Objektorientierte Programmierung

Studiengang Medieninformatik

Hans-Werner Lang

Hochschule Flensburg

Vorlesung 4
05.04.2017

Was bisher geschah...

Hochschule
Flensburg
University of
Applied Sciences

- Klassen und Objekte
- Attribute und Methoden
- Klassen ableiten mit `extends`
- Vererbung von Attributen und Methoden
- Methoden überschreiben
- Methoden und Konstruktoren überladen
- Polymorphie
- Abstrakte Klasse, abstrakte Methoden
- Interface

Hochschule Flensburg
University of Applied Sciences

GUI (*Graphical User Interface*) Elemente

Java-Klassenbibliothek javax.swing importieren

Elemente, die wir benutzen:

JFrame
JPanel
JLabel
JButton
JTextField

GameFrame Code (1)

Hochschule Flensburg
University of Applied Sciences

```
public class GameFrame extends JFrame
{
    // Attribute
    private JPanel pnl1, pnl2;
    private JLabel lbl1, lbl2;
    private JTextField txt1, txt2;
    private JButton btn1;

    // Konstruktor
    public GameFrame()
    {
        setTitle("Vier gewinnt");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(500, 300);

        pnl1=new PaintPanel();
        pnl1.setBackground(Color.GREEN);
        add(pnl1, BorderLayout.CENTER);

        pnl2=new JPanel();
        pnl2.setBackground(new Color(220, 220, 220));
        add(pnl2, BorderLayout.SOUTH);

        ...
```

GameFrame Code (2)

Hochschule Flensburg
University of Applied Sciences

```
        ...

        lbl1=new JLabel("Spieler 1: ");
        pnl2.add(lbl1);

        txt1=new JTextField(10);
        pnl2.add(txt1);

        lbl2=new JLabel("Spieler 2: ");
        pnl2.add(lbl2);

        txt2=new JTextField(10);
        pnl2.add(txt2);

        btn1=new JButton("Neues Spiel");
        pnl2.add(btn1);

        setVisible(true); // als letzte Anweisung
    }
```

Grafik *Circle* (1)

Hochschule Flensburg
University of Applied Sciences

Wir wollen Kreise, Rechtecke und Linien als Klassen modellieren.

Wir versehen diese Klassen mit einer Methode *draw*, um sie auf dem Bildschirm anzuzeigen.

```
public class Circle
{
    // Attribute
    public int left, top, diameter;
    public Color fill=Color.BLACK;   // Initialisierung des Attributs fill

    // Konstruktor
    public Circle(int l, int t, int d)
    {
        left=l;
        top=t;
        diameter=d;
    }

    ...
```

Grafik *Circle* (2)

Hochschule Flensburg
University of Applied Sciences

```
    ...

    // Farbe festlegen
    public void setFill(Color f)
    {
        fill=f;
    }

    // Objekt zeichnen
    public void draw(Graphics gr)
    {
        gr.setColor(fill);
        gr.fillOval(left, top, diameter, diameter);
    }

}
```

Grafik *Circle* (3) – in *PaintPanel* zeichnen

Hochschule Flensburg
University of Applied Sciences

```
public class PaintPanel extends JPanel
{

    Circle c=new Circle(50, 60, 40);

    @Override
    public void paint(Graphics gr)
    {
        super.paint(gr);
        c.draw(gr);        // Kreis zeichnen
    }

}
```

Ereignisse (*Events*) (1)

Hochschule Flensburg
University of Applied Sciences

Um auf Ereignisse zu reagieren, wie z.B. auf einen Maus-Klick, werden Ereignis-empfänger (*Listener*) benötigt. Dies sind Klassen, die ein entsprechendes Listener-Interface implementieren.

Wir schreiben keine eigene Klasse, sondern implementieren das *MouseListener*-Interface in der Klasse *PaintPanel*.

Dann übergeben wir der Klasse *PaintPanel* im Konstruktor ein *MouseListener*-Objekt – hier das aktuelle Objekt `this` selbst.

```
public class PaintPanel extends JPanel implements MouseListener
{
    ...
    // Konstruktor
    public PaintPanel()
    {
        addMouseListener(this);
    }

    ...
}
```

Ereignisse (*Events*) (2)

Hochschule Flensburg
University of Applied Sciences

Das *MouseListener*-Interface enthält eine ganze Reihe von Methoden, die implementiert werden müssen. Wir lassen die meisten Methoden-Rümpfe leer und implementieren tatsächlich nur die Methode *mousePressed*:

```
    ...
    @Override
    public void mouseEntered(MouseEvent evt)
    {
    }

    @Override
    public void mousePressed(MouseEvent evt)
    {
        c.setFill(Color.BLUE);
        repaint();
    }

    ...

}
```

Ereignisse (*Events*) (3)

Hochschule Flensburg
University of Applied Sciences

Um zu prüfen, ob in den Kreis geklickt worden ist, ermitteln wir die Koordinaten *x* und *y* des Klicks. Dann übergeben wir dem *Circle*-Objekt *c* diese Koordinaten und lassen dort prüfen, ob *x* und *y* innerhalb des Kreises liegen.

```
    ...
    @Override
    public void mousePressed(MouseEvent evt)
    {
        int x=evt.getX();
        int y=evt.getY();
        if (c.clicked(x, y))
            c.setFill(Color.BLUE);
        repaint();
    }
    ...

}
```

Zusammenfassung

Hochschule Flensburg
University of Applied Sciences

- GUI-Steuerelemente *JFrame*, *JPanel*, *JLabel*, *JButton* und *JTextField*
- Grafik-Element *Circle* als Klasse mit Methode *draw*
- Interface *MouseListener*

Beim nächsten Mal:

- Klassenattribute und Klassenmethoden
- Datenstruktur *ArrayList*
- Typ-Parameter