Java Basics - Wiederholung

DHBW Mannheim - WWI17SCB - May 2018

1. Basics

Basics - Lernziele

* Bedingungen
* Texteingabe und Ausgabe
* Schleifen
* Arrays

Bedingungen - if

```java
int a = 5;
int b = 3;

if (a > b) {
  System.out.println("A is larger than B");
} else if (a < b) {
  System.out.println("B is larger than A");
} else {
  System.out.println("A and B are equal!");
}
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html

Bedingungen - switch/case

```
1 int a = 5;
2 int b = 3;
3 int diff = a - b;
4
5 switch (diff) {
6   case 2:
7     System.out.println("A is larger than B");
8     break;
9   case -2:
10    System.out.println("This will never happen");
11    break;
12  default:
13    System.out.println("Something went REALLY wrong.");
14 }
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html

Texteingabe mit Scanner

```
1 import java.util.Scanner;
2
3 Scanner input = new Scanner(System.in);
```

Texteingabe mit Scanner - wichtige Methoden

Scanner.next()

* Liest das nächste Token von der Eingabe und gibt es zurück

* Datentyp: String

```
1 Scanner input = new Scanner(System.in);
2
3 String s = input.next();
```

Texteingabe mit Scanner - wichtige Methoden

Scanner.nextInt()

* Liefert die nächste Eingabe als **int**

* Wirft eine InputMismatchException, wenn der Datentyp nicht kompatibel ist

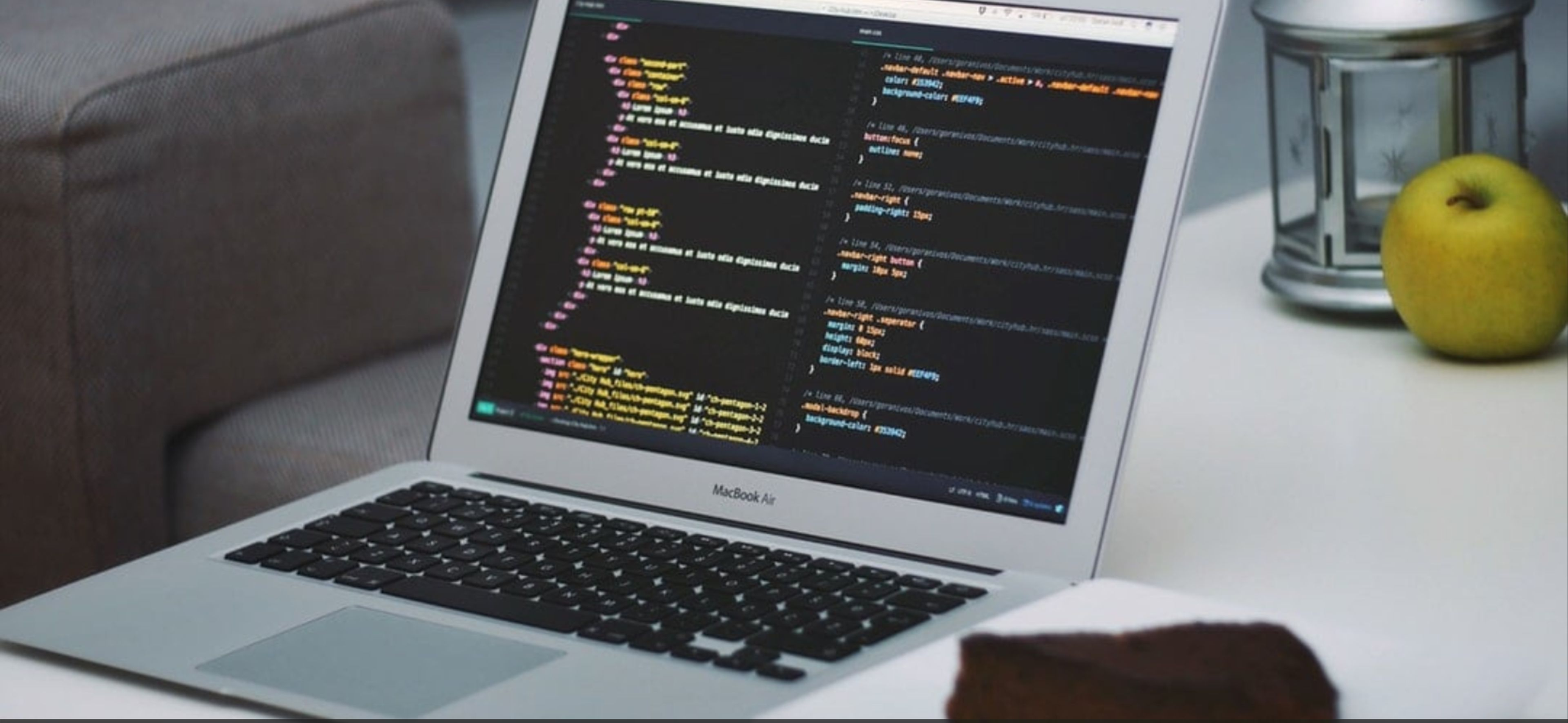
* Analog zu: **Integer.parseInt(input.next())**

* Existiert für alle skalaren Datentypen, also auch nextFloat, nextLong, nextChar, ...

Texteingabe mit Scanner - wichtige Methoden

Scanner.hasNextInt()

* Prüft, ob das nächste Token zum Datentyp int (float, double, char...) kompatibel ist
* Liefert true oder false zurück



Coding Time!

Bedingungen - Codingaufgabe

15m

Schreibt ein Programm, das:

* zwei Zahlen von der Console einliest
* die beiden Zahlen vergleicht
* ausgibt, welche der beiden Zahlen größer ist.
* Wenn die beiden Zahlen gleich sind, soll das ebenfalls ausgegeben werden

Bedingungen - Codingaufgabe - Musterlösung

```
1  Scanner input = new Scanner(System.in);
2
3  System.out.println("Zahl A:");
4  int a = input.nextInt();
5
6  System.out.println("Zahl B:");
7  int b = input.nextInt();
8
9  if (a > b) {
10   System.out.println("A größer als B");
11 } else if (a < b) {
12   System.out.println("B größer als A");
13 } else {
14   System.out.println("A und B sind gleich!");
15 }
```

Schleifen - while

```
1 int a = 0;
2
3 while (a < 10) {
4   a++;
5   System.out.println(a);
6 }
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html

Schleifen - do-while

```
1 int a = 0;
2
3 do {
4   a++;
5   System.out.println(a);
6 } while (a < 10);
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html

Schleifen - while vs do-while

```java
int a = 10;

while (a < 10) {
  a++;
  System.out.println(a);
}
```

```
Process finished with exit code 0
```

```java
int a = 10;

do {
  a++;
  System.out.println(a);
} while (a < 10);
```

```
11
Process finished with exit code 0
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html

Schleifen - for

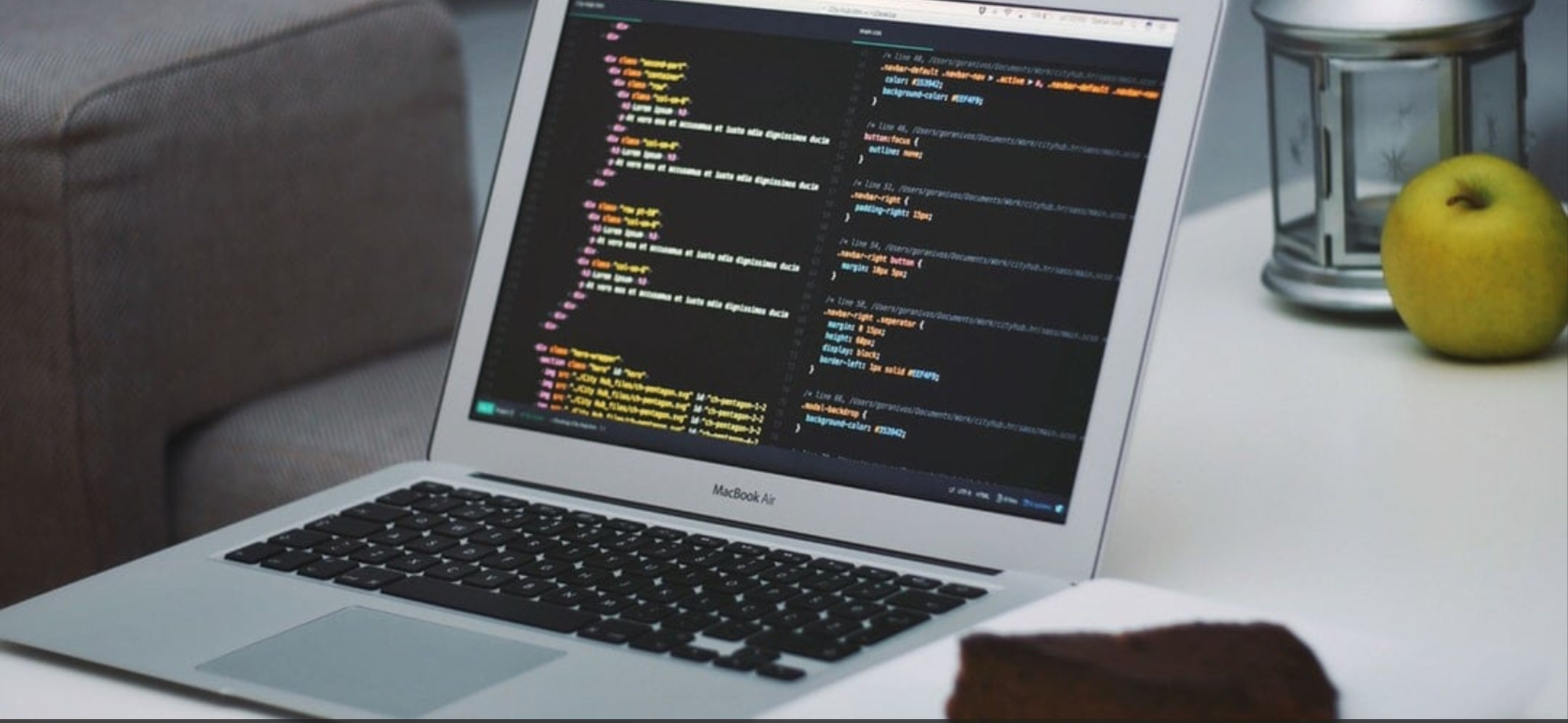
```
1 for (int a = 0; a < 10; a++) {
2   System.out.println(a);
3 }
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html

Schleifen - break

```
1 int a = 0;
2
3 while (true) {
4   a++;
5   if (a >= 10) {
6     break;
7   }
8   System.out.println(a);
9 }
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/branch.html



Coding Time!

Codingaufgabe - Kombination

30m

* Schreibt ein Programm, das so lange von der Eingabe liest, bis eine Zahl eingegeben wurde, und diese dann wieder ausgibt.

* Hierzu benötigt ihr:
  * Eine Endlosschleife
  * Die Texteingabe und Textausgabe
  * Eine Bedingung
  * **Scanner.hasNextInt(), Scanner.nextInt()** und **Scanner.next()**

Bedingungen - Kombination - Musterlösung

```
1 public static void main(String[] args) {
2   System.out.println("Bitte geben Sie eine Zahl ein:");
3   int num = readIntFromConsole();
4   System.out.println("Ihre Eingabe: " + num);
5 }
```

Bedingungen - Kombination - Musterlösung

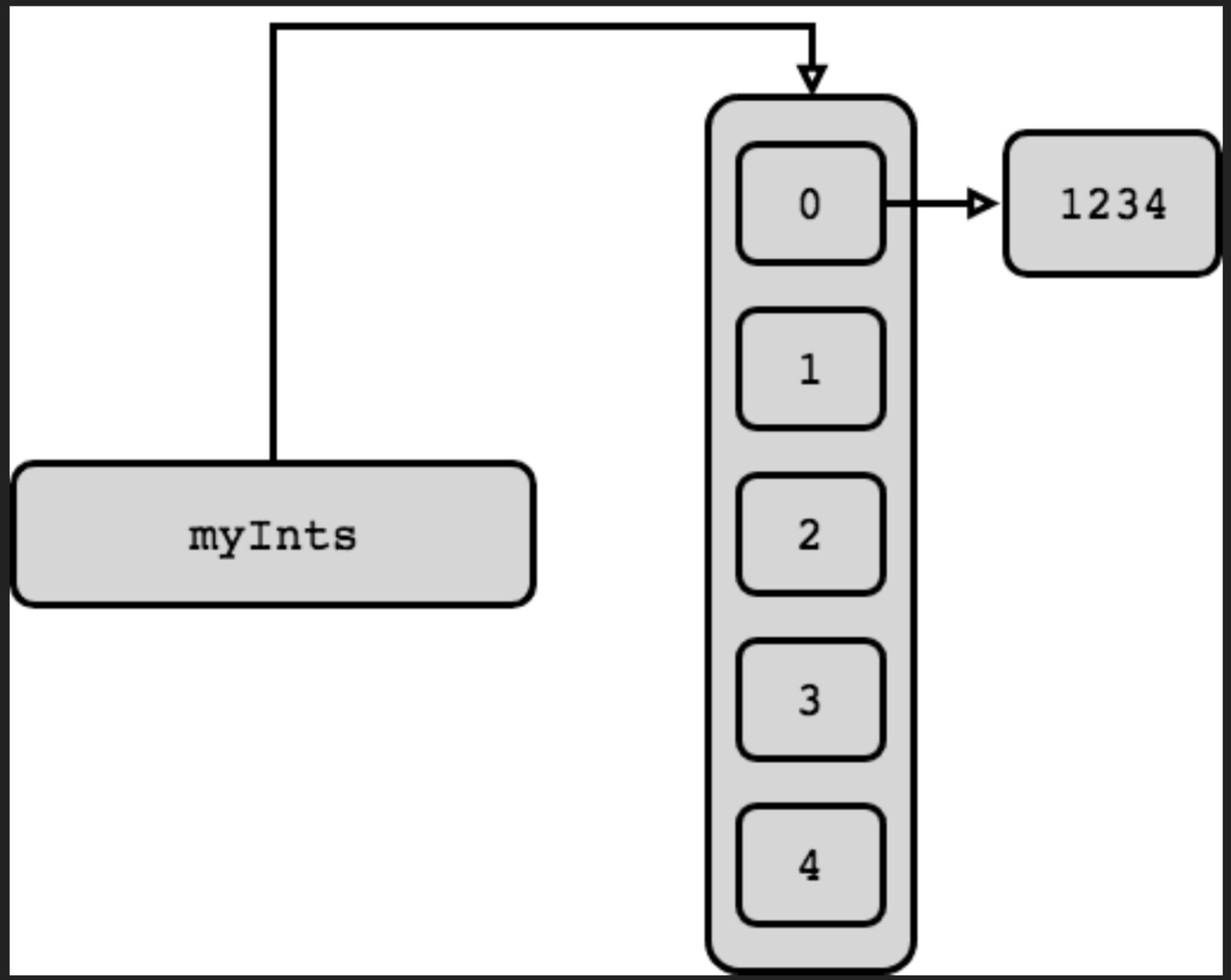
```
1 private static int readIntFromConsole() {
2   Scanner input = new Scanner(System.in);
3
4   while (true) {
5     if (input.hasNextInt()) {
6       return input.nextInt();
7     } else {
8       System.out.println("Bitte geben Sie eine Zahl ein.");
9       // Durch input.next() pausiert die Schleife bis zur nächsten Eingabe.
10      input.next();
11    }
12  }
13 }
```

Arrays

```java
// Declare an array of ints
int[] myInts;

// Actually create the array - before it was null!
myInts = new int[5];

// Or, in one step:
// int[] myInts = new int[5]

// Write something at position 0
myInts[0] = 1234;
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Arrays

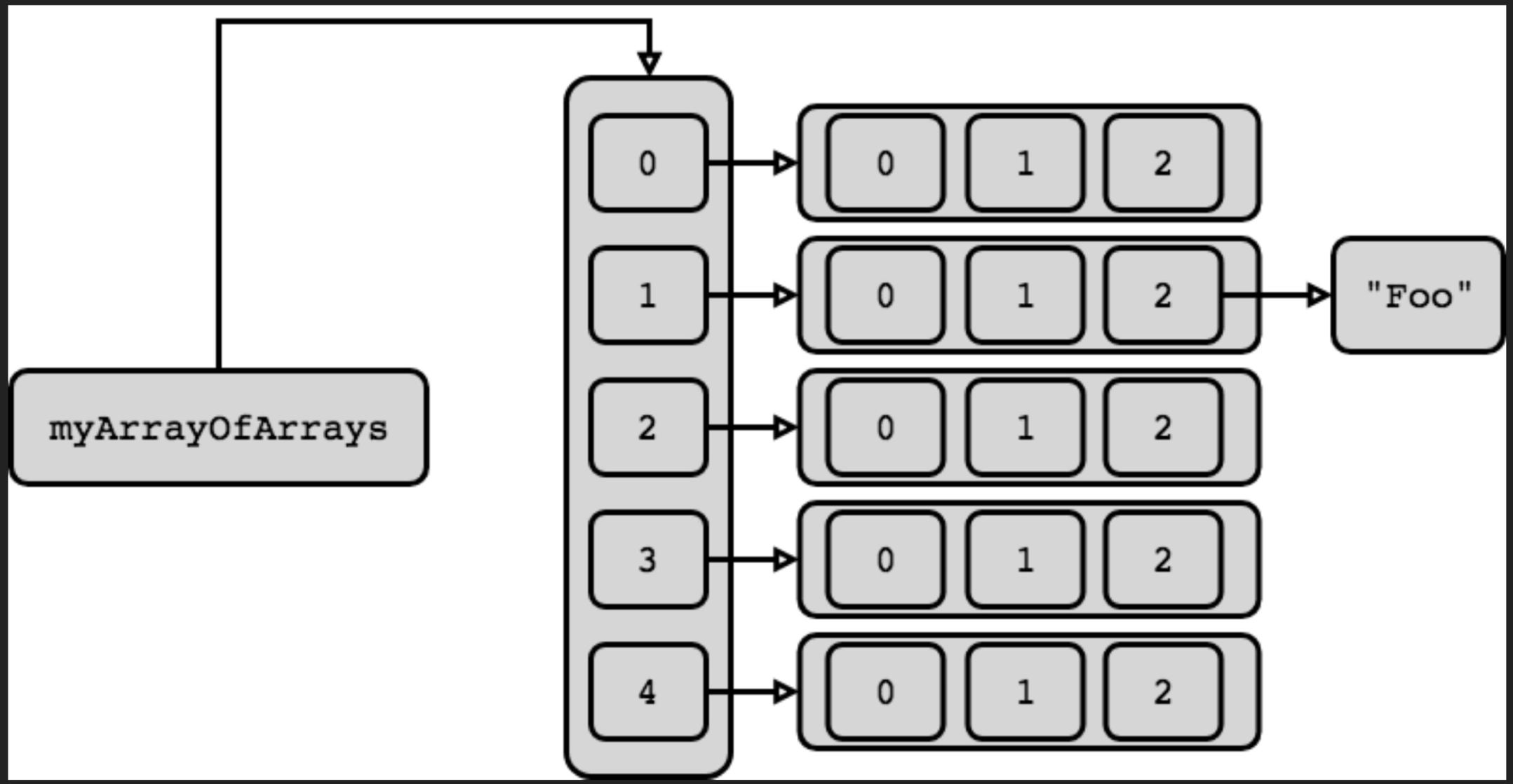


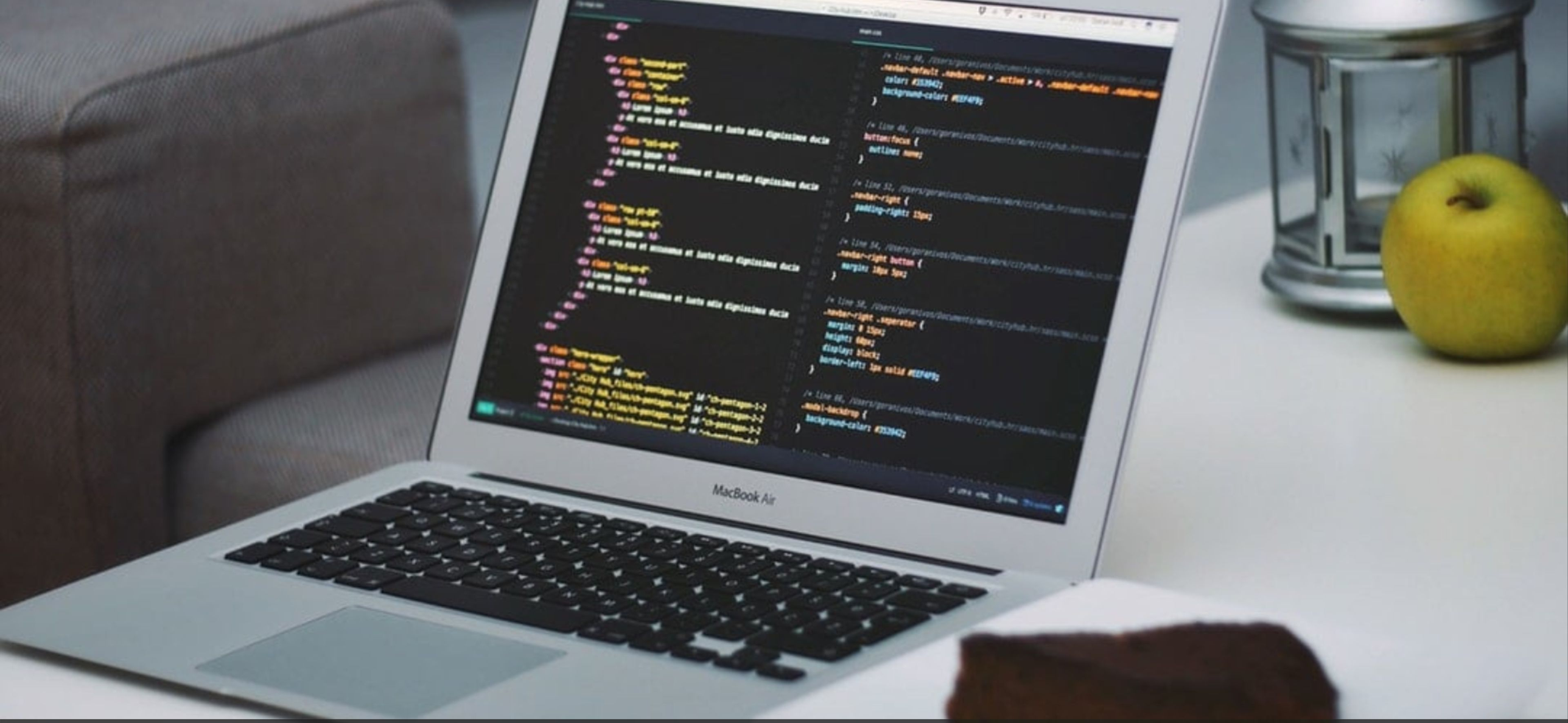
Arrays - Mehrdimensionale Arrays

```
1 // Declare an array of arrays of strings:
2 String[][] myArrayOfArrays = new String[5][3];
3
4 myArrayOfArrays[1][2] = "Foo";
```

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Arrays - Mehrdimensionale Arrays





Coding Time!

Codingaufgabe - Arrays

20m

* Schreibt ein Programm, das 10 Werte von der Eingabe liest.
* Ist der eingegebene Wert ein **int**, soll er in einem int[] Array gespeichert werden
* Ansonsten soll er in ein String[] Array gespeichert werden
* Nach 10 eingegebenen Werten sollen beide Arrays ausgegeben werden

2. Klassen und Objekte

Prinzipien der Objektorientierung

1. Kapselung
2. Abstraktion
3. Vererbung
4. Polymorphie

Zum Nachlesen:

http://codebetter.com/raymondlewallen/2005/07/19/4-major-principles-of-object-oriented-programming/

Kapselung

* Der innere Zustand eines Objekts ist nur diesem selbst bekannt
* Der innere Zustand kann von außen nicht direkt verändert werden
* Ein Objekt (bzw. eine Klasse) definiert eine Schnittstelle, über die es mit der Außenwelt interagieren kann

Kapselung

```
1  public class Auto {
2      // Leistung: intern in kW
3      private int leistung = 162;
4
5      // Umrechnungsfaktor
6      private static final double KW_IN_PS = 1.35962;
7
8      public int getLeistungInKW() {
9          return leistung;
10     }
11
12     public int getLeistungInPS() {
13         return Double.valueOf(leistung * KW_IN_PS).intValue();
14     }
15 }
```

Abstraktion

* Objekte **enthalten Wissen über die Domäne**
* Sie werden benutzt, um die Domäne semantisch (d.h. in der Sprache der Domäne selbst!) zu modellieren
* Es werden nur die Eigenschaften modelliert, die für die Lösung des Problems relevant sind
* Eine Klasse sollte einen möglichst kleinen und genau definierten Verantwortungsbereich haben --> **Single Responsibility Principle**

Abstraktion

“An abstraction denotes **the essential characteristics** of an object that distinguish it from all other kinds of object and thus provide **crisply defined conceptual boundaries**, relative to the perspective of the viewer.”

Klassen deklarieren

* Schlüsselwort **class;** eine (public) Klasse pro Datei

* Konvention: Zuerst alle Attribute, dann Methoden deklarieren

```
1  public class Buch() {
2    private String titel;
3    private String isbn;
4
5    public String getTitel() {
6      return titel;
7    }
8
9    public void setTitel(String titel) {
10     this.titel = titel;
11   }
12   ...
13 }
```

Klassen deklarieren

* Attribute können auch komplexere Felder sein
* Beispiele: Verweise auf andere Objekte, Arrays von Objekten

```
1 public class Bestellung() {
2   private Kunde besteller;
3   private Position[] bestellPositionen;
4 }
```

Objekte erzeugen

* Schlüsselwort **new**
* Es wird der Konstruktor der Klasse aufgerufen -> Name wie Klasse
* Es lassen sich beliebig viele Konstruktoren anlegen

```
1 public class Buch() {
2   private String titel;
3
4   public Buch(String titel){
5     this.titel = titel;
6   }
7 }
8
9 // Woanders...
10 Buch b = new Buch("The 7 habits of highly effective people");
```

Referenzvariablen

* Objekte sind - im Gegensatz zu skalaren Datentypen - nicht in Variablen gespeichert
* Eine Referenzvariable enthält lediglich eine Referenz (d.h. einen Zeiger) auf ein Objekt. Das Objekt selbst liegt irgendwo im Speicher
* Das wirkt sich insb. beim Methodenaufruf und bei der Zuweisung von Objekte an Variablen aus - es wird nur die Speicheradresse weitergegeben, sie zeigt auf das selbe Objekt!

Assignment "By Value"

```
1 int a = 5;
2 int b;
3 // Kopieren des WERTES von a nach b
4 b = a;
5 a = 0;
6 System.out.println(a);
7 System.out.println(b);
```

```
1 0
2 5
3
4 Process finished with exit code 0
```

Assignment "By Reference"

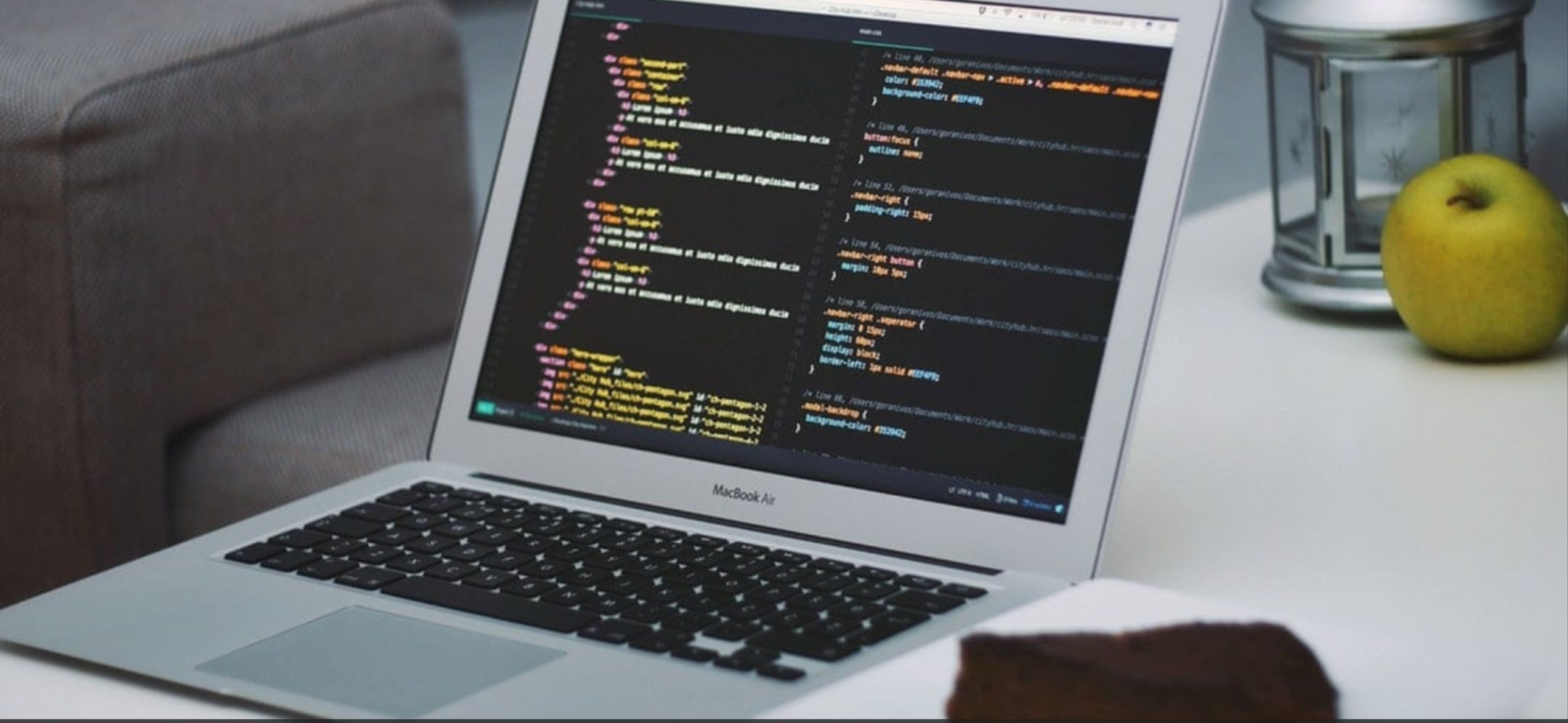
```java
Buch book1 = new Buch("Die Bibel");
Buch book2;
// Wir kopieren die Referenz auf das Buch-Objekt in die Variable book2
// Jetzt verweisen zwei Variablen auf das SELBE Objekt!
book2 = book1;
book1.setTitel("Das kommunistische Manifest");
System.out.println(book1.getTitel());
System.out.println(book2.getTitel());
```

```
Das kommunistische Manifest
Das kommunistische Manifest

Process finished with exit code 0
```

Statische Methoden und Attribute

* Klassen- vs Objektvariablen (Attribute)
* Klassenvariablen existieren nur 1x für alle Objekte einer Klasse
* Deklaration mittels **static**
* Ebenso für Methoden möglich

```
1 public class Ferrari() {
2   private static color = "red";
3 }
```



Coding Time!

Codingaufgabe - Klassen und Objekte

30m

Modelliert eine Uni mit folgenden Objekten:

* Universität (Name, n Vorlesungen, m Studenten)

* Vorlesung (Titel, Studenten)

* Student (Name, hört 1-n Vorlesungen)

Legt einige Objekte an und weist Studenten an Vorlesungen zu. Am Ende sollte die Uni eine print-Methode haben, die ca. das tut:

```
Uni 'DHBW Mannheim' hat insgesamt 3 Studenten und 4 Vorlesungen.

Paul Müller hört 'Programmierung 2', 'Rhetorik'
Lisa Engel hört 'Rhetorik', 'Bilanzierung'
Franz Faul hört 'Einführung in die Wirtschaftsinformatik'
```

Vererbung

* Objekte können zueinander in Beziehungen stehen
* Aussagen können z.B. sein: x benutzt y, x hat y, x ist ein y
* "ist ein"-Aussagen können mittels Vererbung modelliert werden
* Beispiel: Bibliothek verleiht Bücher, DVDs, CDs, Magazine
* Alle haben einen Titel, können ausgeliehen werden, haben eine bestimmte Anzahl Exemplare im Bestand, ...
* Aber: Bücher haben eine ISBN, Magazine eine Ausgabennummer, CDs eine Trackliste und DVDs einen Regisseur.

Vererbung

```
1 public class Medium {
2     private String titel;
3     private boolean verliehen;
4     private Date verliehenVon;
5     private Date verliehenBis;
6
7     // Getter und Setter (Accessoren) folgen...
8 }
```

Vererbung

```
1  public class Buch extends Medium {
2    private String isbn;
3
4    public String getISBN(){
5      return isbn;
6    }
7
8    public void setISBN(String isbn){
9      this.isbn = isbn;
10   }
11 }
```

Polymorphie

* "One Name, many Forms"
* Objekte können die "Form" (d.h. Klasse) aller kompatiblen Typen (d.h. aller Typen im eigenen Vererbungsbaum) annehmen

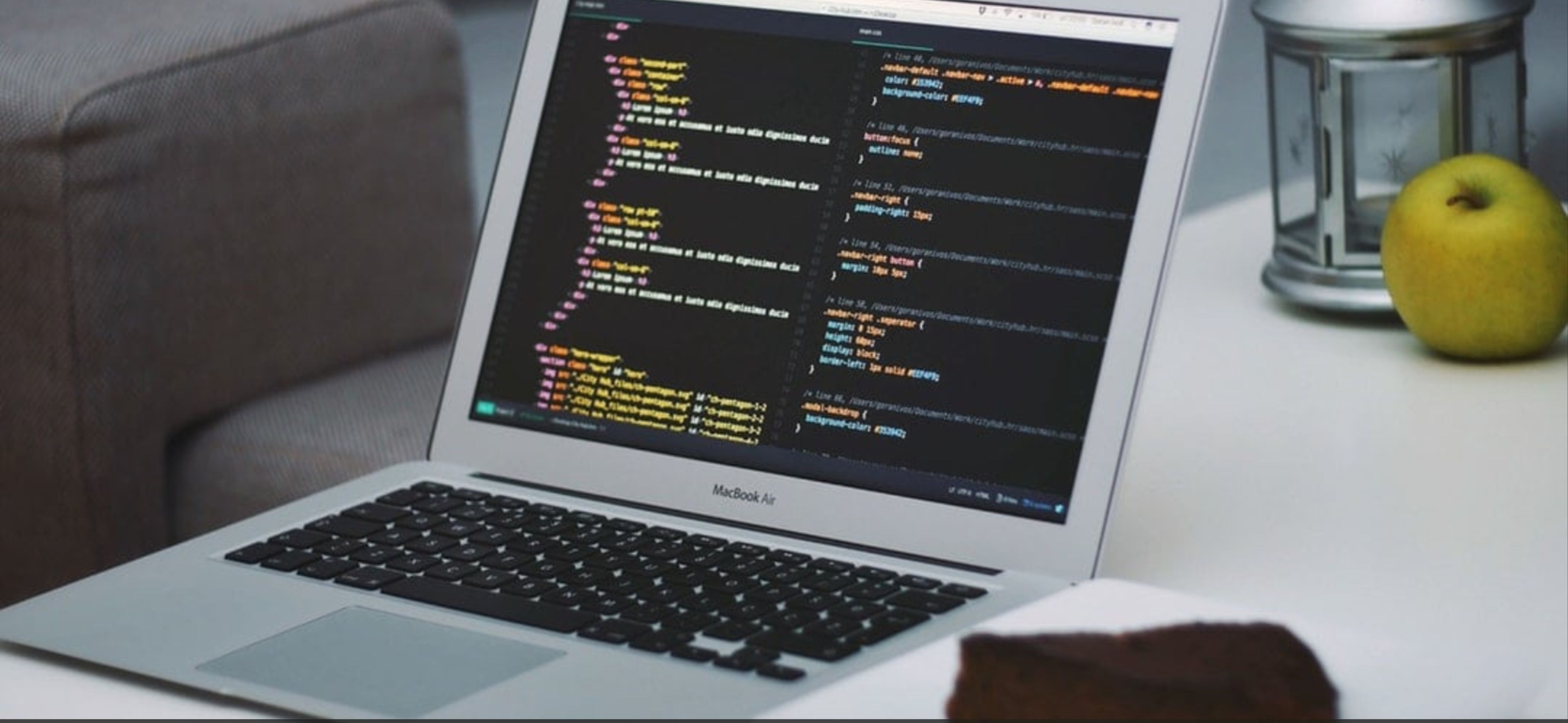
Polymorphie

```java
public class Medium {
    protected Date verliehenBis;
    protected String title;

    public Medium(String title) {
        this.title = title;
    }

    public void verleihen(Date verliehenBis) {
        this.verliehenBis = verliehenBis;
        System.out.println(title + " ist jetzt verliehen!");
    }
}
```

Polymorphie

```java
Buch buch1 = new Buch("Java für Anfänger");
DVD dvd1 = new DVD("Die Hard 3");

Medium[] ausleihe = new Medium[2];
ausleihe[0] = buch1;
ausleihe[1] = dvd1;

// Hier ist uns egal, was drinsteckt
for (int i = 0; i < ausleihe.length; i++) {
  ausleihe[i].verleihenBis("30.06.2018");
}
```

```
Java für Anfänger ist jetzt verliehen!
Die Hard 3 ist jetzt verliehen!

Process finished with exit code 0
```



Coding Time!

Codingaufgabe - Vererbung

30m

* Erweitert euer Uni-Projekt um Dozenten, die jeweils mindestens einer Vorlesung zugeordnet sind
* Studenten und Dozenten erben von der gemeinsamen abstrakten Klasse Person, die Vor- und Nachnamen hat
* Ausgabe (Beispiel):

```
Uni 'DHBW Mannheim' hat insgesamt 3 Studenten, 3 Dozenten und 4 Vorlesungen.

Paul Müller hört 'Programmierung 2' bei 'Dr. Fritz Meier', 'Rhetorik' bei 'Max Muster'
Lisa Engel hört 'Rhetorik' bei 'Max Muster', 'Bilanzierung' bei 'Dr. Fritz Meier'
Franz Faul hört 'Einführung in die Wirtschaftsinformatik' bei 'Werner Wernersen'
```