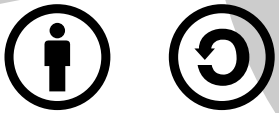
Javakurs 2013 – Objektorientierung

Objektorientierte Programmierung I

Armelle Vérité

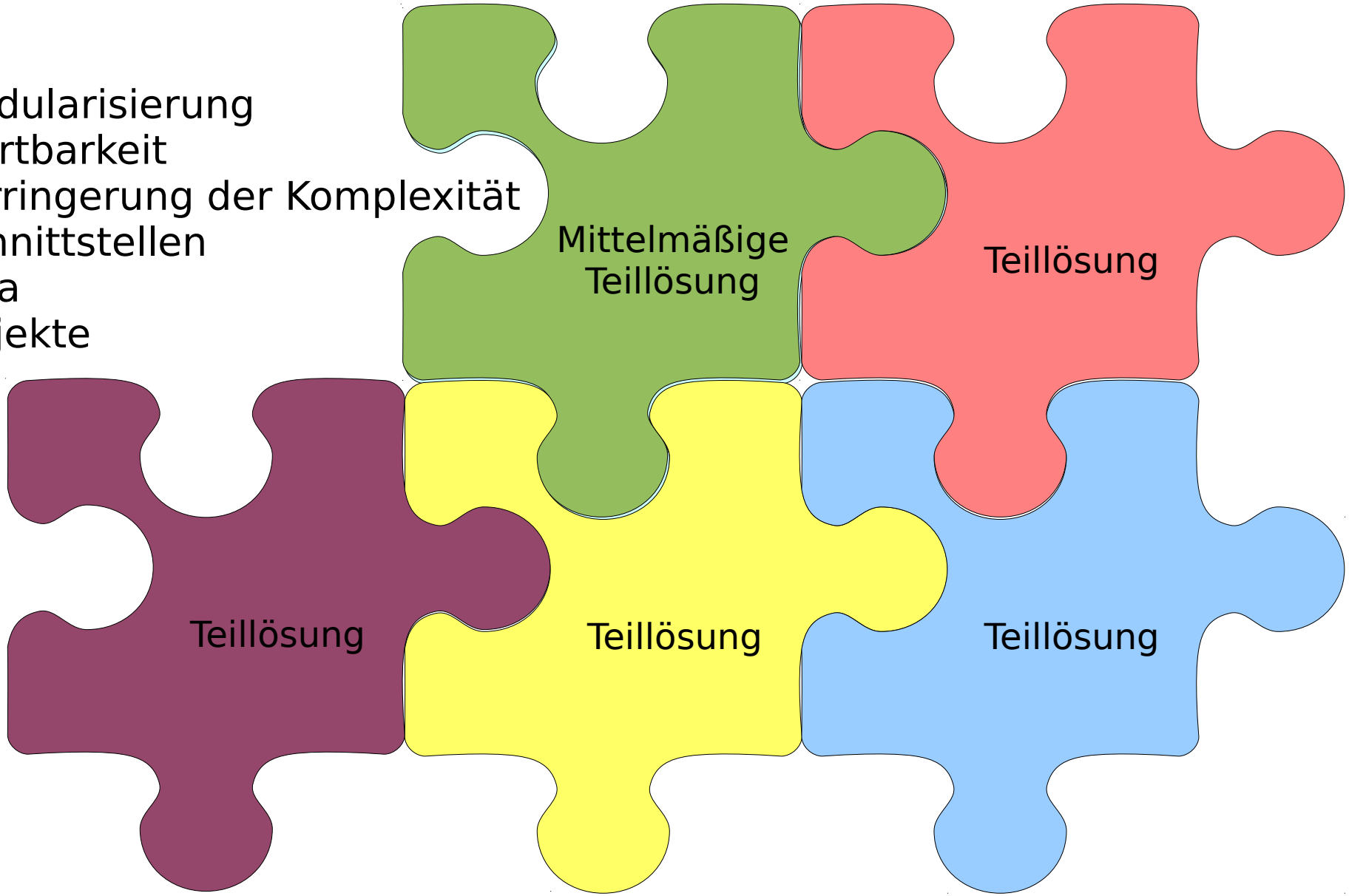
7 März 2013

Technische Universität Berlin

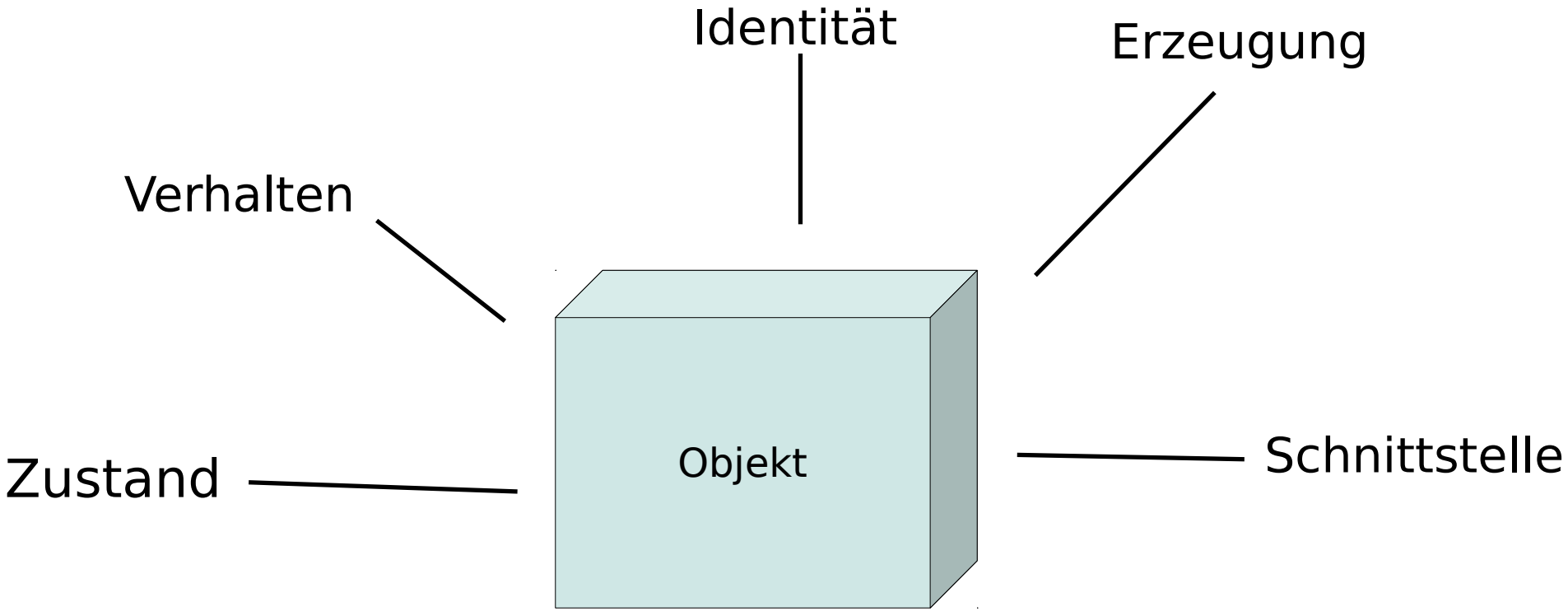


Objektorientierte Programmierung

- Modularisierung
- Wartbarkeit
- Verringerung der Komplexität
- Schnittstellen
- Java
- Objekte



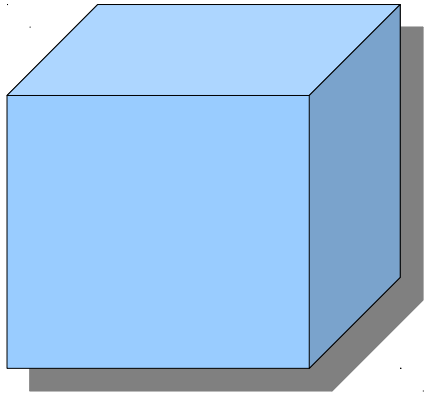
Was macht einen Objekt aus?

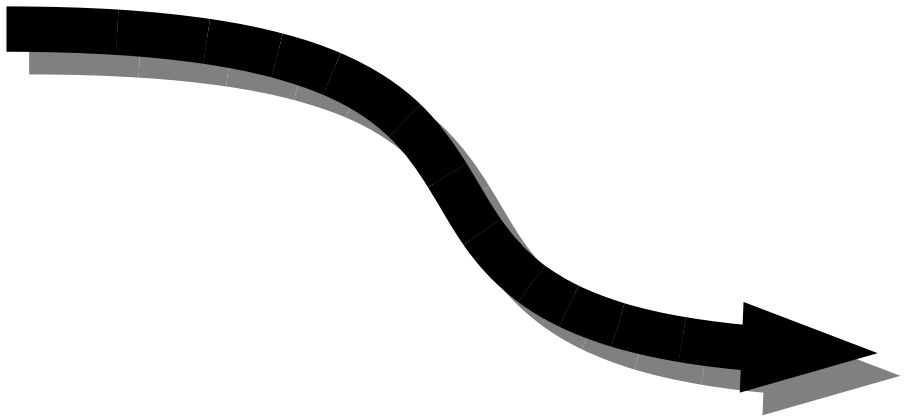


Was davon könnte kein Java Objekt sein?

1010110100101010

Eine Folge von Nullen und Einsen...
Eine dreidimensionale Form...
Eine Verbindung...
Die Zahl 42...

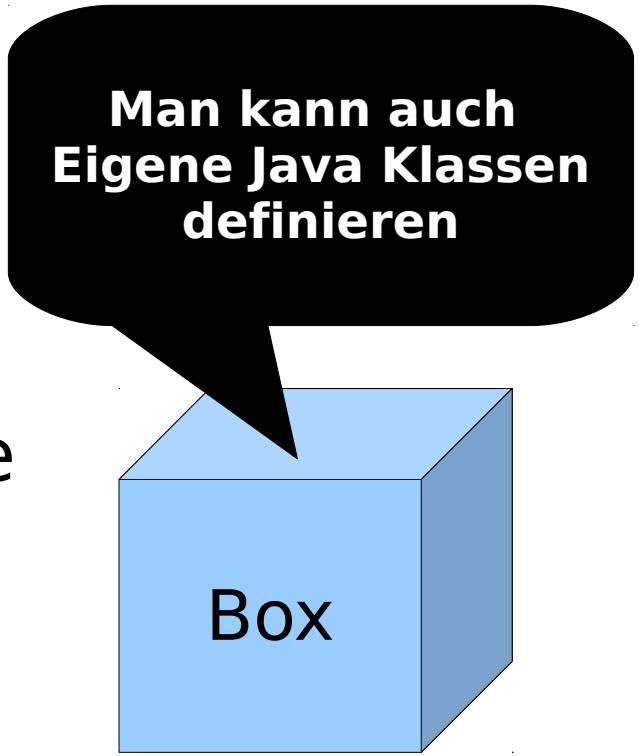




42

Antwort : alle könnten Objekte sein!

java.lang.Byte

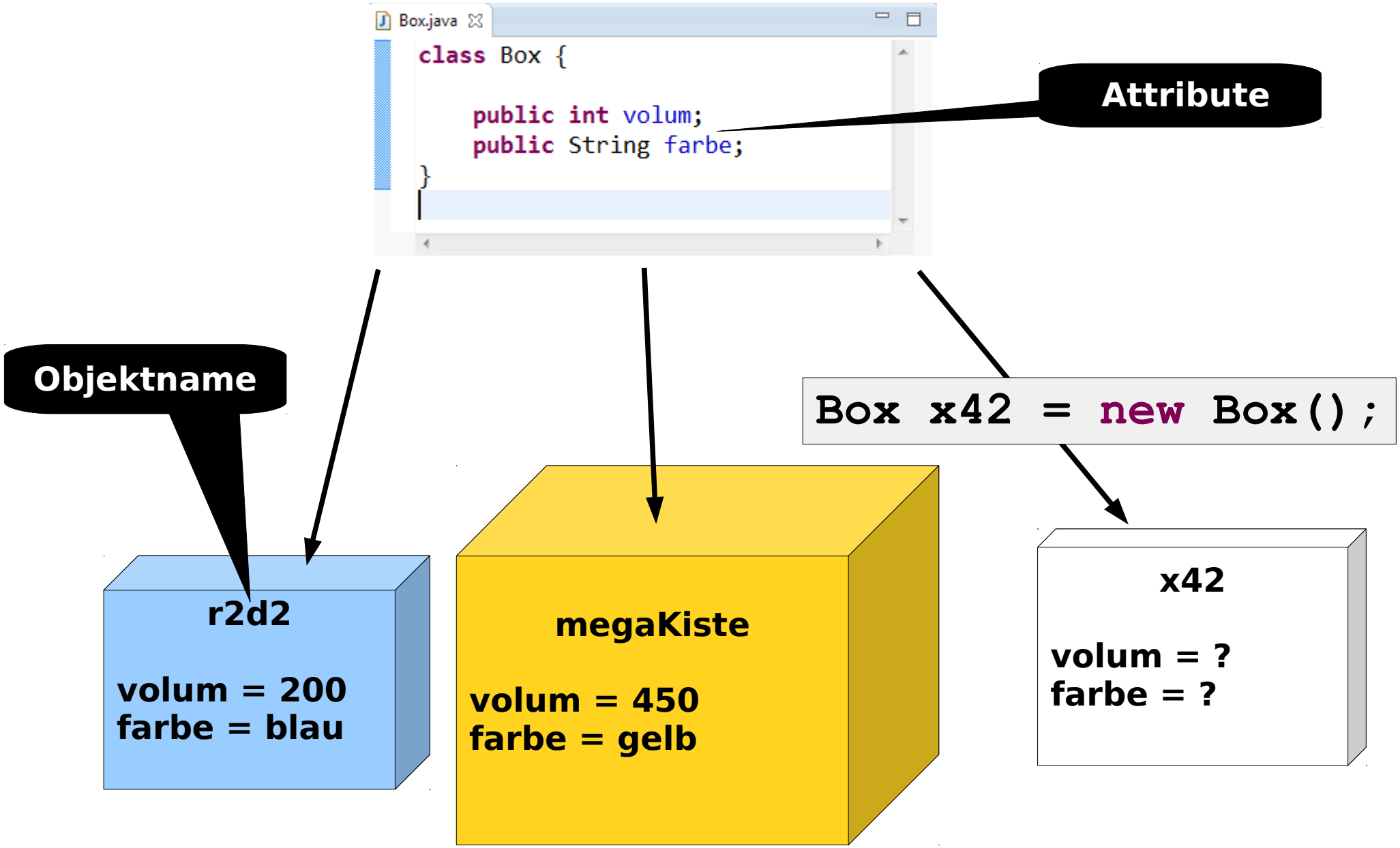


Ein Objekt ist ein Objekt...
Wenn es eine java **Klasse** gibt, die dieses Objekt erzeugt

java.sql.Connection

java.lang.Integer

Ein Objekt ist eine Instanz einer Klasse

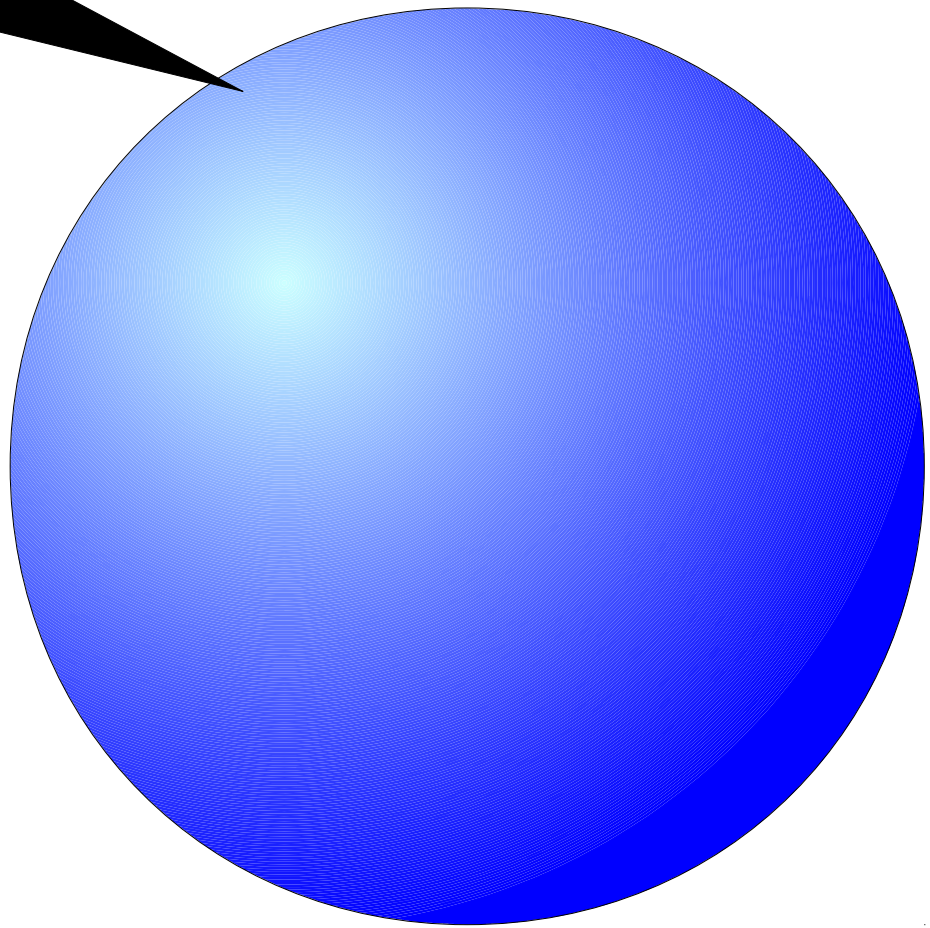


Abstraktion

Bin ich eine Instanz von Box?

Box.java

```
class Box {

    public int volum;
    public String farbe;
}
```



Arbeiten mit mehreren Klassen

Box.java

```
import java.awt.Color;

class Box {

    public String volum;
    public Color farbe;
}
```

Box.java | Schuh.java

```
public class Schuh {
    public Box verpackung;
    public int modellNr;

    public static void main(String[] args) {
    System.out.println( "I love new shoes!");
    }
}
```

```
$ javac Schuh.java
$ ls
Box.class     Box.java      Schuh.class  Schuh.java
$ java Schuh
```

Variablen vor dem Zugriff definieren

Box.java | Schuh.java | Schuhladen.java

```
public class Schuhladen {
    public Schuh[] sortiment;
    public String adresse;

    public String printRechnung(int i){
        String Rechnung = adresse + sortiment[i].modellNr;
        return Rechnung;
    }
}
```

Box.java | Schuh.java | Schuhladen.java | Main.java

```
public class Main {

    public static void main(String[] args) {
        Schuhladen shop = new Schuhladen();
        shop.printRechnung(42);
        }
}
```

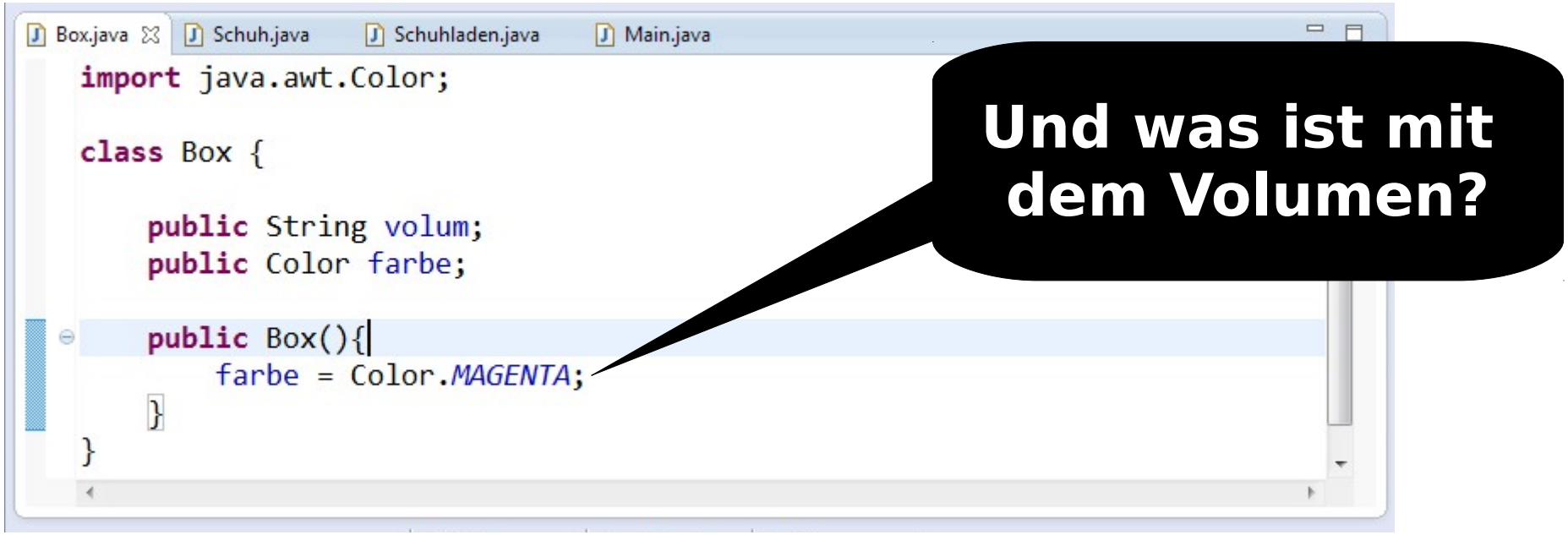
Console

```
<terminated> Main [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (04.03.2013 13:00:15)
Exception in thread "main" java.lang.NullPointerException
        at Schuhladen.printRechnung(Schuhladen.java:6)
        at Main.main(Main.java:6)
```

Standard-Konstruktor

Erzeugung eines Objektes:

```
Box x42 = new Box();
```

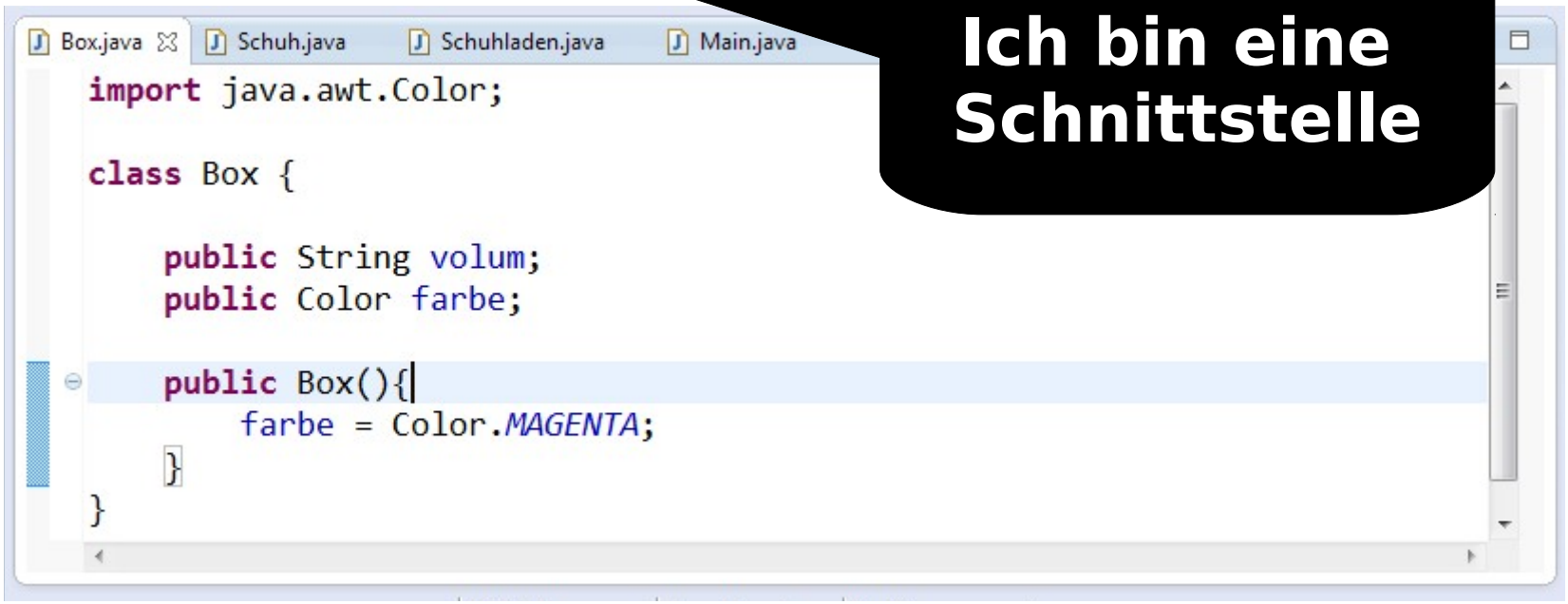


Eclipse Tipp:
Source / Generate Constructor using Field

Punktnotation

Definieren von Schnittstellen mit dem Schlüsselwort public:

```
Box x42 = new Box();
 x42.volum = 420;
```



Konstruktoren mit Parametern

Erzeugung eines Objektes mit Parametern:

```
Box Schuhkarton = new Box(630,Color.RED);
```

Box.java | Schuh.java | Schuhladen.java | Main.java

```
import java.awt.Color;

class Box {

    public int volum;
    public Color farbe;

    public Box(int volum, Color farbe){
        this.volum = volum;
        this.farbe = farbe;
    }
```

Zusammenfassung Konstruktoren

- Konstruktoren sind Methoden zum Initialisieren von Objekten
- Name = Klassenname
- Keine Angabe zum Rückgabewert
- Wird kein Konstruktor definiert, existiert ein impliziter Konstruktor ohne Parameter
- Mehrere Konstruktoren in einer Klasse möglich, wenn sie sich an die Parametern unterscheiden
- Eclipse-Tipp: Source → Generate Constructor

Zugriff auf public Attribute

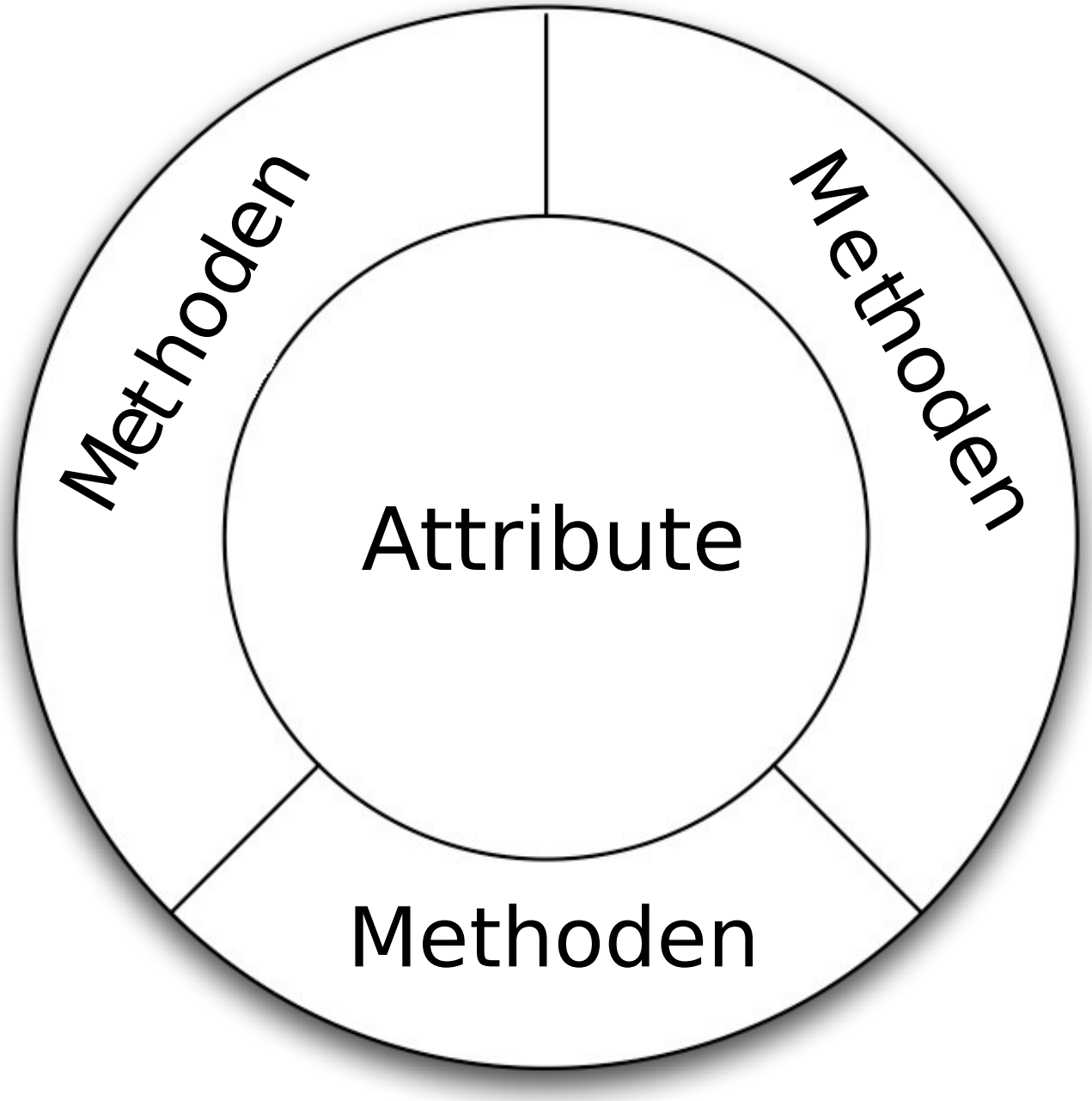
Wir verlassen uns darauf, dass alles gut ge

Box.java | Schuh.java | Schuhladen.java | Main.java

```
import java.awt.Color;

class Box {

    public int volum;
    public Color farbe;

    public Box(){
        farbe = Color.MAGENTA;
    }

    public Box(int volum, Color farbe){
        this.volum = volum;
        this.farbe = farbe;
    }
}
```

```
Box schwarzesLoch = new Box();
SchwarzesLoch.volum = -42;
```

Die Idee von Kapselung



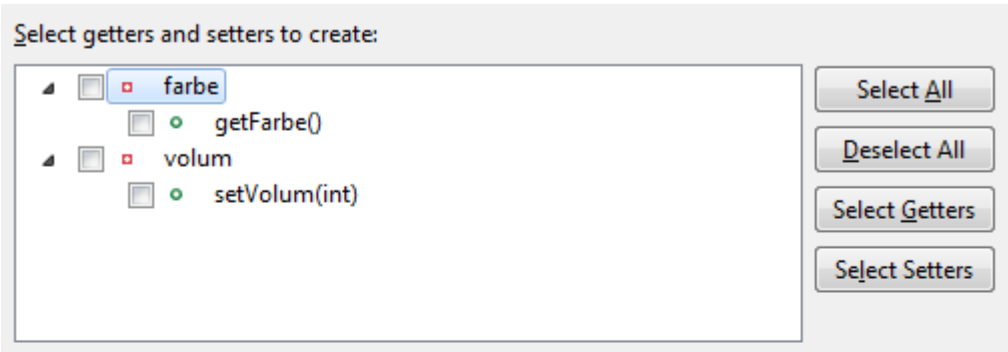
Getter und Setter Methoden

Box.java | Schuh.java | Schuhladen.java | Main.java

```
import java.awt.Color;

class Box {

    private int volum;
    private Color farbe;

    public int getVolum() {
        return volum;
    }
    public void setFarbe(Color farbe) {
        this.farbe = farbe;
    }
    public void paintBlue(){
        setFarbe(Color.CYAN);
    }
```

PRIVATE

Eclipse Tipp:
Source / Generate Getter and Setter



Frage

Wir wollen einen Attribut der Klasse Box definieren, der angibt, wie viele Instanzen von Box existieren.

Wie machen wir das?

Klassenvariable = gehört zur Klasse selbst

Box.java | Schuh.java | Schuhladen.java | Main.java

```
import java.awt.Color;

class Box {

    private int volum;
    private Color farbe;
    private static int total = 0;

    public Box() {
        farbe= Color.MAGENTA;
        total ++;
    }

    public Box(int volum, Color farbe) {
        super();
        this.volum = volum;
        this.farbe = farbe;
        total ++;
    }
```

Utilities sind oft static Methoden

```
int wuerfel;
wuerfel= (int)Math.round((Math.random()*5)+1);
```

Math

public static long **round**(double a)

Returns the closest long to the argument.
The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long.
In other words, the result is equal to the value of the expression:
(long)Math.**floor**(a + 0.5d)

public static double **random**()

Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

Zusammenfassung *static*

statische Methoden	**nicht-statische Methoden**
• Klassen-Methoden	• Instanz-Methoden
• nur Zugriff auf Klassenvariablen	• Zugriff auf Klassen- und Instanzvariablen
• Können direkt auf Klasse aufgerufen werden	• erfordern Erstellung eines Objektes

Zusammenfassung

- Attribute speichern Zustände eines Objektes
  - Konstanten mit Schlüsselwort **final**
  - Format der Deklaration :
    - Sichtbarkeit (private, public)
    - Schlüsselwörter (final, static)
    - Typ (int, String, Box ...)
    - Name (klein & darf nicht mit einer Zahl beginnen)
- Methoden implementieren Verhalten eines Objektes
  - Getter und Setter für private Attribute
  - Format der Deklaration :
    - Sichtbarkeit (private, public)
    - Rückgabetyp (int, String, Box...) außer bei Konstruktoren
    - methodenName (ParameterTyp parameterName)

Reference / value

```
int a = 0;
int b = a;

a++;

System.out.println(a);
System.out.println(b);
```

Ergebnis:

a: 1
b: 0

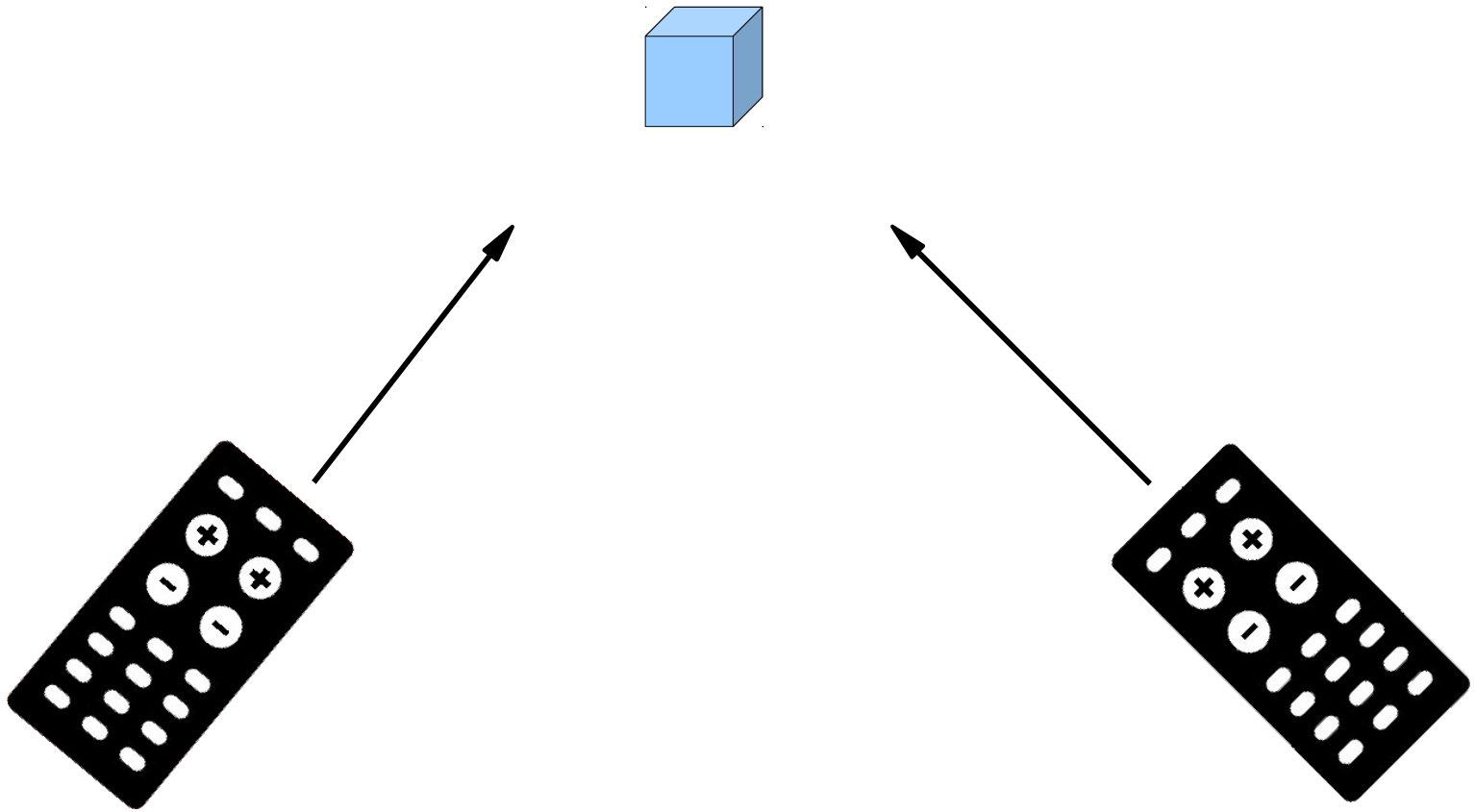
```
Box a = new Box(100, Color.ROT);
Box b = a;

a.volum++;

System.out.println(a.volum);
System.out.println(b.volum);
```

Ergebnis:

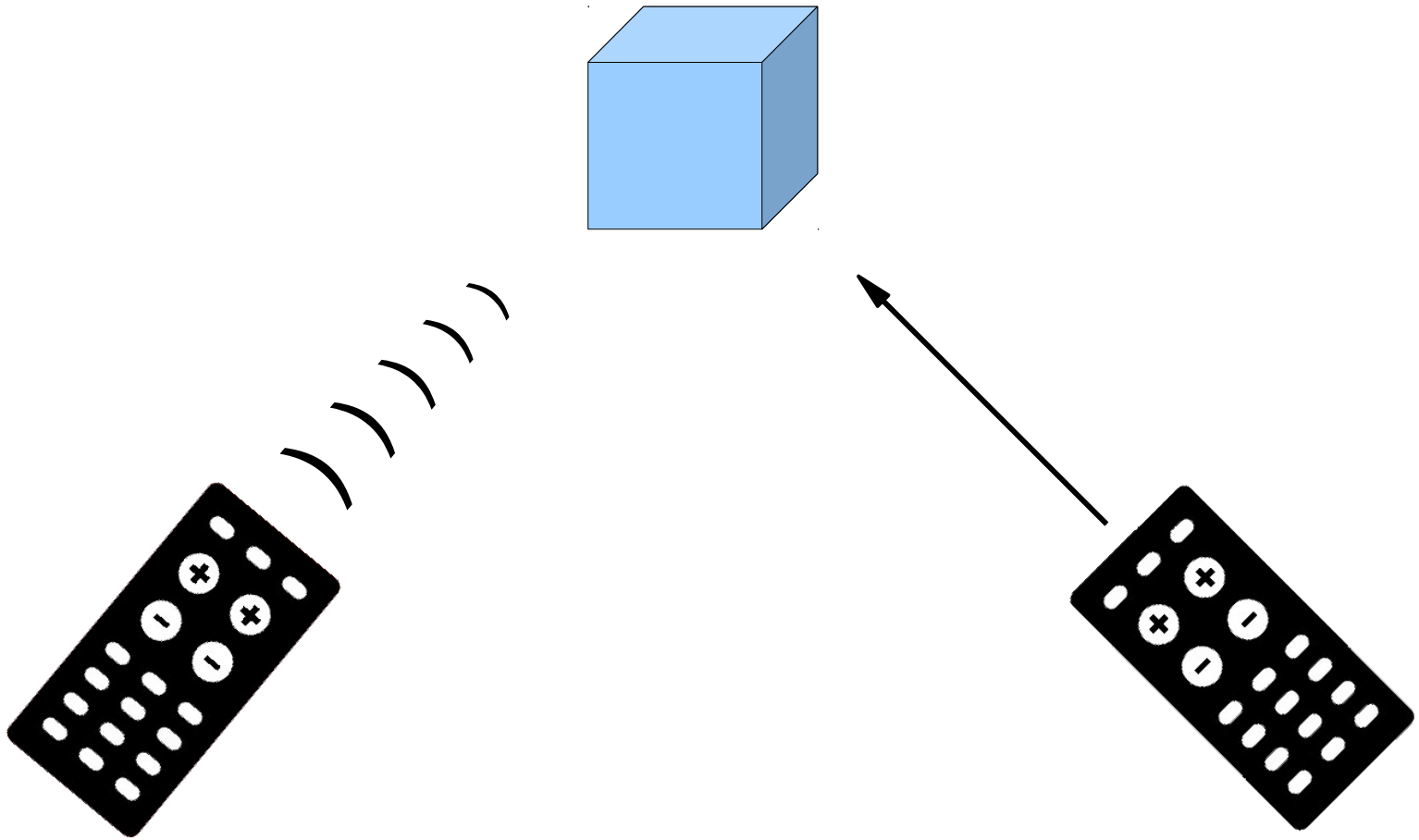
a.volum: 101
b.volum: 101

(public volum wird angenommen)

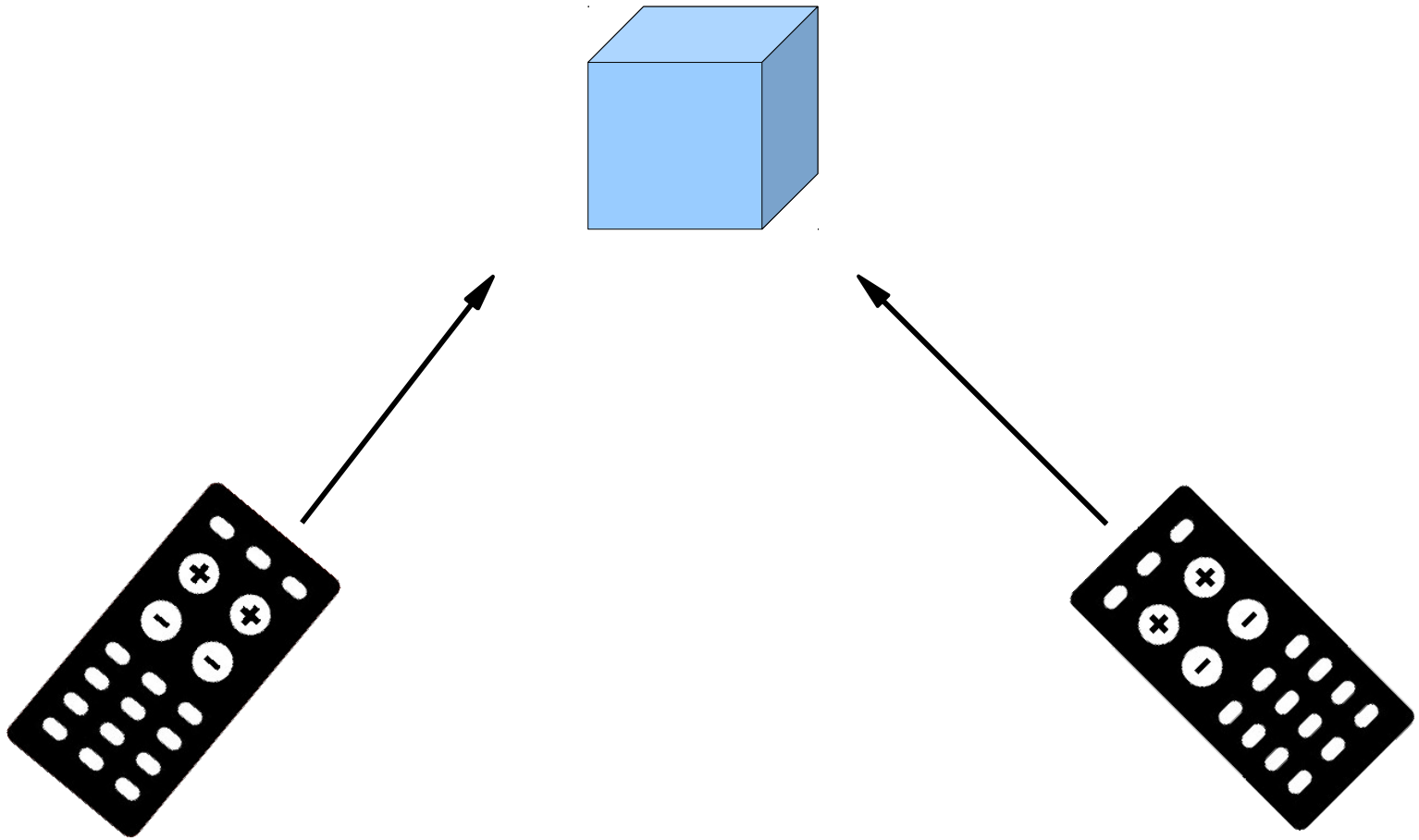
Referenzen



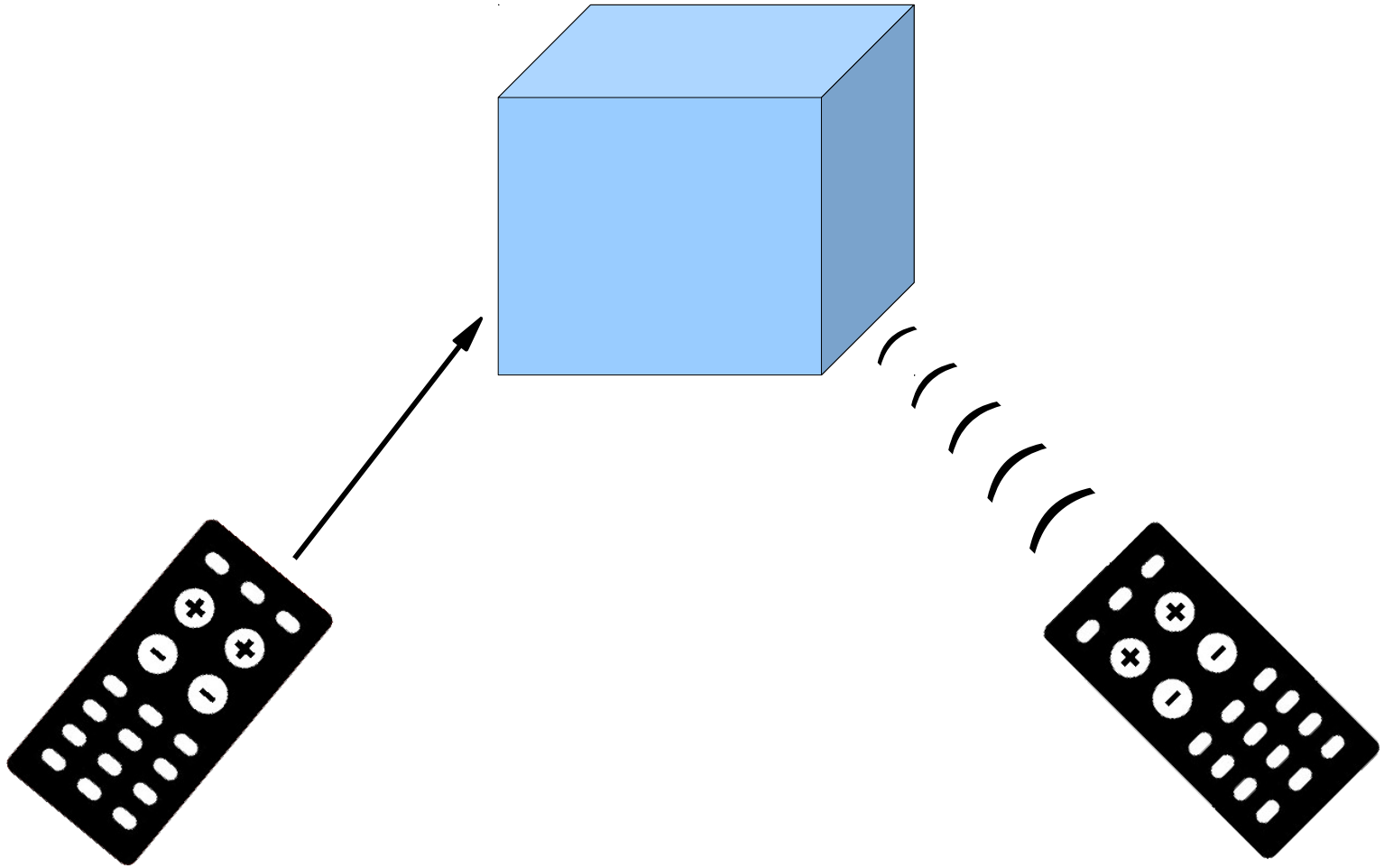
Referenzen



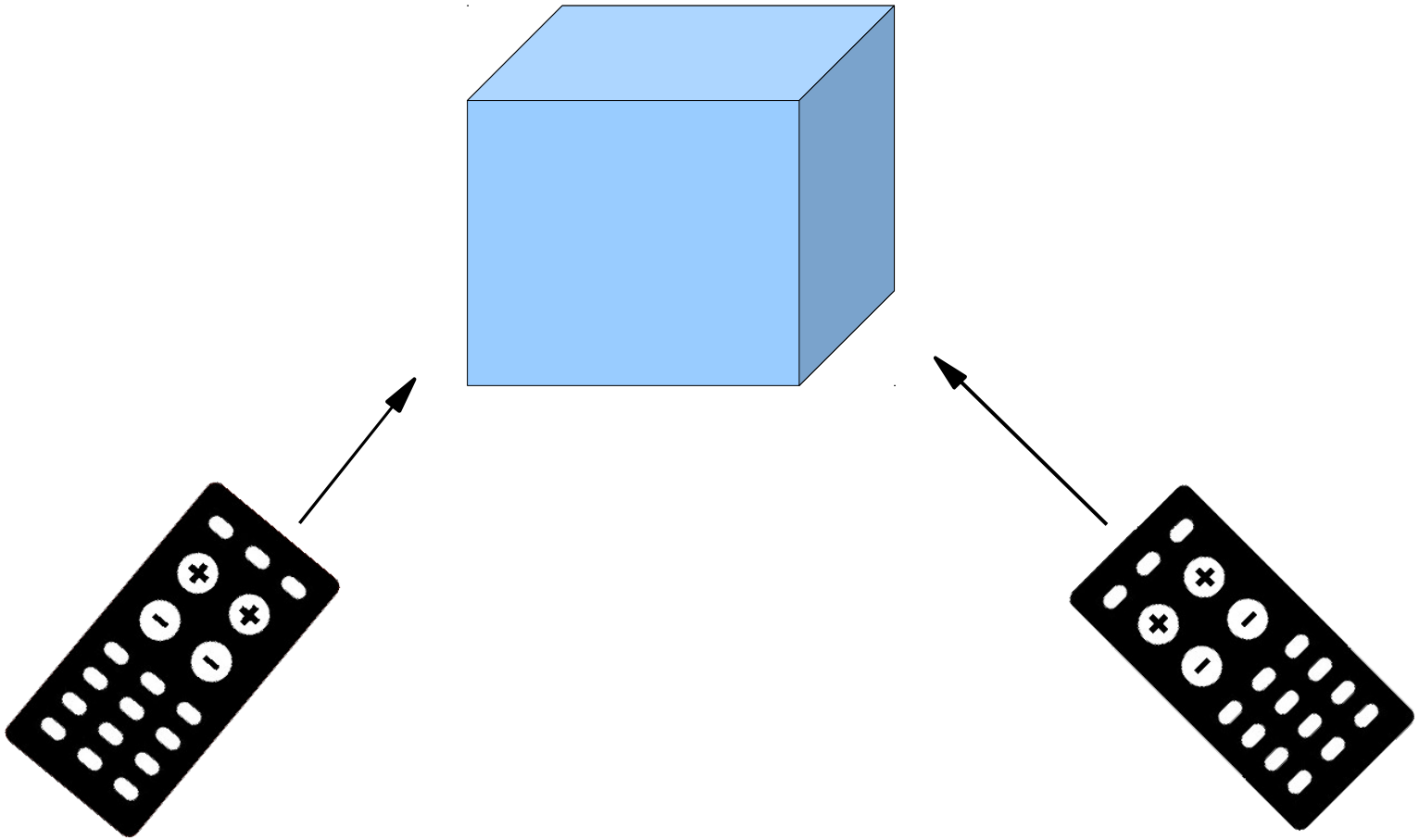
Referenzen



Referenzen



Referenzen



Häufige Compiletime- und Runtime-Fehler

```
1  Box mystery;
2  mystery.getFarbe();
```
Compiletime Fehler 

```
3  mystery = null;
4  mystery.paintBlue();
```
Runtime Fehler 

```
5  Box alias = mystery;
6  alias.getVolum();
```
Runtime Fehler 

```
7  Box bonbons = new Box();
8  bonbons.paintBlue();
```


```
9  System.out.print(100 / bonbons.volum);
```


```
10 Box geschenk = new Box(50, Color.ORANGE);
11 geschenk.getVolum();
```


```
12
13 Box ueberraschung = geschenk;
   ueberraschung.setFarbe(Color.PINK);
```


Merken:

genau eine Klasse pro Datei, Dateiname identisch mit Klassennamen, Klassennamen groß, fängt mit Buchstabe an, Eine Main Methode pro Projekt und...

Benutzt Eclipse!