

Folge 1.3

Wir programmieren ein kleines Bild

Folge 1.3

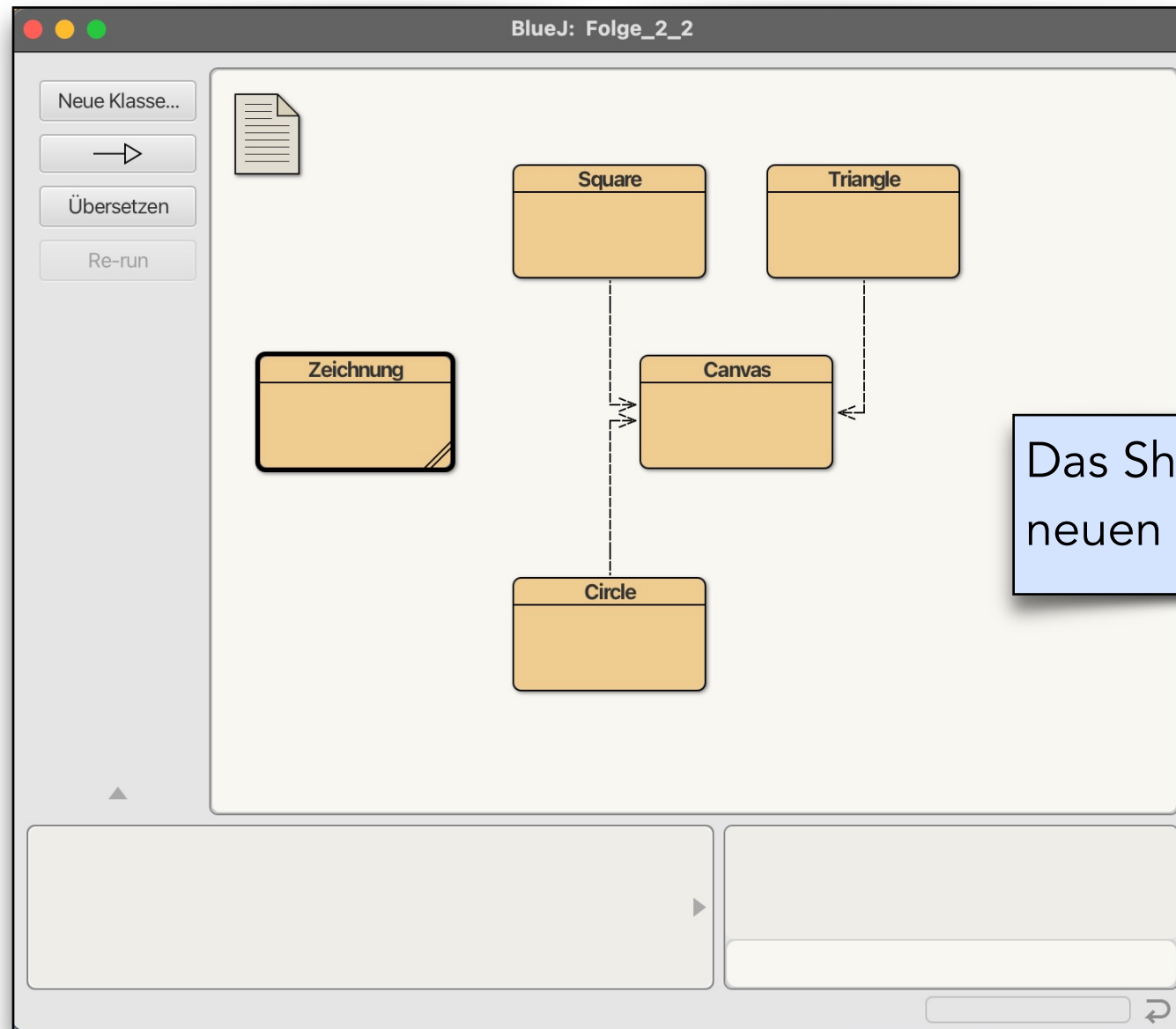
Wir programmieren ein kleines Bild

1.3.1 Das Shapes-Projekt, Fortsetzung

1.3.2 Eigene Methoden erstellen

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

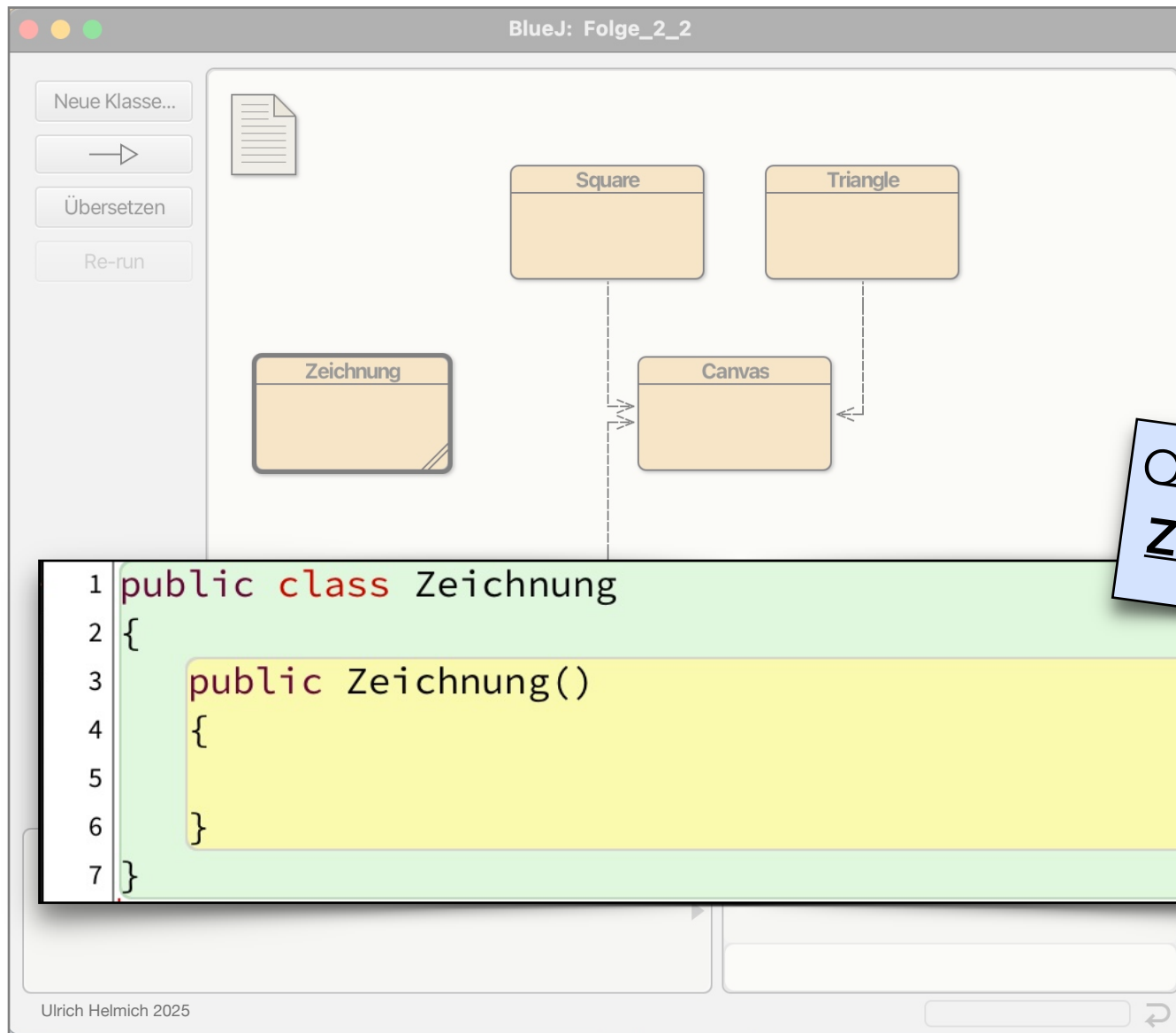


Das Shapes-Projekt mit der neuen Klasse **Zeichnung**.



Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



Quelltext der Klasse
Zeichnung.

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

```
1 public class Zeichnung
2 {
3     public Zeichnung()
4     {
5
6     }
7 }
```

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

Sichtbarkeit

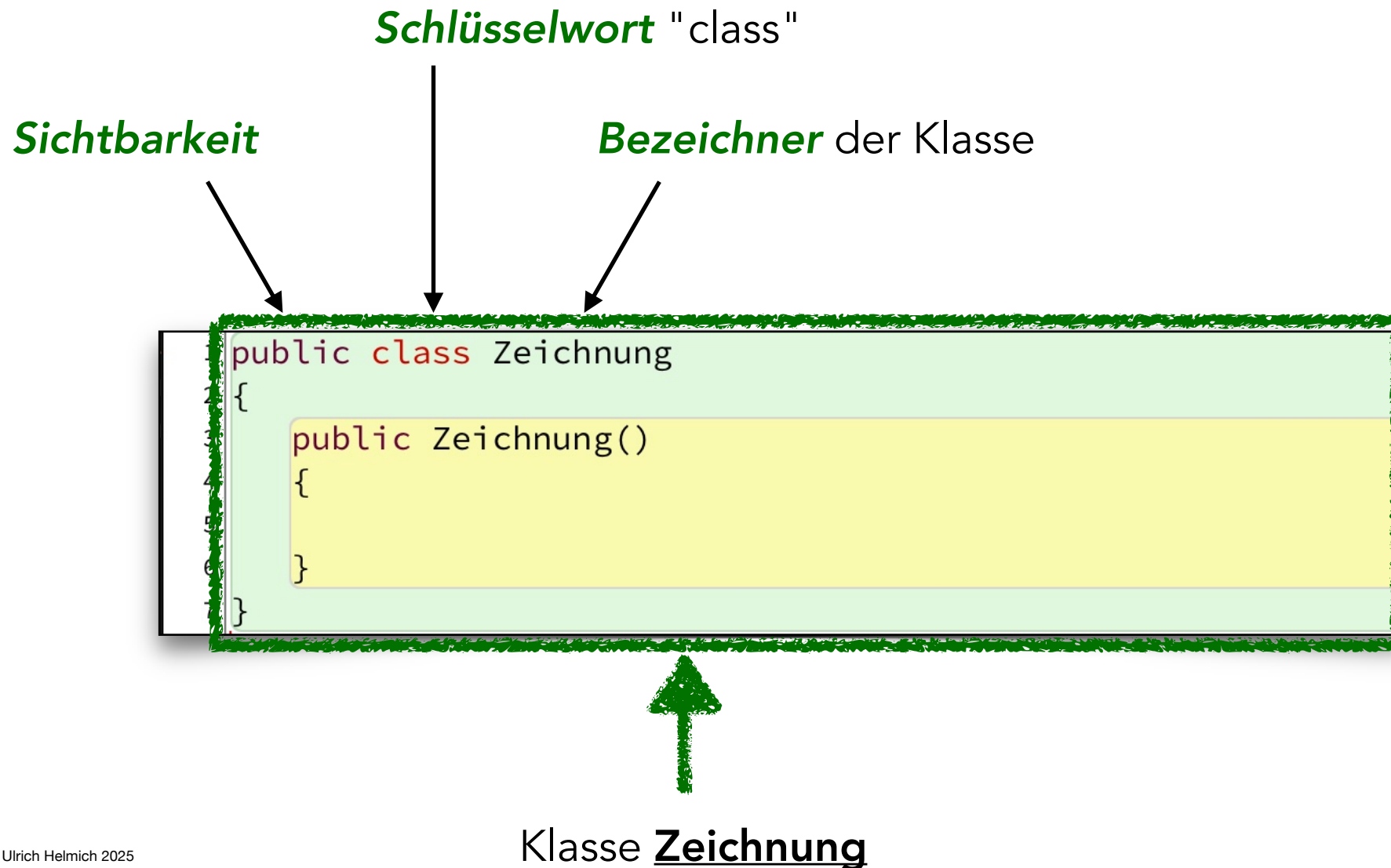
Schlüsselwort "class"

Bezeichner der Klasse

```
1 public class Zeichnung
2 {
3     public Zeichnung()
4     {
5
6     }
7 }
```

Einführung in die Objektorientierte Programmierung (OOP)

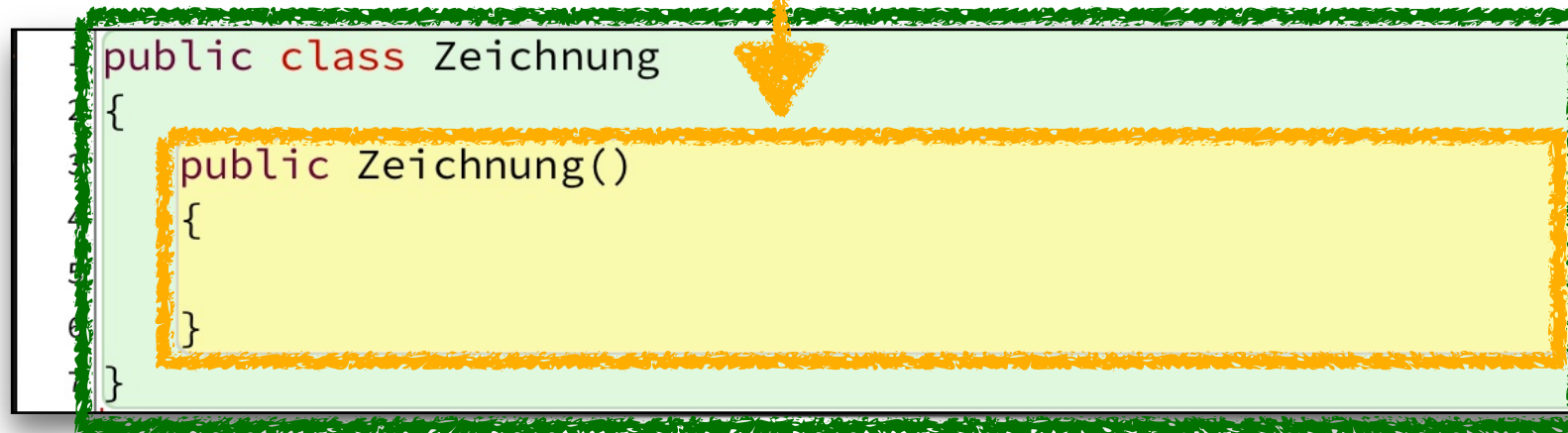
1.3.1 Das Shapes-Projekt, Fortsetzung



Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

Konstruktor der Klasse

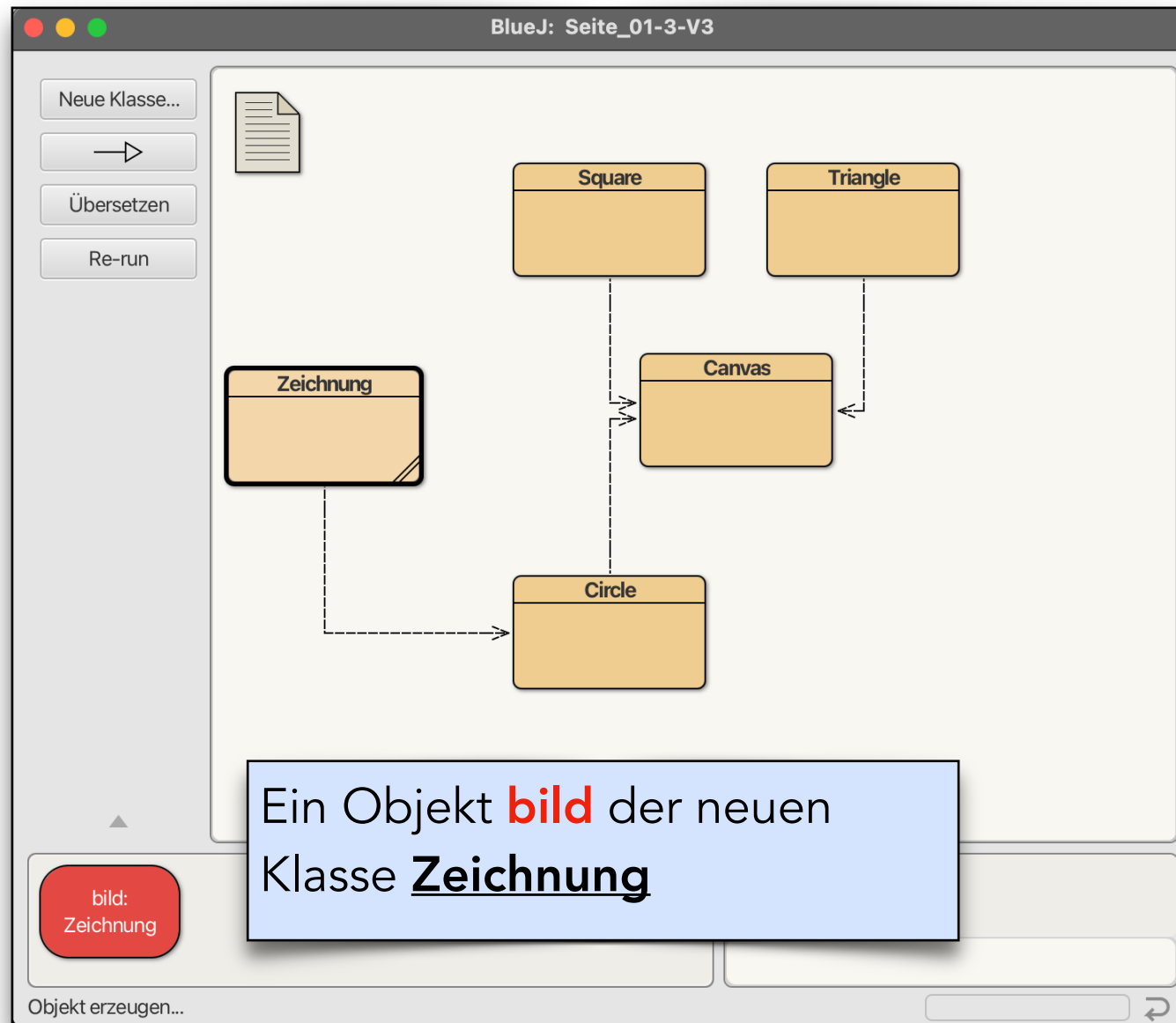


```
1 public class Zeichnung
2 {
3     public Zeichnung()
4     {
5     }
6 }
7 }
```

Klasse Zeichnung

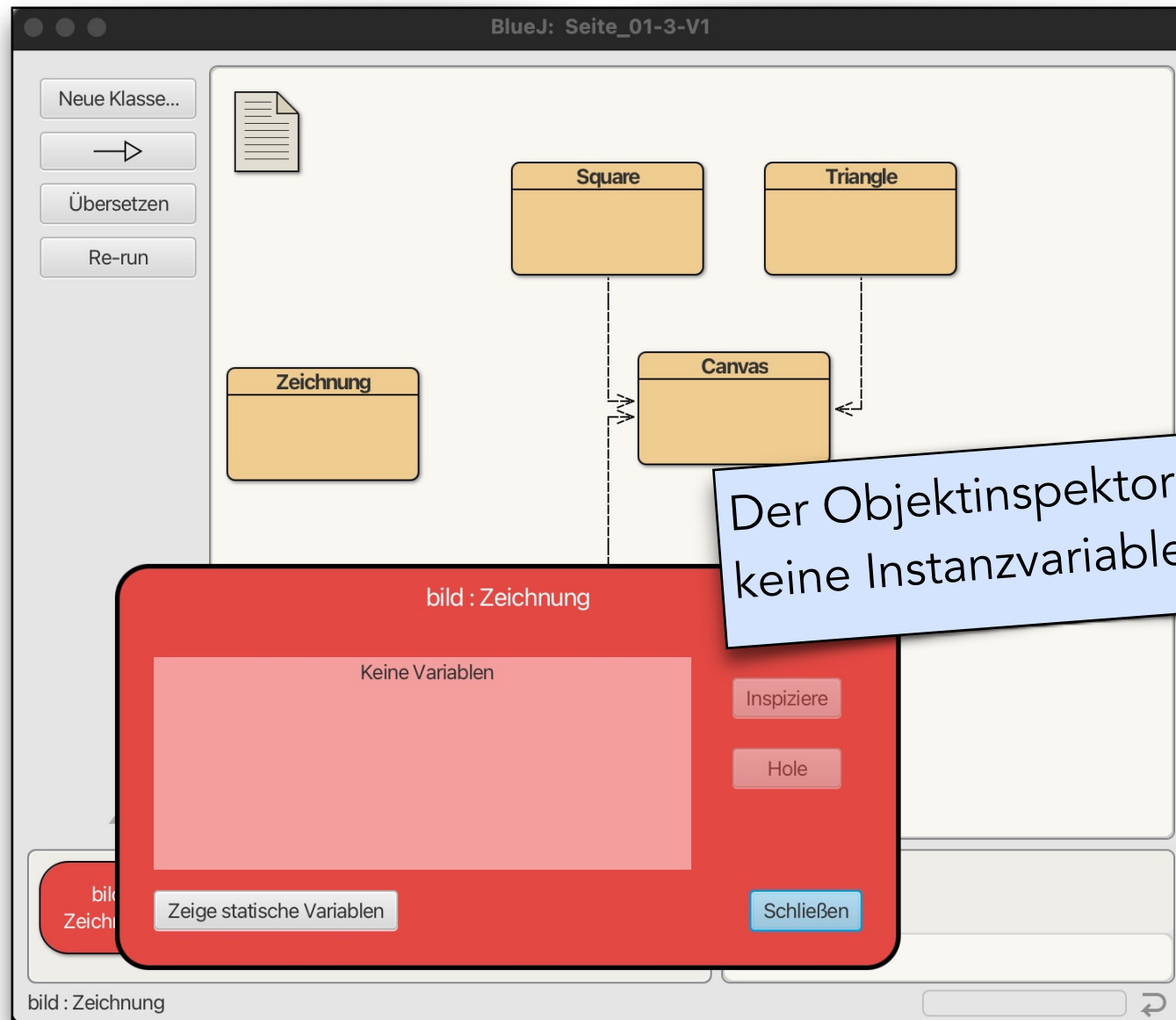
Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

The screenshot shows the BlueJ IDE interface. At the top, the title bar reads 'BlueJ: Seite_01-3-V1'. On the left, there is a sidebar with buttons: 'Neue Klasse...', a right-pointing arrow, 'Übersetzen', and 'Re-run'. The main workspace displays a class hierarchy with 'Square' and 'Triangle' classes, both of which inherit from a base class 'Zeichnung'. Below the workspace, a code editor shows the following Java code for the 'Zeichnung' class:

```
1 public class Zeichnung
2 {
3     public Zeichnung()
4     {
5
6     }
7 }
```

At the bottom of the IDE, there is a red panel with a 'Zeichnung' object. The panel contains buttons for 'Inspiziere', 'Hole', 'Zeige statische Variablen', and 'Schließen'. The status bar at the bottom left indicates 'bild : Zeichnung'.

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

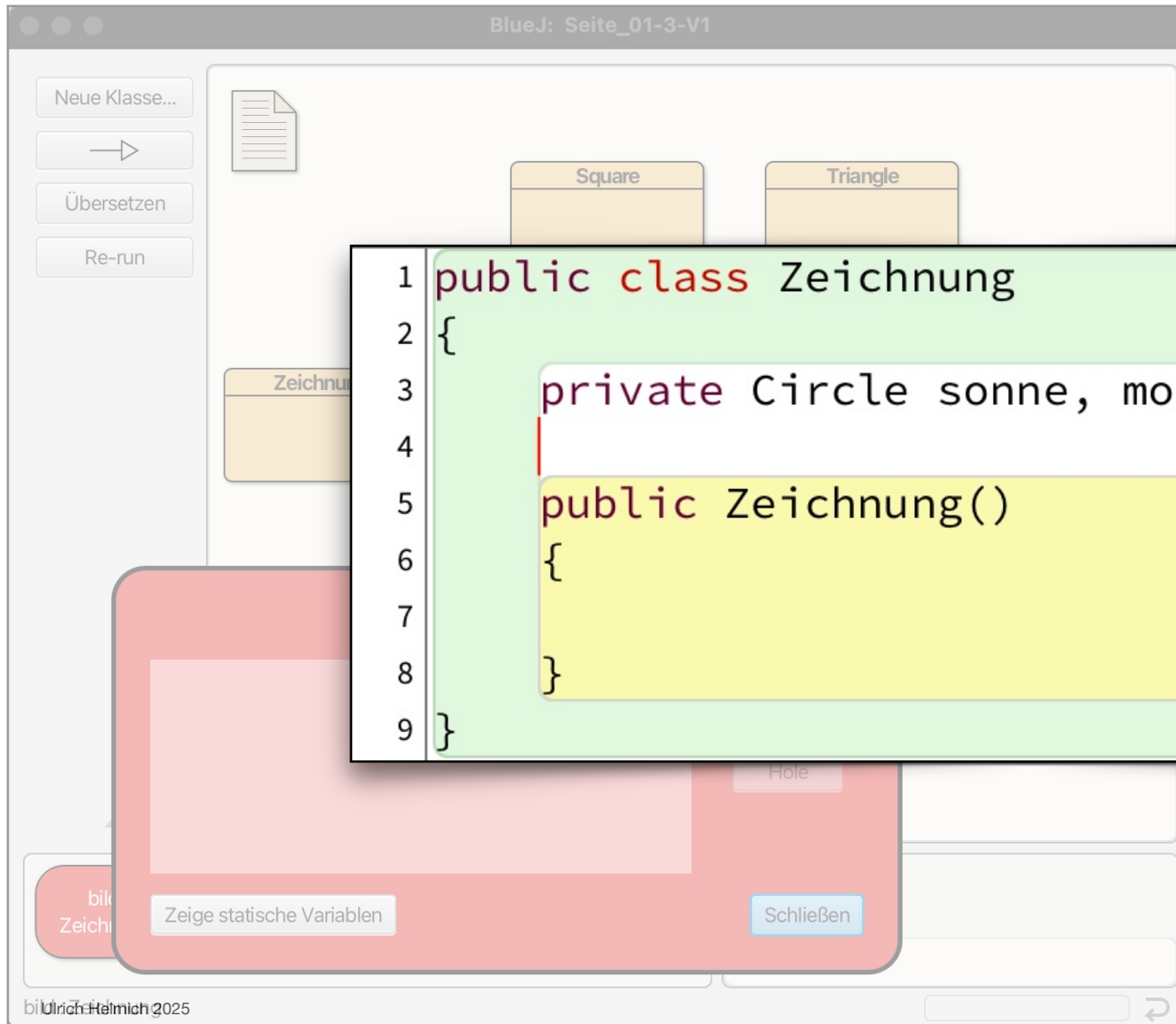
The screenshot shows the BlueJ IDE interface. At the top, the title bar reads 'BlueJ: Seite_01-3-V1'. On the left, there are buttons for 'Neue Klasse...', a right-pointing arrow, 'Übersetzen', and 'Re-run'. The main area displays a class hierarchy with 'Square' and 'Triangle' boxes connected by dashed lines to a 'Zeichnung' box. Below this, a code editor shows the following Java code:

```
1 public class Zeichnung
2 {
3     public Zeichnung()
4     {
5
6     }
7 }
```

A red box highlights the bottom part of the interface, containing a 'Zeichnung' object icon, a 'Zeige statische Variablen' button, and a 'Schließen' button. To the right of the code editor, a blue callout box contains the text: 'Es sind ja auch noch keine Instanzvariablen deklariert!'.

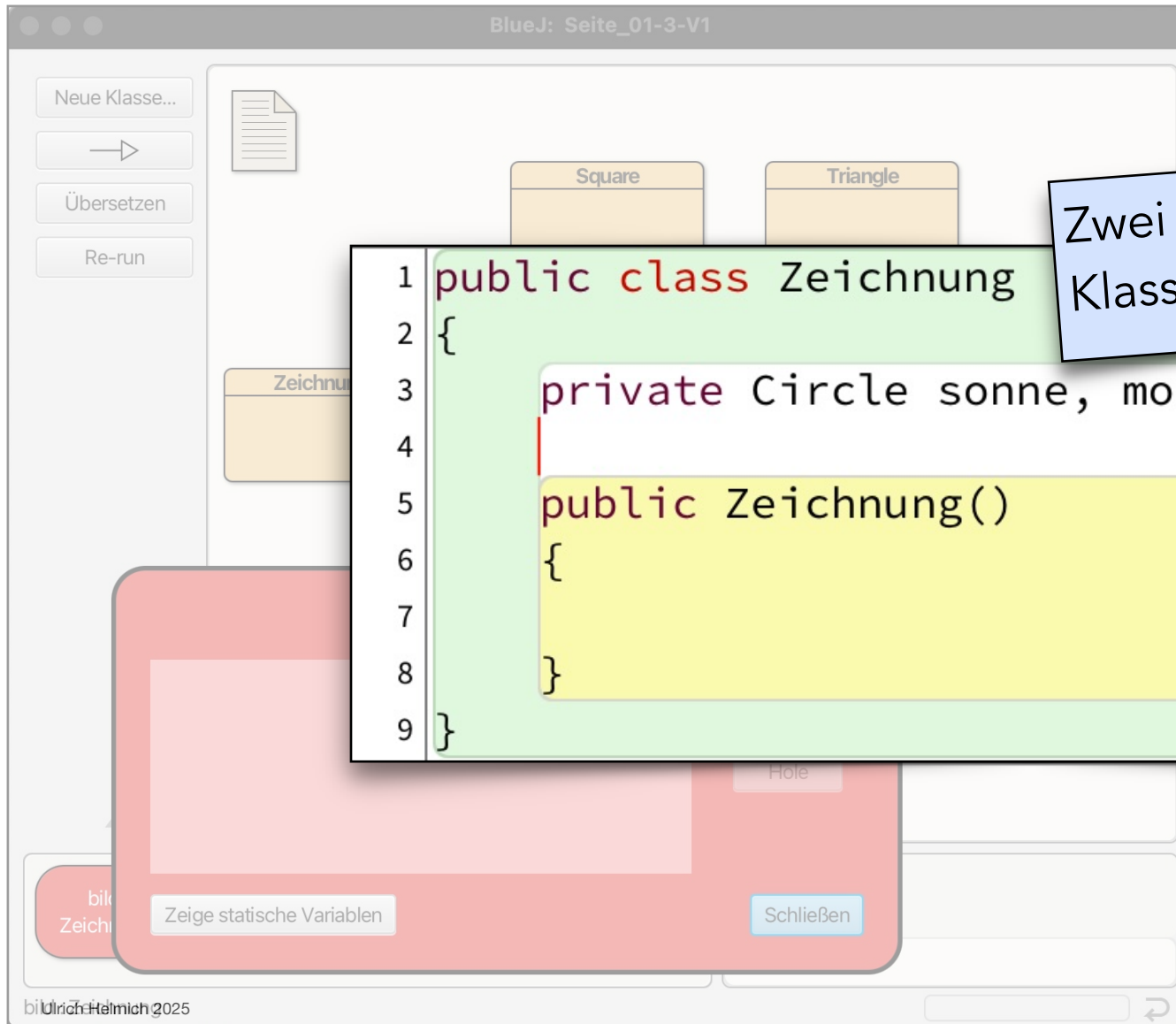
Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



Einführung in die Objektorientierte Programmierung (OOP)

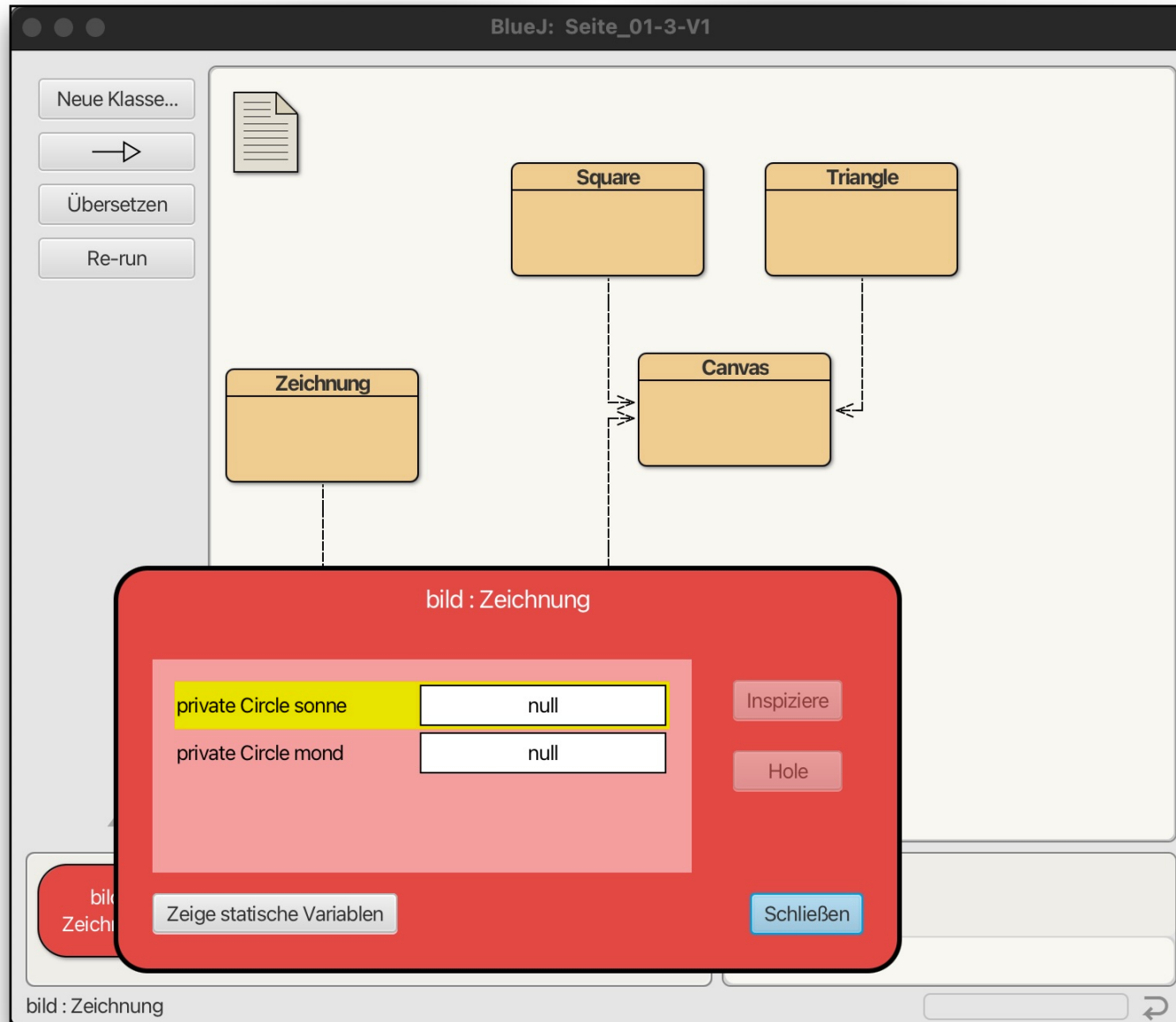
1.3.1 Das Shapes-Projekt, Fortsetzung



Zwei Instanzvariablen der Klasse **Circle**: **sonne** und **mond**

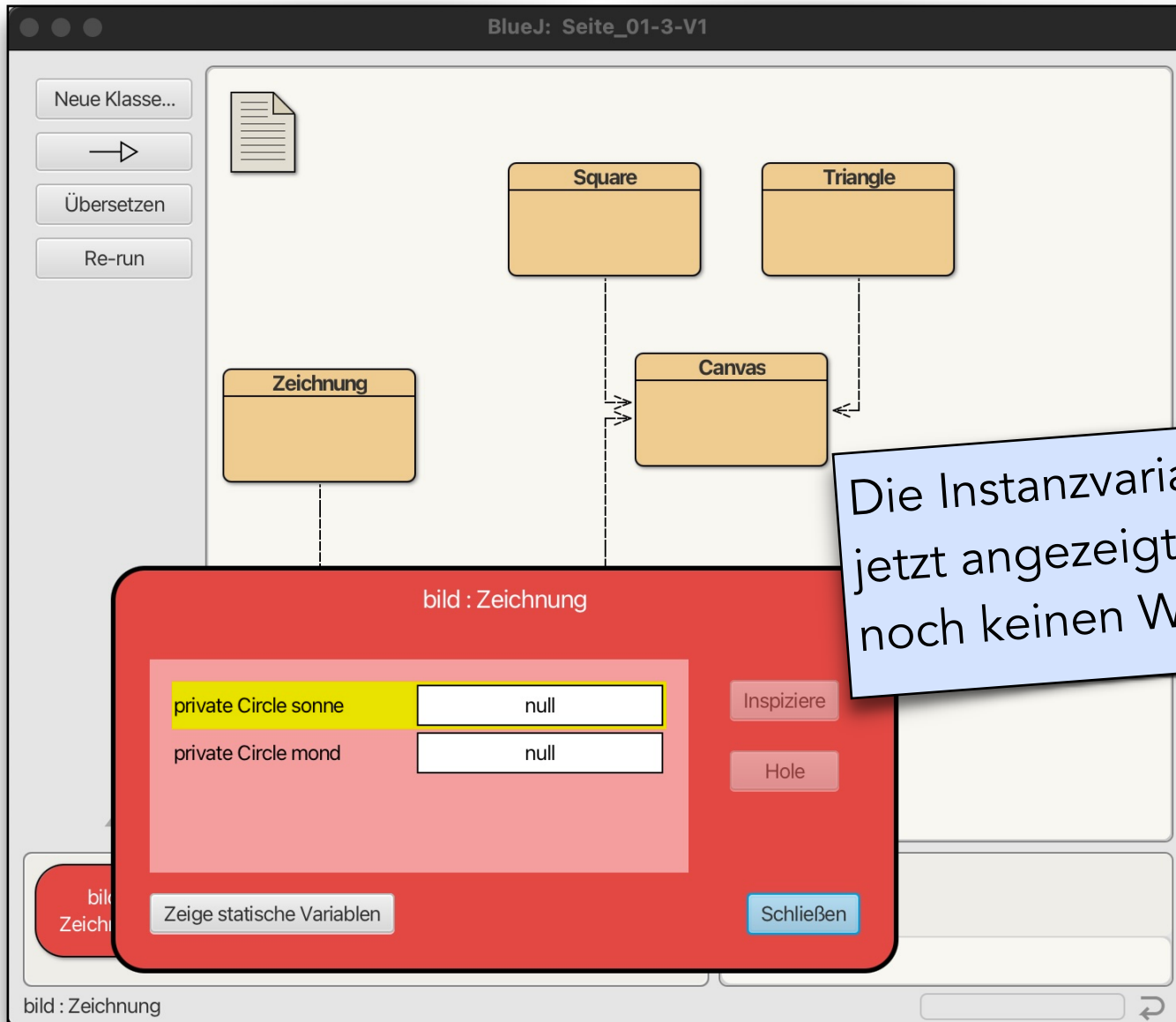
Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



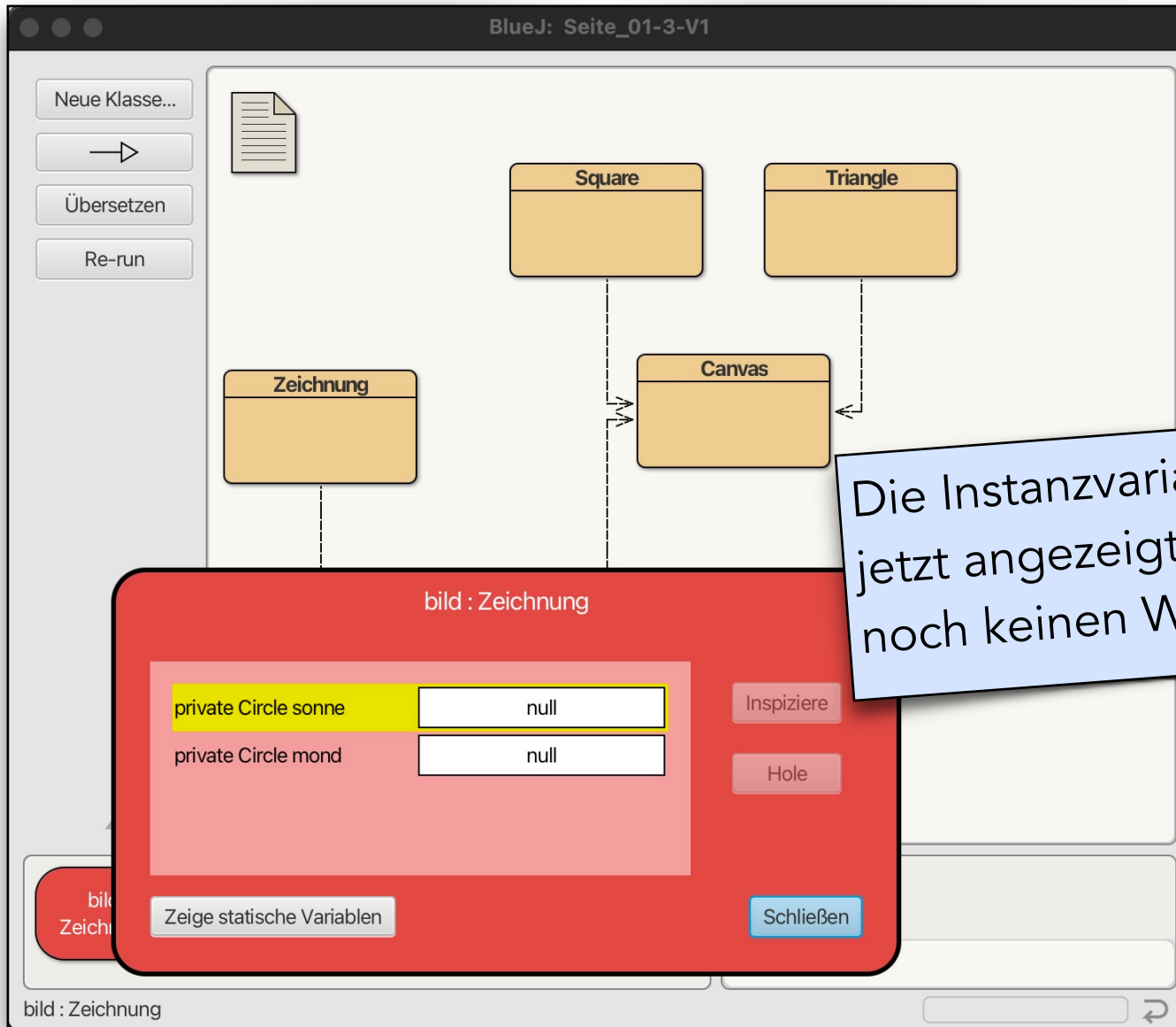
Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

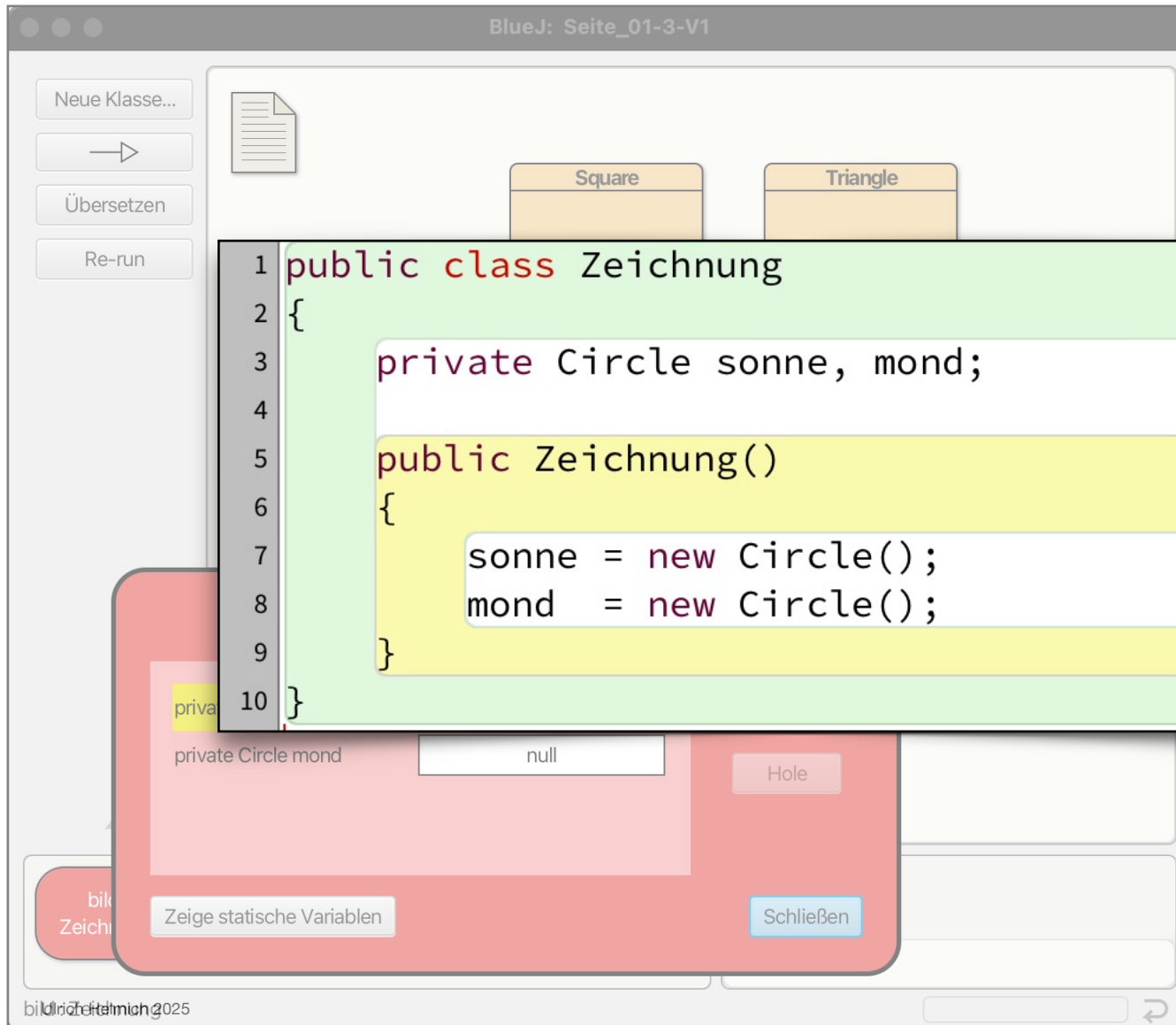


Die Instanzvariablen werden jetzt angezeigt, haben aber noch keinen Wert!

Sie müssen noch initialisiert werden!

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

The screenshot shows the BlueJ IDE interface. At the top, the title bar reads 'BlueJ: Seite_01-3-V1'. On the left, there are buttons for 'Neue Klasse...', a right-pointing arrow, 'Übersetzen', and 'Re-run'. The main area displays the Java code for the 'Zeichnung' class. The code is as follows:

```
1 public class Zeichnung
2 {
3     private Circle sonne, mond;
4
5     public Zeichnung()
6     {
7         sonne = new Circle();
8         mond = new Circle();
9     }
10 }
```

Below the code, a red box highlights the instance variables section, which shows 'private Circle mond' with a value of 'null'. There are buttons for 'Hole' and 'Schließen'. At the bottom left, there is a button for 'Zeige statische Variablen'.

Die Instanzvariablen werden im Konstruktor **initialisiert**.
Erst jetzt existieren sie!

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

BlueJ: Seite_01-3-V1

```
1 public class Zeichnung
2 {
3     private Circle sonne, mond;
4
5     public Zeichnung()
6     {
7         sonne = new Circle();
8         mond  = new Circle();
9     }
10 }
```

Angabe der Klasse, von der ein neues Objekt erzeugt werden soll

new-Befehl zum Erzeugen von neuen Objekten

Bezeichner der Instanzvariablen

Neue Klasse...
→
Übersetzen
Re-run

private Circle sonne null
private Circle mond null

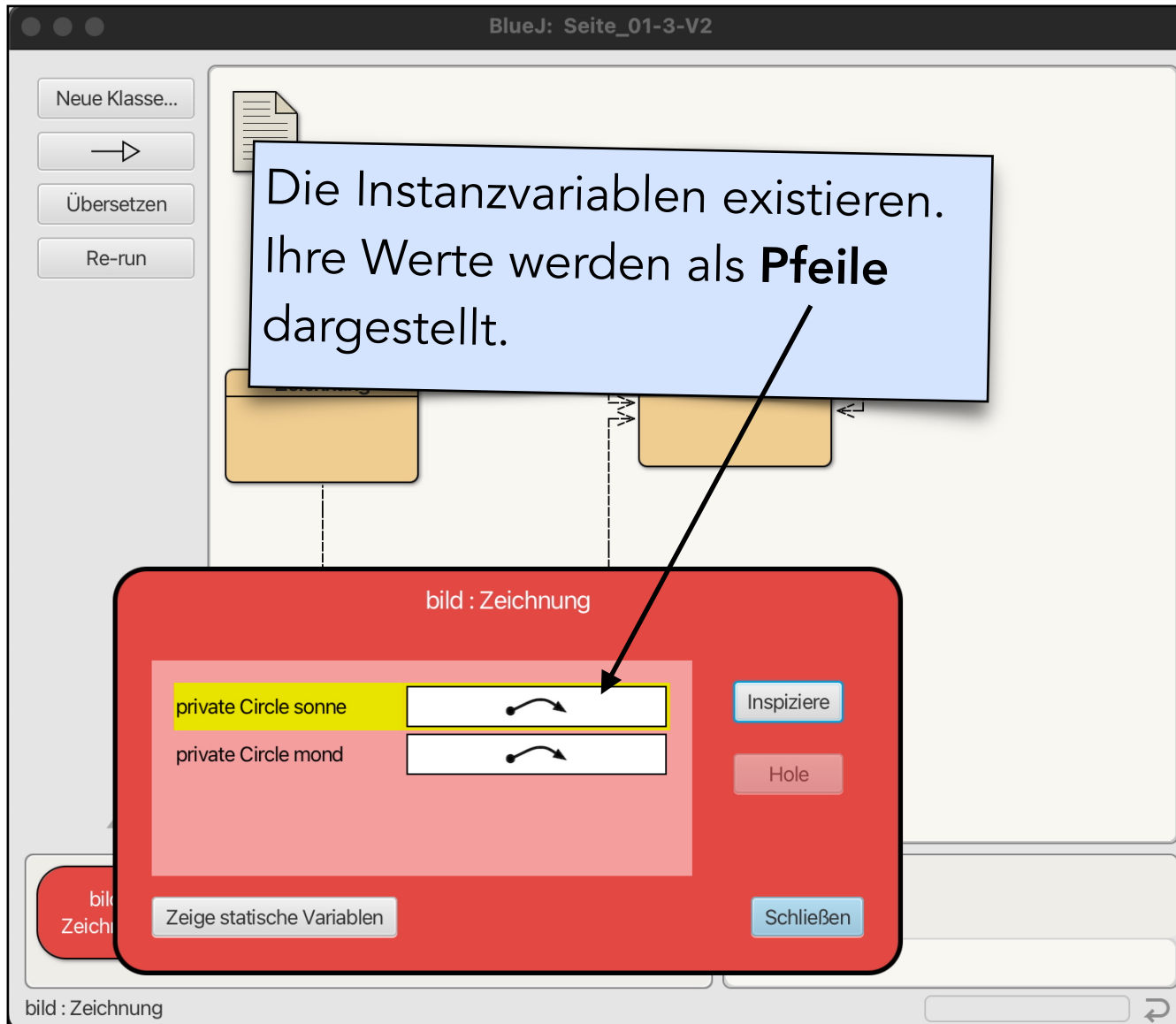
Zeige statische Variablen

Ulrich Helmich 2025

Seite 20 von 31

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung

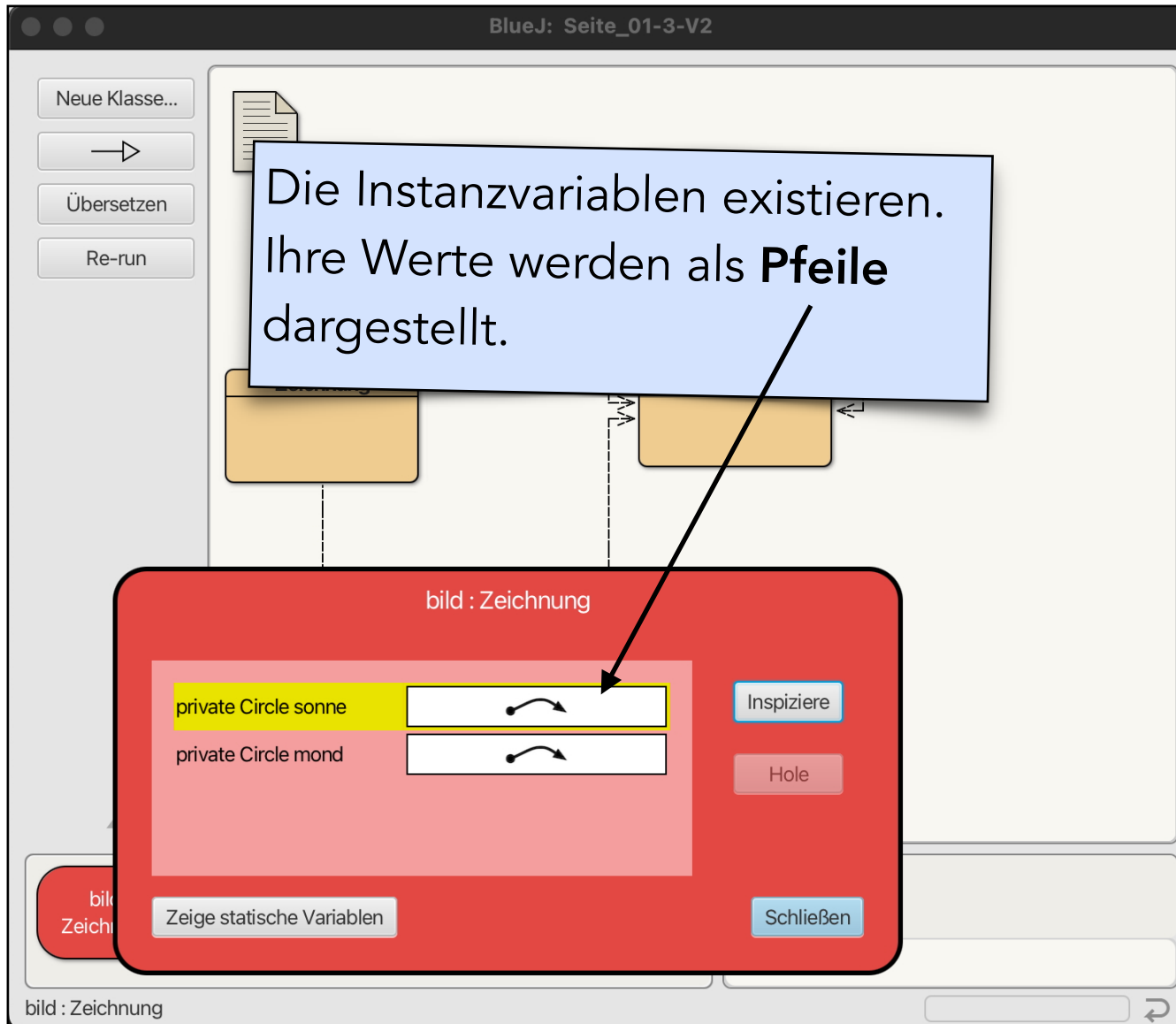


Instanzvariablen speichern keine Daten, sondern **Adressen!**

Sie **verweisen** auf einen Bereich im Speicher, an dem sich die *eigentlichen Daten* der Objekte befinden.

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



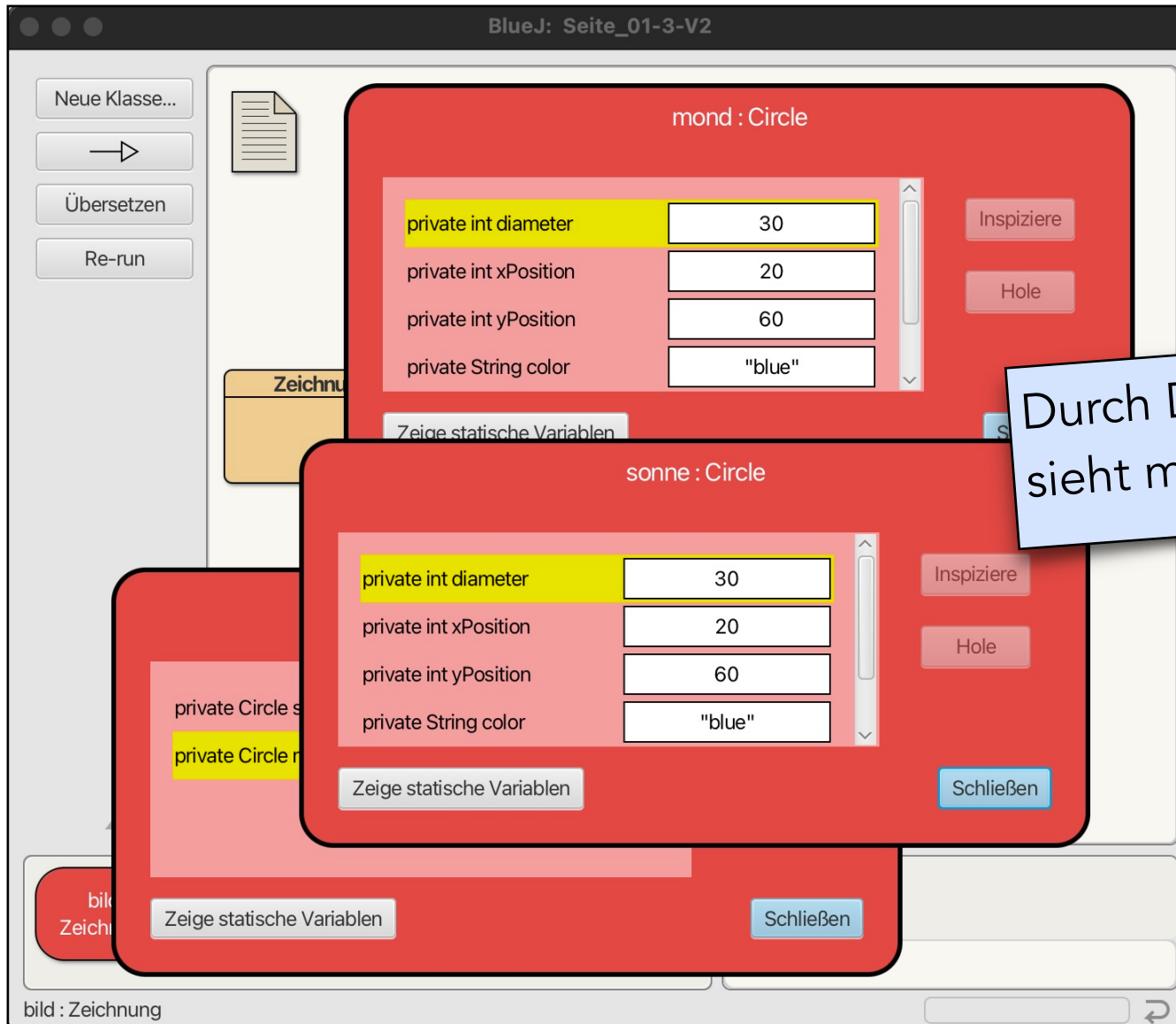
Instanzvariablen speichern keine Daten, sondern **Adressen!**

Sie **verweisen** auf einen Bereich im Speicher, an dem sich die *eigentlichen Daten* der Objekte befinden.

Solche Variablen werden als **Referenzen** bezeichnet.

Einführung in die Objektorientierte Programmierung (OOP)

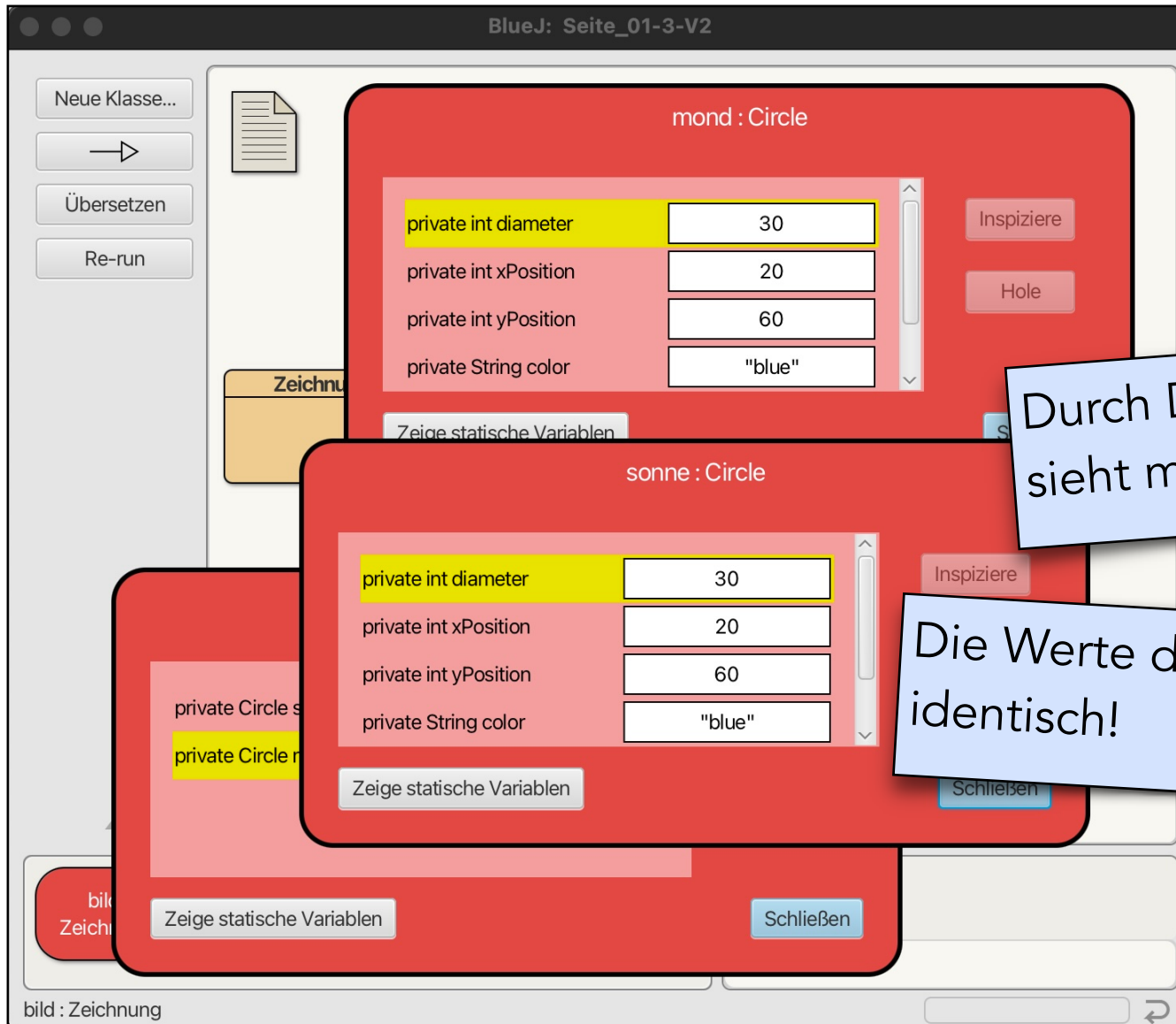
1.3.1 Das Shapes-Projekt, Fortsetzung



Durch Doppelklick auf die Pfeile sieht man die eigentlichen Objekte.

Einführung in die Objektorientierte Programmierung (OOP)

1.3.1 Das Shapes-Projekt, Fortsetzung



Durch Doppelklick auf die Pfeile sieht man die eigentlichen Objekte.

Die Werte der Instanzvariablen sind identisch!

Einführung in die Objektorientierte Programmierung (OOP)

1.3.2 Eigene Methoden erstellen



```
1 public class Zeichnung
2 {
3     private Circle sonne, mond;
4
5     public Zeichnung()
6     {
7         sonne = new Circle();
8         mond = new Circle();
9     }
10
11     public void makeVisible()
12     {
13         sonne.makeVisible();
14         mond.makeVisible();
15     }
16 }
```

Eine eigene Methode
makeVisible()

Einführung in die Objektorientierte Programmierung (OOP)

1.3.2 Eigene Methoden erstellen

Sichtbarkeit der Methode

Rückgabotyp der Methode

Bezeichner der Methode

Parameter der Methode (hier: keine)

```
11 public void makeVisible()  
12 {  
13     sonne.makeVisible();  
14     mond.makeVisible();  
15 }  
16 }
```

Einführung in die Objektorientierte Programmierung (OOP)

1.3.2 Eigene Methoden erstellen

Sichtbarkeit der Methode

Rückgabotyp der Methode

Bezeichner der Methode

Parameter der Methode (hier: keine)

```
11 public void makeVisible()  
12 {  
13     sonne.makeVisible();  
14     mond.makeVisible();  
15 }  
16 }
```

Aufruf der Methode **makeVisible()** des Objekts **mond**

Bezeichner des Objekts

Einführung in die Objektorientierte Programmierung (OOP)

1.3.2 Eigene Methoden erstellen

Sichtbarkeit der Methode

Rückgabetype der Methode

Bezeichner der Methode

Parameter der Methode (hier: keine)

```
11 public void makeVisible()  
12 {  
13     sonne.makeVisible();  
14     mond.makeVisible();  
15 }  
16 }
```

Die Methoden einer Klasse können öffentliche (public) Methoden von Objekten anderer Klassen aufrufen.

Aufruf der Methode **makeVisible()** des Objekts **mond**

Bezeichner des Objekts

Einführung in die Objektorientierte Programmierung (OOP)

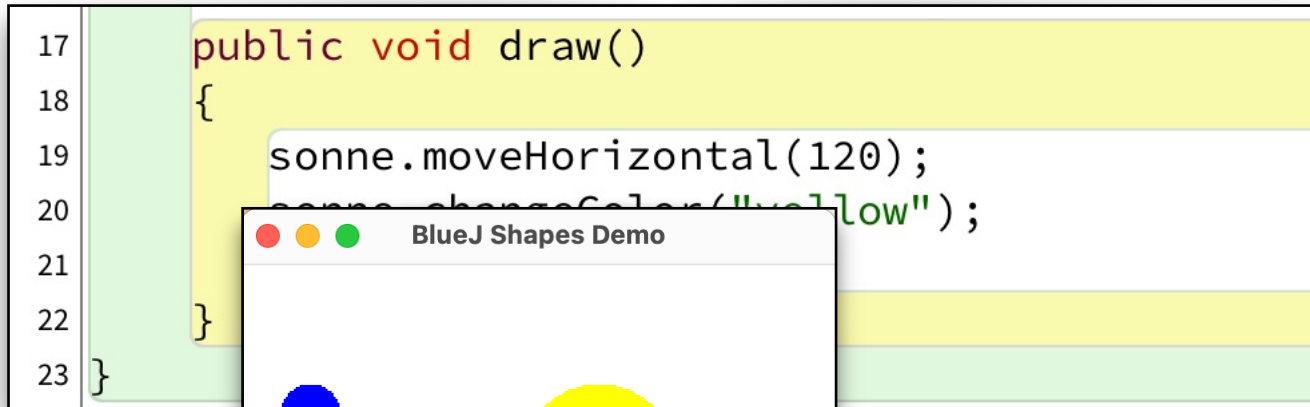
1.3.2 Eigene Methoden erstellen

```
17 public void draw()  
18 {  
19     sonne.moveHorizontal(120);  
20     sonne.changeColor("yellow");  
21     sonne.changeSize(80);  
22 }  
23 }
```

Die neue Methode draw()

Einführung in die Objektorientierte Programmierung (OOP)

1.3.2 Eigene Methoden erstellen



Nach Aufrufen der Methoden

- `bild.makeVisible()`
- `bild.draw()`

mit der Maus in BlueJ

Einführung in die Objektorientierte Programmierung (OOP)

1.3.2 Eigene Methoden erstellen

```
5 public Zeichnung()  
6 {  
7     sonne = new Circle();  
8     mond  = new Circle();  
9     makeVisible();  
10    draw();  
11 }
```

Erweiterung des Konstruktors der Klasse Zeichnung.

makeVisible() und **draw()** werden jetzt bereits beim Erzeugen eines neuen Objekts der Klasse **Zeichnung** automatisch ausgeführt.

Keine weiteren Maus-Aktionen mehr notwendig!