

13.5 Objekte mit internem Zustand

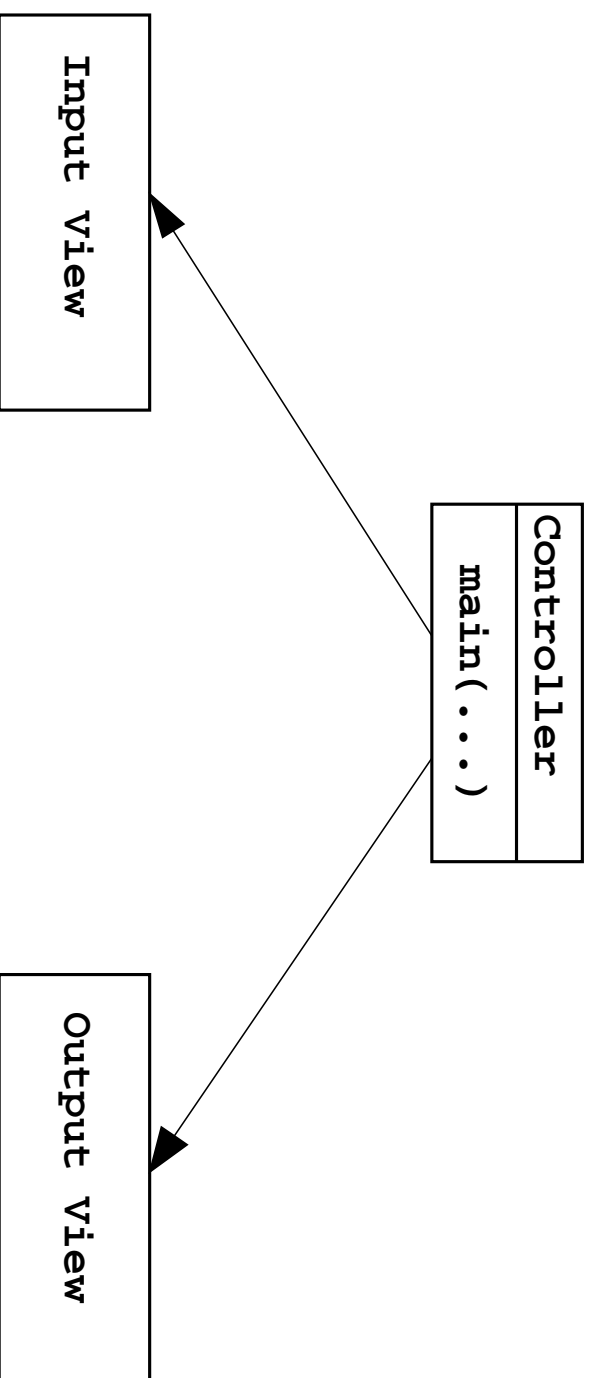
Inhalt

- Interaktive Eingabe
- Ausgabe per GUI
- Vererbung
- Feldvariable (Attribute, Instanzvariable)

13.5.1 Eingabe

Bisher: über Programmparameter

Jetzt: interaktiv im Dialog



```

import java.io.*; // Package mit Ein-/Ausgabeoperationen
public class CelsiusToFahrenheit2
{ public static void main (String[] args)
    throws IOException
    { BufferedReader keyboard =
        new BufferedReader (new InputStreamReader (System.in));
      System.out.print ("Degrees Celsius: ");
      int c = new Integer (keyboard.readLine ()).intValue ();
      double f = ((9.0/5.0) * c) + 32;
      System.out.println("For Celsius degrees " + c + ",");
      System.out.println("Degrees Fahrenheit = " + f);
    }
  }
}

```

- **throws IOException** deklariert, dass Fehler vom Typ `IOException` auftreten können.

Ein-/Ausgabeobjekte

- `System.out` Ausgabeobjekt für Konsole
 - `System.in` Eingabeobjekt für Konsole
- Methode `read ()` liest ein Zeichen
- bessere Funktionalität:

```
BufferedReader keyboard =  
    new BufferedReader (  
        new InputStreamReader (System.in));
```

```
String text = keyboard.readLine ();  
liest eine Zeile von der Konsole
```

13.5.2 Ausgabe

Bisher: Textausgabe auf Konsole

Jetzt: Fenster mit Grafik und Text

Verwendete Packages:

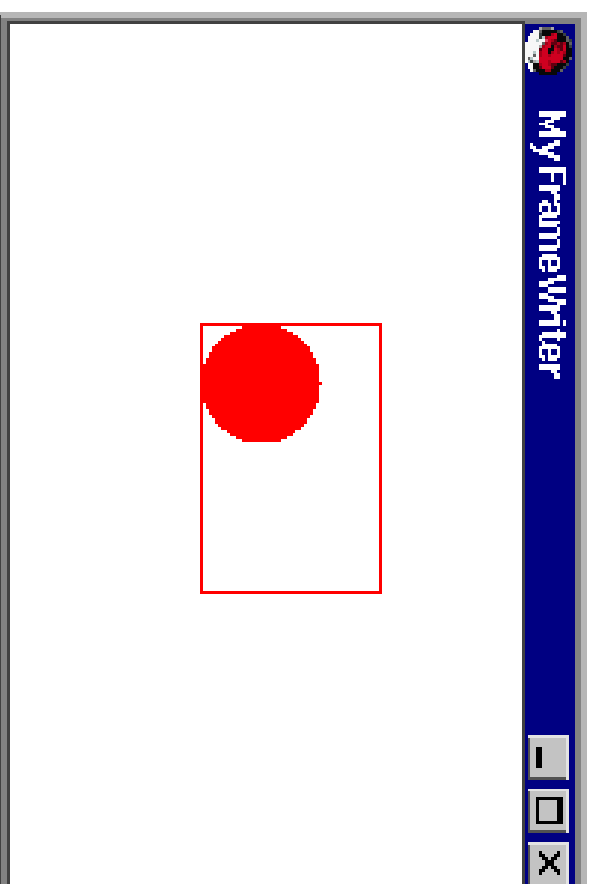
- `java.awt` (abstract windowing toolkit)
- `javax.swing`

Fenster \rightsquigarrow **Objekt der Klasse** JFrame

Frame: Repräsentation eines rechteckigen Bereichs auf dem Bildschirm

\rightsquigarrow Fenster

\rightsquigarrow implementiert durch (Unterklassen von) JFrame



Erzeugen eines Frames

```
import java.awt.*;
import javax.swing.*;
/** FrameTest creates a frame */
public class FrameTest
{ public static void main(String[] args)
  { JFrame f = new JFrame();
    f.setSize (300,200);
    f.setVisible (true);
    System.out.println("There is the frame!");
  }
}
```

... mit Funktionalität durch Vererbung

```
import java.awt.*;
import javax.swing.*;
/** MyFrameWriter creates a white frame and displays it */
public class MyFrameWriter extends JFrame
{
    public MyFrameWriter ()
    {
        setTitle ("MyFrameWriter");
        setBackground (Color.white);
        setSize (300,200);
        setVisible (true);
    }
}
```

- MyFrameWriter ist Unterklasse von **JFrame**
- **Methodenaufrufe ohne Objekt** beziehen sich auf das aktuelle Objekt **this**

Konstruktormethoden

- tragen den Namen der Klasse
`public MyFrameWriter () {statements}`
- `{statements}` werden ausgeführt, **nachdem** das neue Objekt erzeugt worden ist
- dienen der Initialisierung von Attributen
- können Parameter annehmen

Funktionalität durch Überschreiben von paint

```
// in MyFrameWriter:
/** paint fills the window with the items that must be displayed
 * @param g - the graphics pen that draws onto the window */
public void paint (Graphics g)
{
    g.setColor (Color.red);
    int left_edge = 105;
    int bottom = 130;

    int width = 90;
    int depth = 60;
    g.drawRect (left_edge, bottom - depth, width, depth);

    int diameter = 40;
    g.fillOval (left_edge, bottom - diameter, diameter, diameter);
}
```

13.5.3 Feldvariable

„Gedächtnis“ eines Objekts

Bisher: Lebensdauer einer Variable \Leftrightarrow Methodenaufruf

Jetzt:

- Lebensdauer einer Feldvariable \Leftrightarrow Objekt
- Initialisierung vor Start der Konstruktormethode
- `private <type> <identifizier>` in der Klassendefinition
- nur von den Methoden des Objekts zugreifbar (`private`)

Beispiel

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

/** EggWriter creates a graphical window that display an egg. Each time
 * the window is repainted, the egg shrinks by half. */
public class EggWriter extends JFrame
{
    private int egg_width = 600;
    private int egg_height = 400;

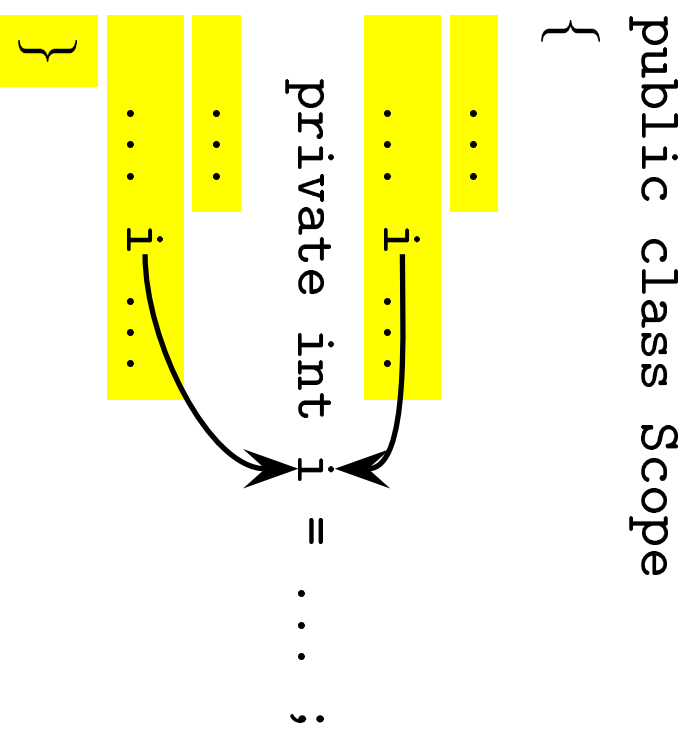
    /** Constructor EggWriter creates the window and makes it visible */
    public EggWriter (int width, int height)
    {
        setTitle ("EggWriter");
        setSize (width, height);
        setBackground (Color.YELLOW);
        setVisible (true);
    }
}
```

```

/** paint draws the egg.
 * @param g - the graphical pen the draws the egg */
public void paint (Graphics g)
{
    int left_border = 10; // horizontal position of the egg
    int baseline = 190; // where to lay the egg
    g.setColor (Color.pink);
    g.fillOval (left_border, baseline - egg_height,
               egg_width, egg_height);
    g.setColor (Color.black);
    g.drawOval (left_border, baseline - egg_height,
               egg_width, egg_height);
    // reset the fields so that the egg shrinks
    // the next time it is painted
    egg_width = egg_width / 2;
    egg_height = egg_height / 2;
}
}

```

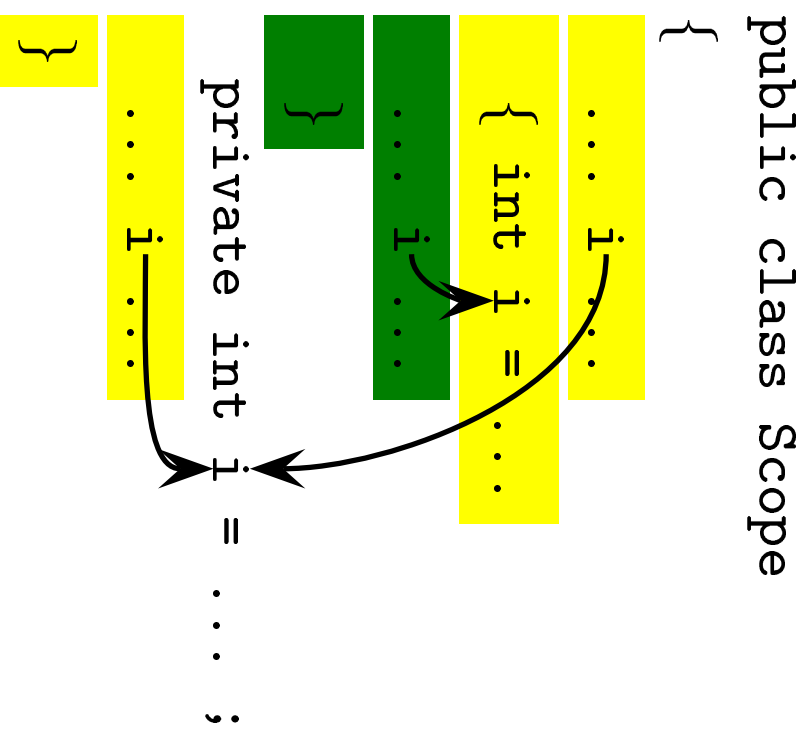
Gültigkeitsbereich einer Feldvariable



Alle Komponentendeklarationen der Klasse (ausser der rechten

Seite der Initialisierung von i)

Löcher im Gültigkeitsbereich



13.5.4 Subtyping und Zuweisung

Falls class A extends B , so verstehen Objekte der Klasse A sämtliche Methoden der Klasse B .

Also: A ist Subtyp von B , d.h., $A < B$.

Setze $C = \text{typeof}(x)$, falls x ist als Variable deklariert durch Cx ;

Zuweisung $x = y$ ist nur erlaubt, falls $\text{typeof}(y) < \text{typeof}(x)$. Andernfalls: Fehlermeldung vom Compiler.

Beispiel: MyFrameWriter < JFrame und EggWriter < JFrame

```
import java.awt.*;
import javax.swing.*;
public class ShowEgg
{ public static void main (String[] args)
  { JFrame f = new EggWriter (300, 200);
  }
}
```

13.5.5 Ergänzungen zur Syntax

```
<class> ::= public class <class name>
                extends <class name> { <components> }
<components> ::= <component> <components> | ε
<component> ::= <method> | <constructor> | <field>
<field> ::= <type> <identifier> ;
                | <type> <identifier> = <expr> ;
<constructor> ::= public <class name> ( <formals> )
                { <statements> }
<formals> ::= <formals-1> | ε
<formals-1> ::= <formal> | <formal> , <formals-1>
<formal> ::= <type> <identifier>
```