Java Kurs

Objektorientierung II & die Java Klassenbibliothek

Kristian Bergmann und Arthur Lochstampfer

Vergleich

```
class Apfel {
  String farbe;
  int gewicht;
  String geerntetIn;

  void essen() {
    gewicht = 0;
  }

  void schaelen() {
    gewicht = gewicht - 10;
  }
}
```

```
class Erdbeere {
  String farbe;
  int gewicht;
  String geerntetIn;

  void essen() {
    gewicht = 0;
  }

  void entferneStiel() {
    gewicht = gewicht - 1;
  }
}
```

Vererbung - Inheritance

```
class Obst {
  String farbe;
  int gewicht;
  String geerntetIn;

  void essen() {
    gewicht = 0;
  }
}
```

```
class Apfel
          extends Obst {

  void schaelen() {
    gewicht = gewicht - 10;
  }
}
```

```
class Erdbeere
          extends Obst {

  void entferneStiel() {
    gewicht = gewicht - 1;
  }
}
```

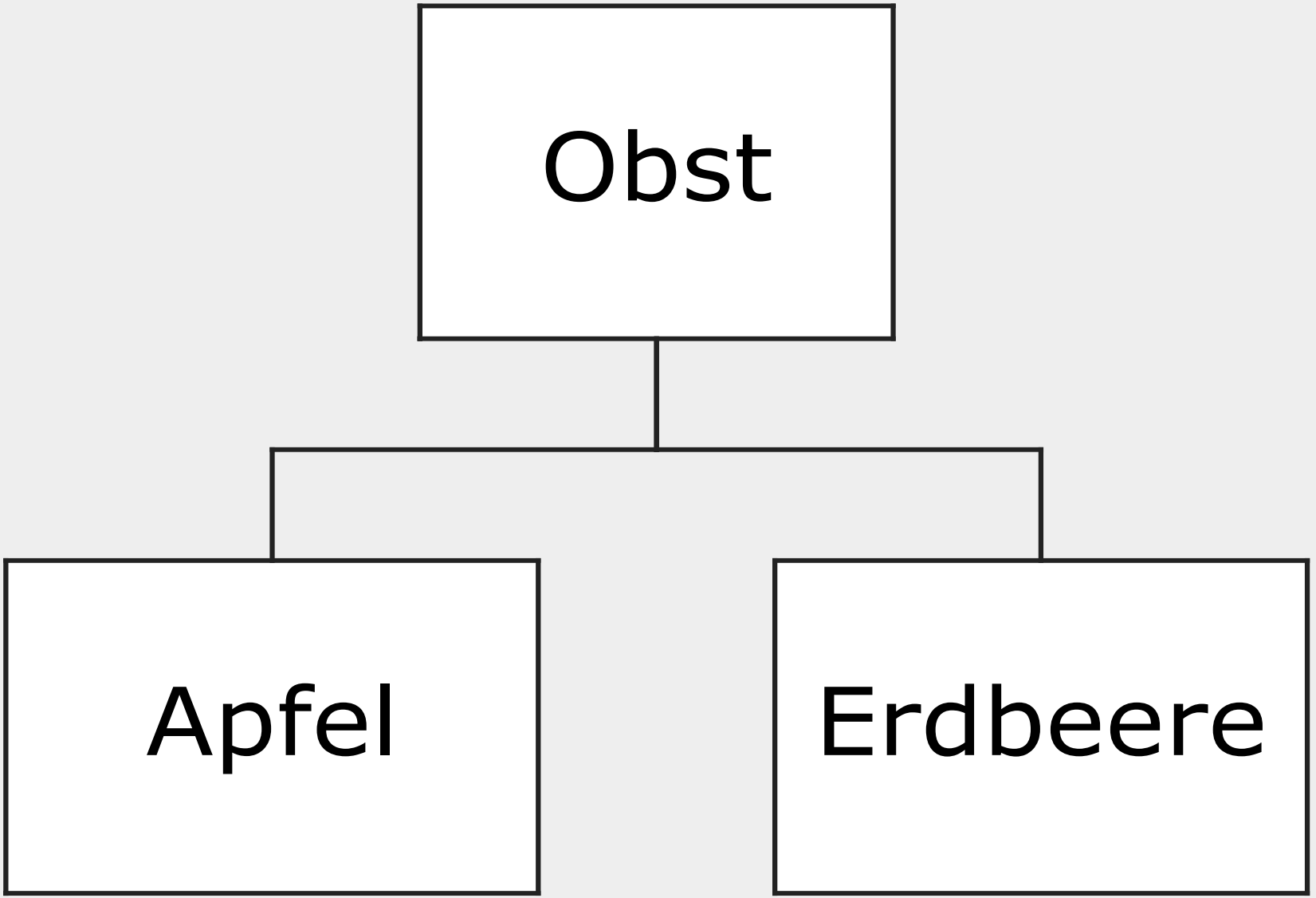
Geerbtes benutzen

```
class Obst {
  String farbe;
  int gewicht;
  String geerntetIn;

  void essen() {
    gewicht = 0;
  }
}
```

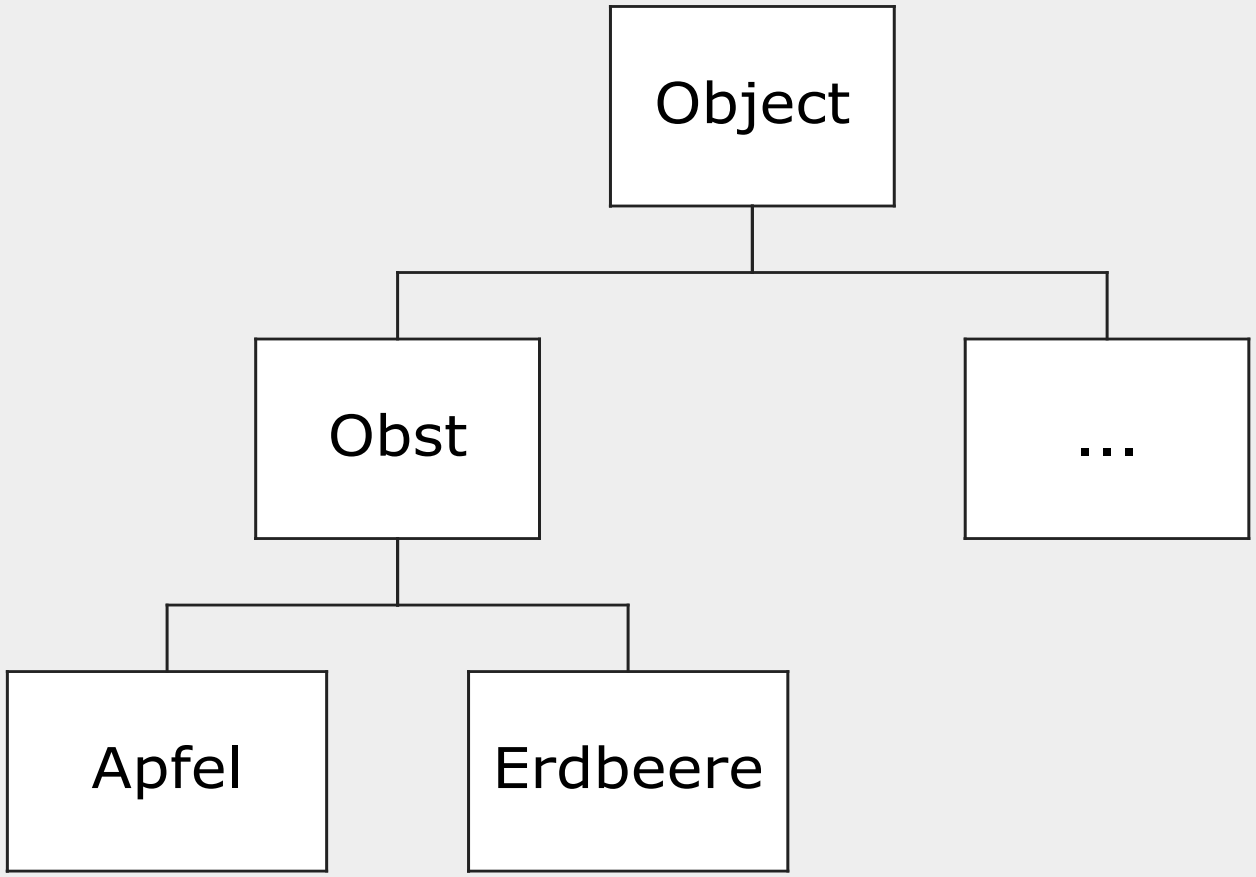
```
class Apfel
          extends Obst {

  void schaelen() {
    gewicht = gewicht - 10;
  }
}
```

```
public static void main ( String[] arguments ) {
  Apfel boskop = new Apfel();
  boskop.farbe = "rot";
  boskop.gewicht = 200;
  boskop.essen();
}
```

Ableitungsbaum



Alles ist ein „Object“



Folgen

- **Jede Klasse erbt einige Standard-Methoden:**
  - `String toString()`
  - `boolean equals(Object o)`
  - `...`

```
public static void main ( String[] arguments ) {
  Apfel boskop = new Apfel();

  String beschreibung = boskop.toString();
  System.out.println( beschreibung );
}
```

Ausgabe: `Apfel@765291`

Methoden können überschrieben werden

Was heißt „überschreiben“?

```
class Obst {
  int gewicht;
  ...
  void essen() {
    gewicht = 0;
  }
}
```

```
class Apfel extends Obst {


  void essen() {
    gewicht = 10;
  }
  ...
}
```

```
public static void main ( String[] arguments ) {
  Apfel boskop = new Apfel();
  boskop.gewicht = 200;
  boskop.essen();
  System.out.println(boskop.gewicht);
}
```

Ausgabe: 10

String toString()

```
class Obst {
  String farbe;
  ...
  public String toString() {
    return "Farbe: " + farbe;
  }
}
```

```
public static void main ( String[] arguments ) {
  Apfel boskop = new Apfel();
  boskop.farbe = "rot";

  System.out.println( boskop );
}
```

Ausgabe:

```
Farbe: rot
```

Was vergleicht "=="?

```
Apfel boskop1 = new Apfel();
boskop1.farbe = "rot";
boskop1.gewicht = 200;

Apfel boskop2 = new Apfel();
boskop2.farbe = "rot";
boskop2.gewicht = 200;

Apfel boskop3 = boskop1;

System.out.println( boskop1 == boskop2 );
System.out.println( boskop1 == boskop3 );
```

Ausgabe:

```
false
true
```

Ein Beispiel für "equals()"

```
class Apfel {
  ...
  public boolean equals ( Object o ) {
    Apfel parameter = (Apfel)o;
    boolean gleicheFarbe =
                    this.farbe.equals(paramter.farbe);
    boolean gleichesGewicht =
                    (this.gewicht == paramter.gewicht);
    return gleicheFarbe && gleichesGewicht;
  }
}
```

"equals()" ist nicht immer "=="

```
class Apfel {
  ...
  public boolean equals ( Object o ) {
    ...
```

```
  Apfel boskop1 = new Apfel();
  boskop1.farbe = "rot";
  boskop1.gewicht = 200;
  Apfel boskop2 = new Apfel();
  boskop2.farbe = "rot";
  boskop2.gewicht = 200;

  System.out.println( boskop1 == boskop2 );
  System.out.println( boskop1.equals( boskop2 ) );
```

Ausgabe:

```
false
true
```

Gleichheit

Ordnung


STEPHEN BALDWIN
GABRIEL BYRNE
BENICIO DEL TORO
CHAZZ PALMINTERI
KEVIN POLLAK
PETE POSTLETHWAITE
KEVIN SPACEY
The Usual Suspects

Interface Comparable

```
public interface Comparable {
    public int compareTo(Object arg);
}
```

- **Ergebnis von compareTo():**

  kleiner 0 ⇔ this kleiner als arg

  größer 0 ⇔ this größer als arg

  gleich 0 ⇔ this gleich arg

```
class Apfel implements Comparable {
  ...
  public int compareTo ( Object o ) {
    Apfel parameter = (Apfel)o;
    return (this.gewicht - parameter.gewicht);
  }
}
```

Interface Comparable

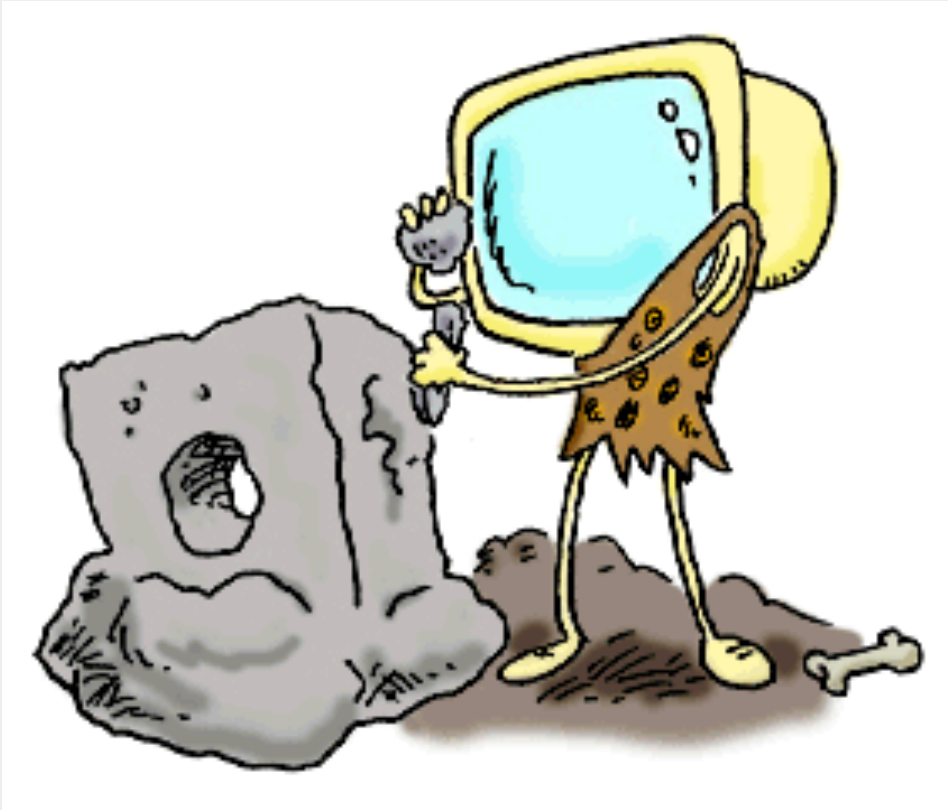
```
public int compareTo ( Object o ) {
  Apfel parameter = (Apfel)o;
  return (this.gewicht - parameter.gewicht);
}
```

```
Apfel boskop1 = new Apfel();
boskop1.gewicht = 200;
Apfel boskop2 = new Apfel();
boskop2.gewicht = 190;

System.out.println( boskop1.compareTo( boskop2 ) );
System.out.println( boskop2.compareTo( boskop1 ) );
```

Ausgabe:

```
10
-10
```

So it's true? We don't have to build it ourselves?



Java API

- **Java bietet unglaublich viele Klassen an:**
  - Mathematische Funktionen
  - Ein- / Ausgabe
  - Standard-Datenstrukturen
  - ...
- **Wie die richtige finden?**
  - Google
  - Java Doc (http://java.sun.com/j2se/1.4.2/docs/api/)

Die liebe Mathematik

PI

public static final double PI

The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

```
System.out.println ( Math.PI );
```

Ausgabe:

```
3.141592653589793
```

Zufallszahlen

random

public static double random()

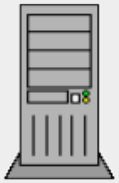
Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range. ...

Returns:

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

```
double zufallszahl1 =  Math.random();
double zufallszahl2 =  Math.random() * 100.0;
int wurf = (int)( Math.random() * 6.0 + 1.0 );
```

Von Strömen

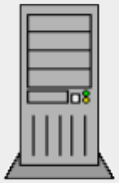


Web server

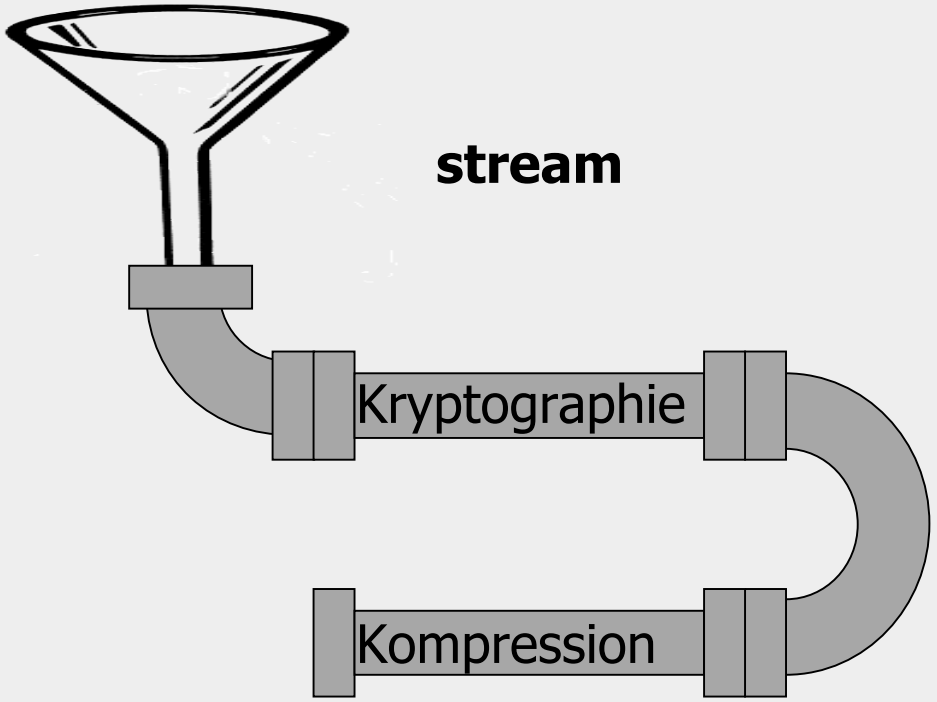
stream

13
111
108
108
101
72

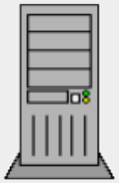
Erweiterbarkeit von Strömen



Web server



Ströme und Text



Web server

stream

Reader +

Buffered Reader

Hello

Zeilenweise lesen von der Tastatur

```
import java.io.*
public static void main ( String [] args ) {

    InputStreamReader reader =
                        new InputStreamReader(System.in);

    BufferedReader lineReader =
                        new BufferedReader(reader);

    String line = lineReader.readLine();

    System.out.println(line);
}
```

Exkurs: Wenn mal was schief geht

1. Lösung: Rückgabewert

```
// Näherung der Wurzelfunktion für positive Zahlen
// für negative Zahlen wird -1 zurückgegeben
static int wurzel ( int radikant ) {
    if (radikant < 0) {
        return -1;
    }
    int zahl = 1;
    while (zahl*zahl <= radikant) {
        zahl = zahl + 1;
    }
    return zahl-1;
}
```

Exkurs: Wenn mal was schief geht

Funktioniert nicht immer!

```
static int teilen ( int divident, int divisor ) {
    if (divisor == 0) {
        // Fehler!! Durch 0 kann nicht geteilt werden!
    } else {
        return divident / divisor;
    }
}
```

Javas Konstrukt: Exceptions

- **Beispiel:**

  readLine() des **BufferedReader** kann eine **IOException** werfen

```
static String getInput(String prompt) {
    System.out.print(prompt);
    BufferedReader lineReader = new BufferedReader(
        new InputStreamReader(System.in) );
    try {
        return lineReader.readLine();
    } catch (IOException e) {
        // Fehler "behandeln"
    }
    return null;
}
```

„Da kann ich nichts tun“

```
static String getInput(String prompt)
                                throws IOException {

    System.out.print(prompt);

    BufferedReader lineReader = new BufferedReader(
        new InputStreamReader(
        System.in) );

    return lineReader.readLine();
}
```

try-catch in Aktion

```
static FileInputStream openFile() throws IOException {
   FileInputStream result = null;
   String fileName = getInput("please enter filename");
   while (result == null) {
      try {
         result = new FileInputStream(fileName);
      } catch (FileNotFoundException e)  {
         fileName = getInput(
              “file not found - please try again");
      }
   }
   return result;
}
```

Jetzt viel Spaß bei der Übung!