

**IBBB 2010**



# Workshop 6

## Einführung in die objektorientierte Programmierung

Dozenten: J. Penon, J. Frank, A. Schindler



### Teil: Java mit BlueJ

Dozent: A. Schindler



# Gliederung

1. Java
2. BlueJ?
3. Grundlegende Techniken der Objektorientierung
4. Objects First! ?
5. Unterrichtsbeispiele
6. Literatur - Links - Quellen

# Java



*»Java soll eine einfache, objektorientierte, verteilte, interpretierte, robuste, sichere, architekturneutrale, portable, performante, nebenläufige, dynamische Programmiersprache sein.«*

# Java Vorteile

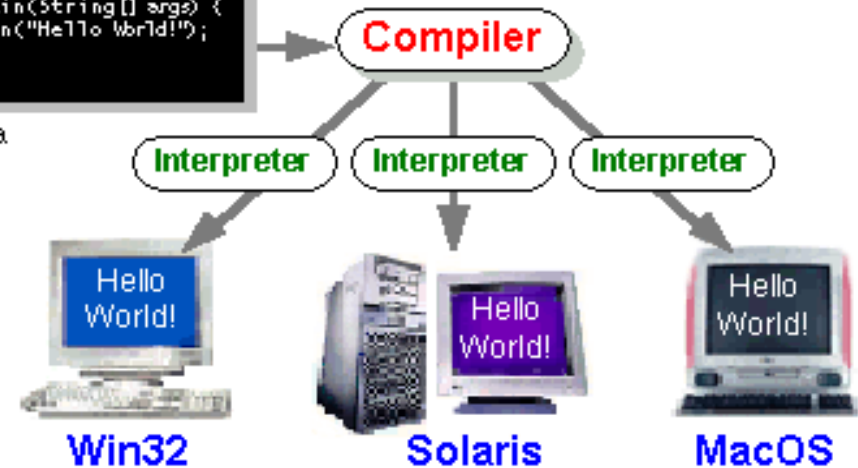
- breite Anwendungsbasis (Industrie, Wissenschaft, Bildung)
- nicht so fehleranfällig wie C / C++
- viele unterstützende Angebote
- kostenlos
- volle Kontrolle über den Code
- plattformunabhängig



## Java Program

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

HelloWorldApp.java



# Java Nachteile

- Dokumentation auf Englisch
- schwieriges Erstellen einer GUI
- nicht für die Schule konzipiert
- Zukunft von SUN ungewiss
- wird ständig weiterentwickelt



# Java Unterstützung



- Reference Card (im Share)
- Handbuch der Java-Programmierung (im Share)
- Bildungsserver:

[http://bildungsserver.berlin-brandenburg.de/fortbildungen\\_informatik.html](http://bildungsserver.berlin-brandenburg.de/fortbildungen_informatik.html)

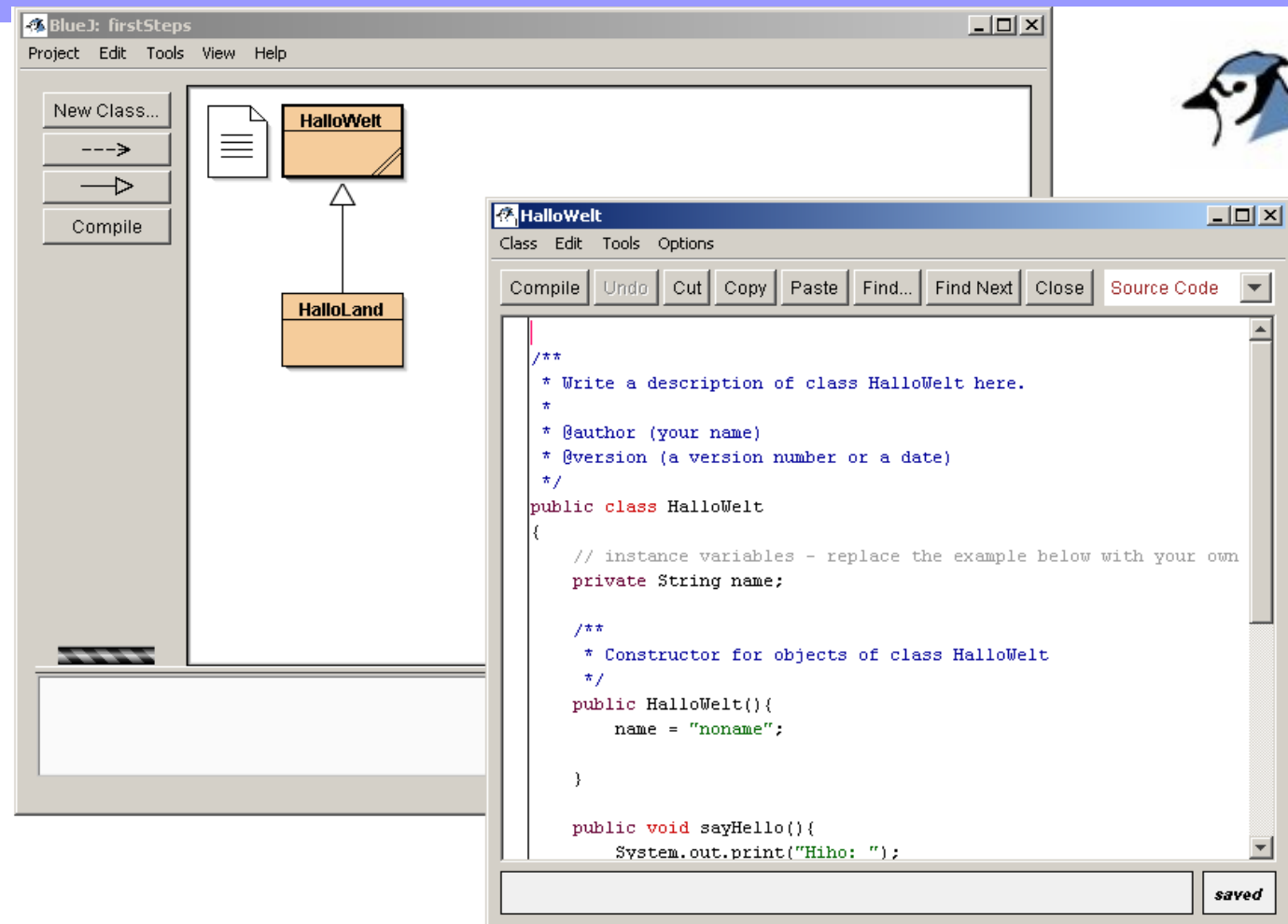
[http://bildungsserver.berlin-brandenburg.de/inf\\_pspr\\_bluej.html](http://bildungsserver.berlin-brandenburg.de/inf_pspr_bluej.html)

- Java API: <http://java.sun.com/javase/6/docs/api/>
- Tutorials: <http://java.sun.com/docs/books/tutorial/>
- Foren: <http://forums.sun.com/> (engl.)  
[www.java-forum.org/](http://www.java-forum.org/) (deutsch)

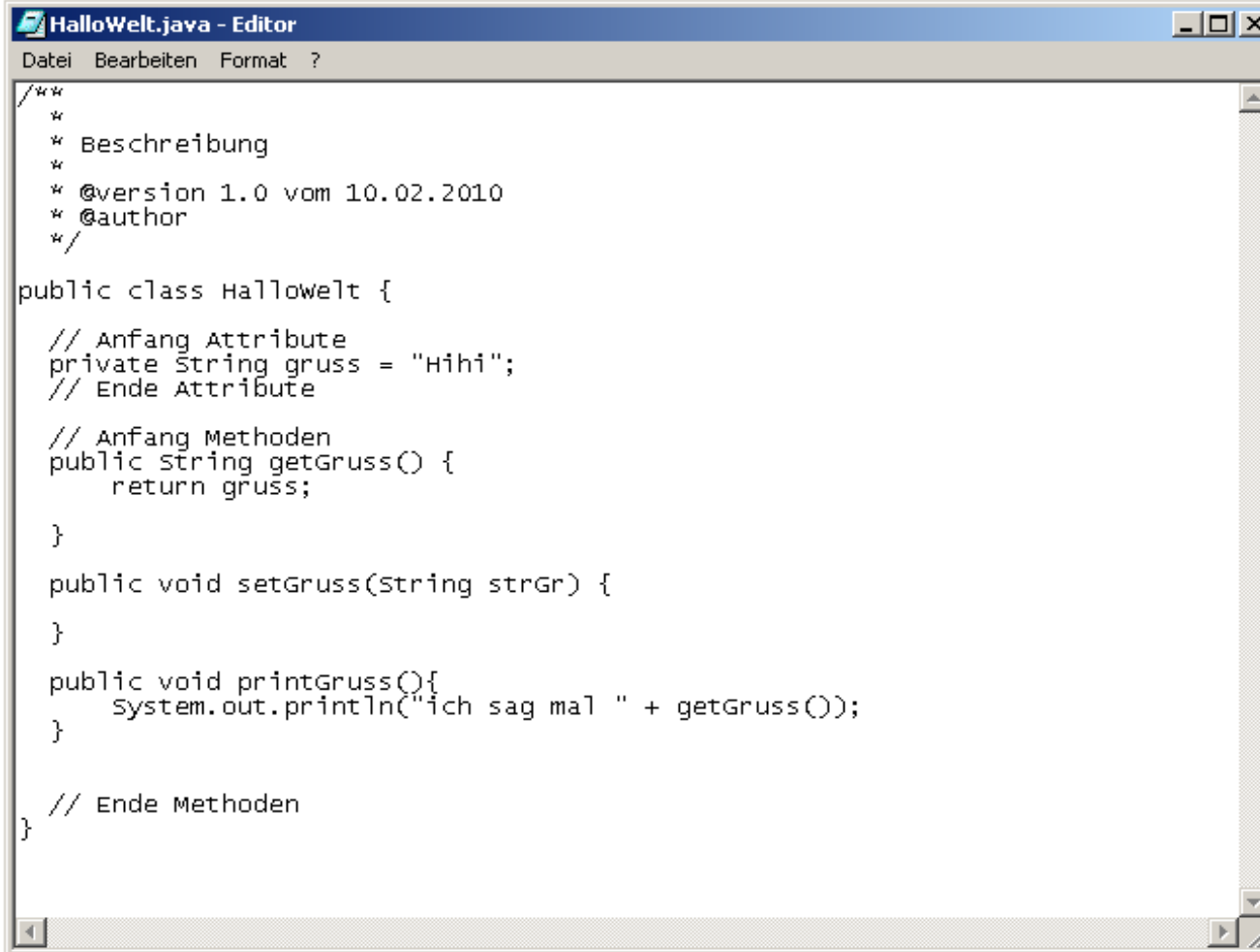


# BlueJ ver. 2.5.3 - Wieso BlueJ?

- reduziert
- UML
- OOD/ OOP
- Debugger
- Syntax highlighting



# BlueJ im Vergleich



```
HalloWelt.java - Editor
Datei Bearbeiten Format ?

/**
 *
 * Beschreibung
 *
 * @version 1.0 vom 10.02.2010
 * @author
 */

public class Hallowelt {

    // Anfang Attribute
    private String gruss = "Hihi";
    // Ende Attribute

    // Anfang Methoden
    public String getGruss() {
        return gruss;
    }

    public void setGruss(String strGr) {

    }

    public void printGruss(){
        System.out.println("ich sag mal " + getGruss());
    }

    // Ende Methoden
}
```



# BlueJ im Vergleich



```
C:\prjs\java\javaeditorTest\HaloWelt.java - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
LK13klausur3Tester.hs impressum.htm new 2 HaloWelt.java
1  /**
2   *
3   * Beschreibung
4   *
5   * @version 1.0 vom 10.02.2010
6   * @author
7   */
8
9  public class HaloWelt {
10
11     // Anfang Attribute
12     private String gruss = "Hihi";
13     // Ende Attribute
14
15     // Anfang Methoden
16     public String getGruss() {
17         return gruss;
18     }
19
20
21     public void setGruss(String strGr) {
22
23     }
24
25     public void printGruss(){
26         System.out.println("ich sag mal " + getGruss());
27     }
28
29
Java source file  nb char : 445  Ln : 1  Col : 1  Sel : 0  Dos\Windows  ANSI  INS
```

# BlueJ im Vergleich



```
package de.alexanderschindler.risi.gui;
import de.alexanderschindler.risi.*;

public class PanelPlanFreieTage extends JPanel{

    private String panelTitel = "";

    public PanelPlanFreieTage(String text, MainRisi mainR){
        panelTitel = text;
        init();
    }

    public void init(){

        //generatePdfPlan();

        GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints c = new GridBagConstraints();
        setLayout(gridbag);
    }
}
```

| Description                              | Resource     | Path                      | Location |
|--|--------------|---------------------------|----------|
| TODO : Sollte als Singleton umgesetzt we | MainRisi.jav | risiPlan2/src/de/alexande | line 34  |
| TODO als append!!!                       | DCRessourc   | risiPlan2/src/de/alexande | line 66  |
| TODO Auslagern in paintBelegungen(...    | MainRisi.jav | risiPlan2/src/de/alexande | line 424 |
| TODO buildGUI()                          | MainRisi.jav | risiPlan2/src/de/alexande | line 161 |
| TODO hier geht es weiter                 | PanelPlanBe  | risiPlan2/src/de/alexande | line 142 |
| TODO Planunszeitraum speichern           | MainRisi.iav | risiPlan2/src/de/alexande | line 143 |

# BlueJ im Vergleich



The screenshot displays the Java-Editor IDE interface. The main window shows a Java Swing window with a single button labeled "jButton1". The source code for "MeinFrame.java" is visible in the editor, showing the following code:

```
21  super(title);
22  setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CL
23  int frameWidth = 274;
24  int frameHeight = 238;
25  setSize(frameWidth, frameHeight);
26  Dimension d = Toolkit.getDefaultToolkit().getScreenSiz
27  int x = (d.width - getSize().width) / 2;
28  int y = (d.height - getSize().height) / 2;
29  setLocation(x, y);
30  Container cp = getContentPane();
31  cp.setLayout(null);
32  // Anfang Komponenten
33
34  jButton1.setBounds(24, 40, 107, 25);
35  jButton1.setText("jButton1");
```

The "Objekt-Inspektor" (Object Inspector) panel shows the following attributes for the "MeinFrame: JFrame" object:

| Attribute | Ereignisse |
|-----------|------------|
| Height    | 238        |
| Name      | MeinFrame  |
| Width     | 274        |
| X         | 0          |
| Y         | 0          |

The status bar at the bottom indicates "Zeile: 24 Spalte: 1" and "Einf ANSI/Dos".

# BlueJ im Vergleich



| <i>PRO</i>                 | <i>Contra</i>                 |
|----------------------------|-------------------------------|
| sinnvoll reduziert         | GUI-Erstellung schwierig      |
| gute OOP/OOD Unterstützung | keine Code Completion         |
| Visualisierung             | keine volle OOM Unterstützung |
| kostenlos                  | nur für Anfangsunterricht     |

==> Besondere Eignung für die Schule

- Download: [www.bluej.org/](http://www.bluej.org/)
- Voraussetzung: installiertes JDK 6

<http://java.sun.com/javase/downloads/index.jsp> oder JDK 5

# Grundlegende Techniken der OO

## Erstellen eines Objektes mit BlueJ

- BlueJ starten
- Neues Projekt anlegen: Project → New Project
- Klasse anlegen: Buttonclick New Class → Name vergeben (beginnend mit Großbuchstaben)
- Rechte Maustaste auf Klasse → Compile
- Rechte Maustaste auf Klasse → new <KlassenName>

**Objekt wird erstellt.**

Code hinzufügen: `System.out.println ("Hallo Welt !");`

# OO – Was ist ein Objekt?





# OO – Was ist ein Objekt?



Bild einer Tür



Bild einer Kaffeemaschine

# Ü - Turtle

## Arbeiten mit der Turtle:

- Turtle.Java in Projektverzeichnis kopieren
- BlueJ neu starten
- neues Objekt der Klasse Turtle anlegen mit *Kontextmenu: new Turtle()*
- Methoden des Objektes benutzen mit: *Kontextmenu des Objektes...*

---

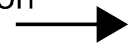
**Aufgabe:** Erstellen Sie eine kleine sinnvolle geometrische Figur.

# Ü - Turtle

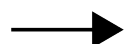
## Anlegen eines Turtle Objektes „von Hand“

- Legen Sie eine eigene Methode in einer eigenen Klasse an.
- Aufbau einer Methode:

Methodendeklaration



Methodenrumpf



```
public Object verschiebeObject (Objekt NameDesObjektes)
{
    //Hier passiert nichts.
    return NameDesObjects;
}
```

- Methodendeklaration:

Acces Level

Rückgabewert

Methodenname

Parameter



```
public Object verschiebeObject(Object NameDesObjektes)
```

# Ü - Turtle

```
Bsp:  public void steuereTurtle(){  
        // ...  
    }
```

- Erstellen eines Turtle Objektes:

```
Turtle meineKroete = new Turtle();
```

- Steuerung der Turtle über ihre Methoden:

```
meineKroete.move(10, 200);
```

```
meineKroete.penDown();
```

```
meineKroete.go(100);
```

```
meineKroete.rotate(90);
```

...USW.

# Ü - Turtle

---

**Aufgabe:** Erstellen Sie ein Quadrat.

( Sollten Sie noch Zeit haben, lassen Sie die Turtle doch das Haus des Nikolaus zeichnen. )

---

Turtle bietet einen ersten Einstieg in die Benutzung von **Klassen** und **Methoden**.

# OO – Klassen

Mindestanforderungen an eine Klasse

```
public class Kaffeemaschine {  
    //Ein Constructor kann ergänzt werden  
    public Kaffeemaschine(){  
    }  
}
```

Klassen besitzen Methoden und Eigenschaften!

# OO – Methoden...

```
public class Kaffeemaschine {  
    //Ein Constructor kann ergänzt werden  
    public Kaffeemaschine() {  
    }  
  
    public void turnOn() {  
    }  
  
    public void turnOff() {  
    }  
}
```

....wie wird gespeichert, ob die Kaffeemaschine an oder aus ist?

# OO – ...und Eigenschaften

```
public class Kaffeemaschine {  
    private String zustand;  
  
    public void turnOn(){  
        zustand = "an";  
    }  
    public void turnOff(){  
        zustand = "aus";  
    }  
}
```



# OO – Klassen und Objekte

- In einer Klasse werden Methoden und Eigenschaften festgelegt.
- Eine Klasse kann der „Bauplan“ von beliebig vielen Objekten sein.
- Erst im Objekt können die Methoden und Eigenschaften benutzt werden.

## Instantiierung einer Klasse (kurze Schreibweise):

```
Turtle meineKroete = new Turtle();
```

## Instantiierung einer Klasse (lange Schreibweise):

```
Turtle meineKroete;           // Deklaration  
meineKroete = new Turtle();   // Instantiierung
```

# OO – Methoden und Eigenschaften

## Methoden

- In den Methoden befindet sich die eigentliche Programmlogik.
- Methoden haben immer einen beschreibenden Namen.
- Methoden bilden einen (kleinen) Teil einer Problemlösung ab.
- Man kann Werte an Methoden übergeben.
- Methoden können Werte zurückgeben.
- Methoden bilden kleine wiederverwendbare Einheiten.

## Eigenschaften

- Klassen und ihre Objekte können Eigenschaften besitzen.
- Eigenschaften sind Variablen, denen man bestimmte Werte zuweist.

# OO – Sichtbarkeit / Access level

## Deklaration

1. lokal
2. - *private*
3. ~ package (ohne Angabe)
4. # `protected`
5. + `public`

## Zugriff

innerhalb der Methode  
Klasse  
Klasse, Package  
Klasse Package, Subklasse  
alle

# OO – Rückgabewert

**void** oder ein beliebiger Typ

**Bsp:**

```
public String getZustand(){  
    return zustand;  
}
```

```
public void setZustand(String neuerZstd){  
    zustand = neuerZstd;  
}
```

# Ü – Projekt: Motorrad

---

## Aufgabe:

- Erstellen Sie eine Klasse Motorrad.
- Überlegen Sie, welche Eigenschaften ihr Motorrad besitzen soll, und setzen Sie diese entsprechend um
- Entwickeln Sie die passenden Methoden dazu. Benutzen Sie getter und setter.
- Geben Sie die jeweiligen Eigenschaften mit den entsprechenden Methoden aus. Anweisung:  
`System.out.println ("Hallo Welt!");`

# OO – Vererbung

Die Vererbung gehört zu den wichtigen Konzepten der objektorientierten Programmierung.

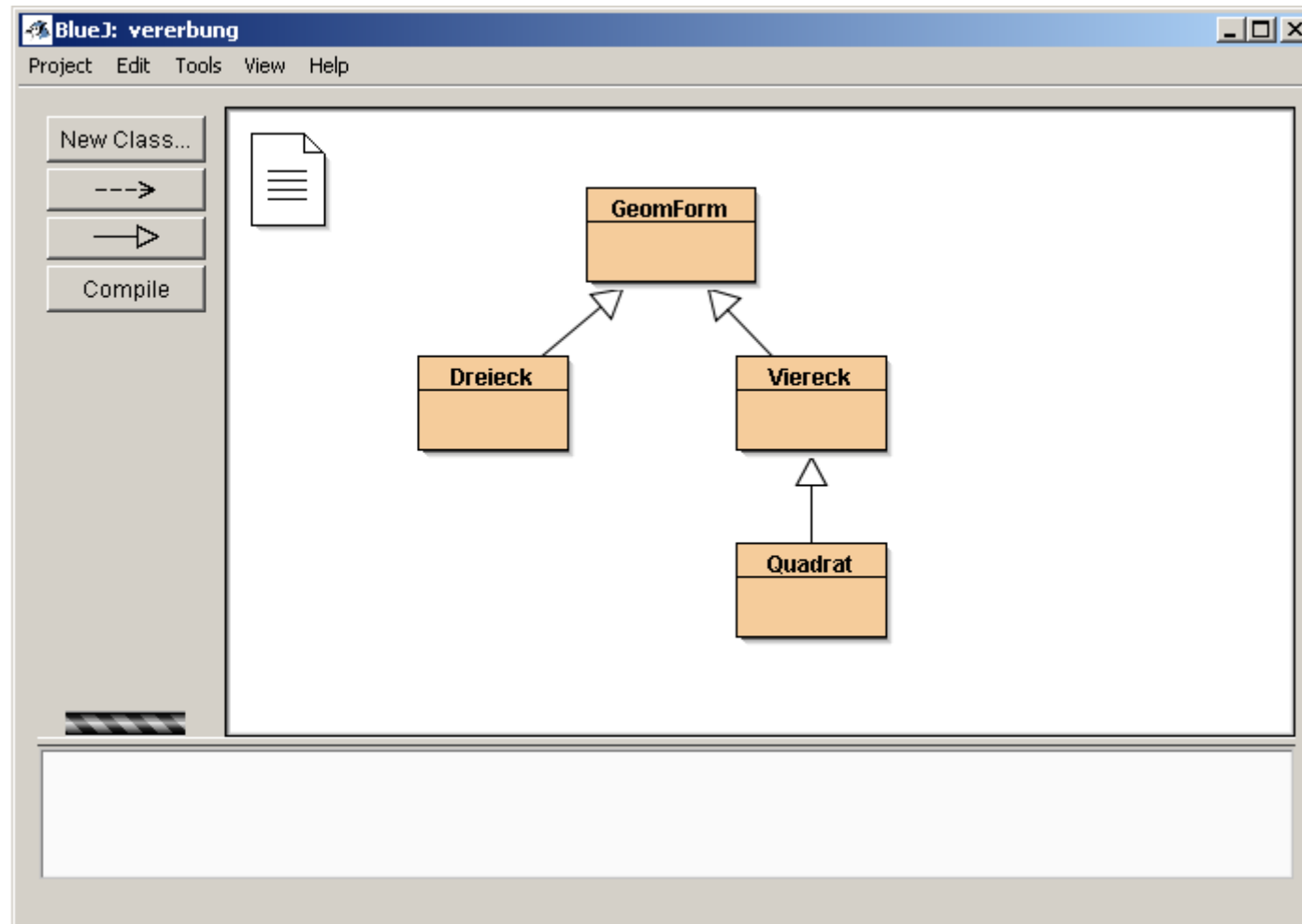
**Idee:** Eine Oberklasse vererbt ihre Methoden und Eigenschaften an andere Klassen weiter.

**Vorteile:** Mehrere Klassen können von einer anderen profitieren.

**Bsp:**

```
public class Viereck extends GeomForm {  
    // ...  
}
```

# OO – Vererbung



# Ü Vererbung - Projekt: Fahrzeuge

---

## Aufgabe:

- Erstellen Sie eine Vererbungshierarchie zum Thema Fahrzeuge.
- Entwickeln Sie die entsprechenden Eigenschaften und Methoden auf den unterschiedlichen Ebenen der Hierarchie.
- Erstellen Sie immer wieder Objekte ihrer Klassen und überprüfen Sie die Vererbung der Methoden und Eigenschaften.



# Objects first! ?

## Diskussion

| <i>Pro</i> | <i>Contra</i> |
|------------|---------------|
| ...        | ...           |

**Objects first?**

**...Objects soon!**

# Unterrichtsbeispiele

share: \Material\docs\UnterrichtsBspSchindler

# Literatur – Links – Quellen

auf y: (share)

- BlueJ
- JDK 6 mit API
- KRÜGER, G., T. STARK [2007]: Handbuch der Java Programmierung
- Java Referenzcard
- DIETZ, MÜLLER, PUNKENBERG [2006]: Java AB

# Literatur – Links – Quellen

## BlueJ

- BARNES, D, M. Kölling [2009]: Objects First with Java

## Java

- FLANAGAN, D. [2002]: Java in a nutshell
- FLANAGAN, D. [2004]: Java examples in a nutshell
- MIDDENDORF, S. [2002]: Java Programmierhandbuch und Referenz
- MITCHELL, W. D. [2000]: Debugging Java