgabegerät).
- Die Klasse `Graphics` enthält verschiedene Methoden zum Zeichnen, wie z.B. `drawString()` zum „Zeichnen“ einer Zeichenkette und `setColor()` zum Setzen der aktuellen Farbe.

Erstes Beispiel (Version 1)

```
import java.awt.Graphics;
import javax.swing.*;

public class FirstPaint1 extends JComponent {

    @Override
    protected void paintComponent(Graphics g) {
        g.drawString("Hello World", 25, 50);
    }

    public static void main(String[] args) {
        JFrame f = new JFrame();
        f.setTitle("First Paint");
        f.add(new FirstPaint1());
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(300, 200);
        f.setVisible(true);
    }

}
```

Erstes Beispiel (Version 2)

```java
import java.awt.Graphics;
import javax.swing.JComponent;

public class DrawComponent extends JComponent {
    @Override
    protected void paintComponent(Graphics g) {
        g.drawString("Hello World", 25, 50);
    }
}
```


```java
import javax.swing.JFrame;

public class FirstPaint2 extends JFrame {
    public FirstPaint2() {
        this.setTitle("First Paint 2");
        this.setSize(300, 200);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.add(new DrawComponent());
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new FirstPaint2();
    }
}
```

Erstes Beispiel (Version 2a)

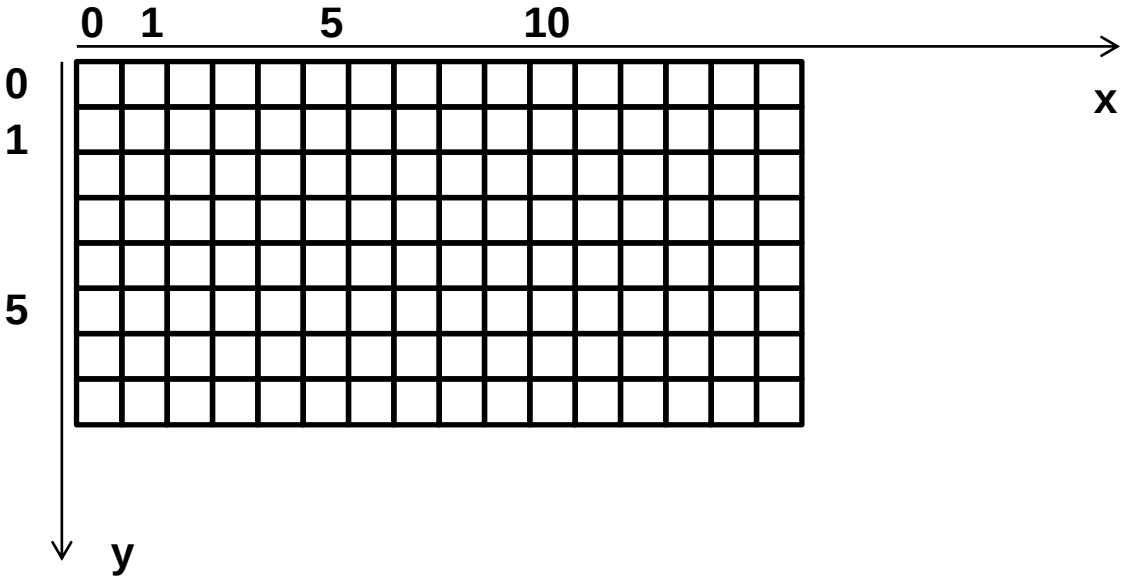
```
import java.awt.BorderLayout;
import javax.swing.*;

public class FirstPaint2a extends JFrame {

 public FirstPaint2a() {
    this.setLayout(new BorderLayout());
        this.setTitle("First Paint 2a");
        this.setSize(300, 200);
        JPanel panel = new JPanel();
        panel.add(new JButton("Links"));
        panel.add(new JButton("Rechts"));
        this.add(panel,BorderLayout.NORTH);
        this.add(new DrawComponent(),BorderLayout.CENTER);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        new FirstPaint2a();
    }

}
```

Erstes Beispiel (Version 3)

```
import java.awt.Graphics;
import javax.swing.JFrame;

public class FirstPaint3 extends JFrame {
    public FirstPaint3() {
        this.setTitle("First Paint 3"); this.setSize(300, 200);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }

    // Zeichnen direkt in JFrame (nicht in hinzugefügte Komponente)
    // daher direktes Überschreiben der Methode paint!
    @Override
    public void paint(Graphics g) {
        // super.paint(g);  // Darstellung enthaltener Komponenten
        g.clearRect(0,0,this.getWidth(),this.getHeight());
        g.drawString("Hello World", 25, 50);
    }

    public static void main(String[] args) {
        new FirstPaint3();
    }
}
```

Grafik-Koordinatensystem

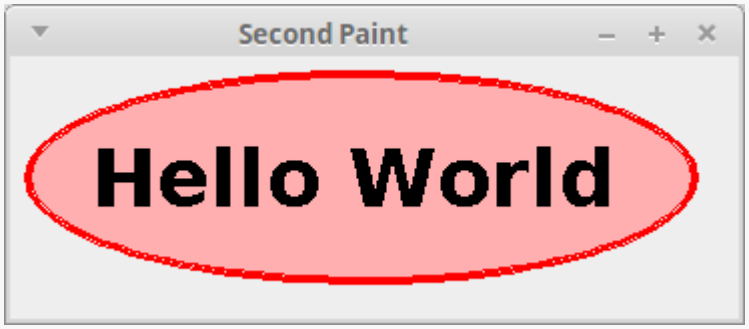
- Dem Grafiksystem liegt ein zweidimensionales Pixel-Koordinatensystem zugrunde



- Der Koordinatenursprung (0,0) liegt in der linken oberen Ecke.
- Die y-Achse verläuft von oben nach unten.

Eine Zeichnung mit Ovalen

```
import java.awt.*;
import javax.swing.*;

public class SecondPaint extends JComponent {
    @Override
    protected void paintComponent(Graphics g) {
        // The pink oval
        g.setColor(Color.PINK);
        g.fillOval(10, 10, 330, 100);
        g.setColor(Color.RED);
        g.drawOval(10, 10, 330, 100);
        g.drawOval(9, 9, 332, 102);
        g.drawOval(8, 8, 334, 104);
        g.drawOval(7, 7, 336, 106);
        // The text
        g.setColor(Color.BLACK);
        g.setFont(new Font("Helvetica", Font.BOLD, 40));
        g.drawString("Hello World", 40, 75);
    }
    public static void main(String[] args) {
        JFrame f = new JFrame();
        f.setTitle("Second Paint");
        f.add(new SecondPaint());
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(370, 160);
        f.setVisible(true);
    }
}
```



Einige Methoden der Klasse `Graphics` (1)

- Die Klasse `java.awt.Graphics` bietet zahlreiche Methoden für das Zeichnen von Grafikobjekten. Auswahl:
  - `void` **`setFont`**`(Font font)`
    Legt den Font für die folgenden Grafik-Operationen fest
  - `void` **`setColor`**`(Color c)`
    Legt die Farbe für die folgenden Grafik-Operationen fest
  - `void` **`drawString`**`(String str, int x, int y)`
    Zeichnet eine Zeichenkette. (`x`,`y`: linke untere Ecke des 1. Zeichens)
  - `void` **`drawLine`**`(int x1, int y1, int x2, int y2)`
    Zeichnet eine Linie
  - `void` **`drawRect`**`(int x, int y, int width, int height)`
    Zeichnet ein Rechteck. `x`, `y`: Koordinaten des linken oberen Ecks
    `width`, `height`: Breite und Höhe des Rechtecks (in Pixeln)
  - `void` **`fillRect`**`(…)`
    Zeichnet ein *gefülltes* Rechteck

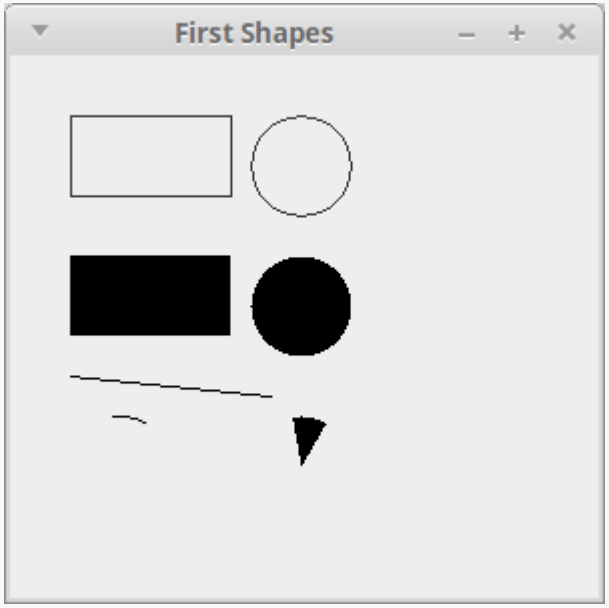
Einige Methoden der Klasse `Graphics` (2)

- `void` **`drawOval`**`(int x, int y, int width, int height)`
  Zeichnet ein Oval. Bedeutung der Argumente: wie oben
- `void` **`fillOval`**`(…)`
  Zeichnet ein *gefülltes* Oval.
- `void` **`drawArc`**`(int x, int y, int width, int height, int startAngle, int arcAngle)`
  Zeichnet einen Ellipsen- oder Kreisbogen. `startAngle`: Startwinkel, `arcAngle`: eigentlicher Winkel des Bogens (jeweils in Grad)
- `void` **`fillArc`**`(int x, int y, int width, int height, int startAngle, int arcAngle)`
  Zeichnet einen *gefüllten* Kreisbogen (Sektor).

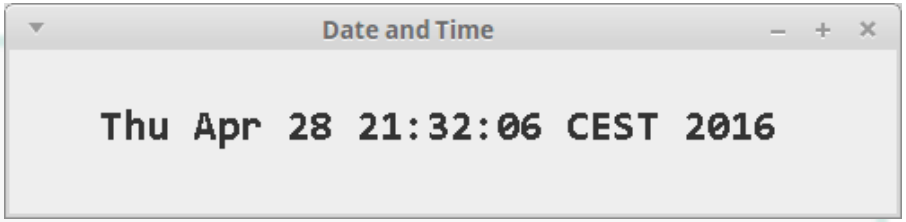
- Weitere Methoden später

Beispiel: Formen

```
import java.awt.*;
import javax.swing.*;

public class FirstShapes extends JComponent {
    @Override
    protected void paintComponent(Graphics g) {
        g.drawRect(30, 30, 80, 40);
        g.drawOval(120, 30, 50, 50);
        g.setColor(Color.BLACK);
        g.fillRect(30, 100, 80, 40);
        g.fillOval(120, 100, 50, 50);
        g.drawLine(30, 160, 130, 170);
        g.drawArc(30, 180, 50, 50, 60, 40);
        g.fillArc(120, 180, 50, 50, 60, 40);
    }

    public static void main(String[] args) {
        JFrame f = new JFrame();
        f.add(new FirstShapes());
          f.setTitle("First Shapes");
          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          f.setSize(300, 300); f.setVisible(true);
      }
}
```



Beispiel: Datum und Uhrzeit




```
import java.awt.*;
import java.util.Date;
import javax.swing.*;

public class DigitalClock extends JComponent {

    @Override
    protected void paintComponent(Graphics g) {
        g.setFont(new Font("Consolas", Font.BOLD, 24));
        Date theDate = new Date();
        g.drawString(theDate.toString(), 50, 50);
    }

    public static void main(String[] args) {
        JFrame f = new JFrame();
        f.add(new DigitalClock());
        f.setTitle("Date and Time");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(500, 120);
        f.setVisible(true);
    }

}
```

Bei jeder Ausgabe oder erneuten Ausgabe des Fensters wird `repaint()` aufgerufen, z.B. auch bei Änderung der Fenstergröße.

Einige weitere Methoden der Klasse `java.awt.Graphics` (1)

- `void clearRect(int x, int y, int width, int height)`
- `void copyArea(int x, int y, int width, int height, int dx, int dy)`
- `void draw3DRect(int x, int y, int width, int height, boolean raised)`
- `void fill3DRect(int x, int y, int width, int height, boolean raised)`
- `void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)`
- `void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)`
- `void drawImage(Image img, int x, int y, ImageObserver obs)`
- `void setClip(int x, int y, int width, int height)`
- `Shape getClip()`
- `Rectangle getClipBounds()`
- `void clipRect(int x, int y, int width, int height)`

Einige weitere Methoden der Klasse `java.awt.Graphics` (2)

- `Color` **`getColor`**`()`
- `Font` **`getFont`**`()`

- Weitere **`fill...`** - analog zu **`draw...`** (s.o.)
- Weitere **`set...`** - analog zu **`get...`** (s.o.)

- Verschiedene Varianten von **`drawImage()`**

Weiteres Beispiel: Reagieren auf MouseEvents (1)

- Im folgenden Beispiel wird das Zeichnen über Maus-Ereignisse gesteuert.
  - D.h. das Zeichnen erfolgt hier mithilfe von Methoden, die über das Event-Handling aufgerufen werden, nicht mithilfe der Methode `paint()` einer GUI-Komponente, die über den Repaint-Manager aufgerufen wird.
- Für Maus-Ereignisse wie „Drücken auf eine Maustaste“ und „Bewegen der Maus“ gibt es die Event-Klasse `MouseEvent` und die zugehörigen Listener-Schnittstellen `MouseListener` und `MouseMotionListener`.
- Gezeichnet wird hier direkt in ein `JFrame`-Fenster.
  Der grafische Kontext (Möglichkeit zum Zeichen) wird erzeugt mit der `JFrame`-Methode
  `Graphics` **`getGraphics`**`()`

Weiteres Beispiel: Reagieren auf MouseEvents (2)

```
import java.awt.*;  import java.awt.event.*; import javax.swing.*;

public class Scribble extends JFrame implements MouseListener, MouseMotionListener {
    private int lastX, lastY;
    Graphics g;

    public Scribble() {
        this.setDefaultCloseOperation(
                    JFrame.EXIT_ON_CLOSE);
        this.addMouseListener(this);
        this.addMouseMotionListener(this);
        this.setSize(200, 200);
        this.setVisible(true);
        this.g = this.getGraphics();
    }
    public void mousePressed(MouseEvent e) {
        this.lastX = e.getX();
        this.lastY = e.getY();
    }
    public void mouseDragged(MouseEvent e) {
        int x = e.getX(), y = e.getY();
        this.g.drawLine(this.lastX,
                        this.lastY, x, y);
        this.lastX = x;
        this.lastY = y;
    }

}
```

```
    public void mouseReleased(MouseEvent e) {
    }
    public void mouseClicked(MouseEvent e) {
    }
    public void mouseEntered(MouseEvent e) {
    }
    public void mouseExited(MouseEvent e) {
    }
    public void mouseMoved(MouseEvent e) {
    }
    public static void main(String[] args) {
       new Scribble();
    }
```

Online-Literatur

- „Painting in AWT and Swing“ von Oracle
  http://www.oracle.com/technetwork/java/painting-140037.html