

Lektion 8

String und char-Array
Komplexe Datentypen
Modellierung
Erstellung von Objekten

String und char-Array

- Ein String(-Objekt) verwendet im Hintergrund ein Array von Einzelbuchstaben (char).
- Ein String stellt im Vergleich zum char-Array weitere Methoden zur Verfügung, die die Arbeit mit Strings erleichtern.
- Bisher kennen wir:
 - equals()
 - compareTo()
- Ähnlich wie bei den Arrays gibt es Methoden, um auf die Länge oder auf eine bestimmte Stelle des Strings zuzugreifen
 - length()
 - charAt()
- oder das hinter dem String liegende char-Array zurückgeben.
 - toCharArray()

Methode	Beschreibung
length()	gibt die Länge des Strings zurück. Bsp.: "Hallo".length() gibt 5 zurück
charAt(int i)	gibt den char an der Stelle i zurück. Bsp.: "Welt".charAt(0) gibt 'W' zurück, "Welt".charAt(1) gibt 'e' zurück.
toCharArray()	gibt das char-Array zurück, das dem String hinterlegt ist.

Großbuchstaben zählen mit charAt()

Wie zählt man die Großbuchstaben in einem String?

Man durchläuft den String buchstabenweise ...

... und überprüft für jeden Buchstaben, ob er
im Bereich von 'A' bis 'Z' liegt

Großbuchstaben zählen mit charAt()

```
public static int countUpperCaseCharacters(String sentence)
{
    int count = 0;
    for (int i = 0; i < sentence.length(); i++)
    {
        char c = sentence.charAt(i);
        if (c >= 'A' && c <= 'Z')
        {
            count++;
        }
    }
    return count;
}
```

String buchstabenweise
durchlaufen

Großbuchstabe?

Zähler erhöhen

Anzahl Großbuchstaben
zurückgeben

Alternative mit toCharArray()

```
public static int countUpperCaseCharacters2(String sentence)
{
    char[] chars = sentence.toCharArray();
    int count = 0;
    for (int i = 0; i < chars.length; i++)
    {
        char c = chars[i];
        if (c >= 'A' && c <= 'Z')
        {
            count++;
        }
    }
    return count;
}
```

String und char-Array

Ein String ist **unveränderlich (immutable)**, d. h. wenn der String verändert werden würde, wird stattdessen ein neuer String zurückgegeben.

```
String a = "Hallo";  
String b = "Welt";  
a + b;
```

- Weder a noch b werden verändert.
Stattdessen wird ein neuer String erstellt.

String: replace

- Mit `replace()` kann ein Einzelzeichen oder eine Zeichenfolge in einer Kopie des Strings getauscht werden. Die Kopie wird zurückgegeben.

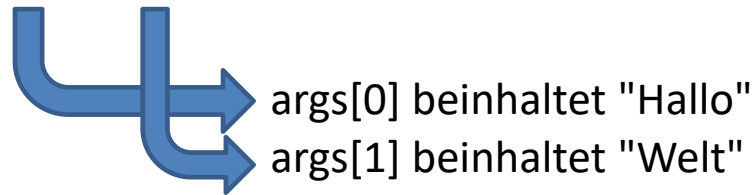
```
public static void main(String[] args)
{
    String a = "Hallo Welt";
    a = a.replace("l", "?");
    System.out.println(a);
}
```

Ein neuer String mit dem Inhalt
Ha??o We?t
wird zurückgegeben.

Argumente an die main-Methode übergeben

- Beim Programmstart kann man Argumente an das zu startende Programm übergeben:
- `java Mainparameter Hallo Welt`

```
public class Mainparameter
{
    public static void main(String[] args)
    {
        for (int i = 0; i < args.length; i++)
        {
            System.out.println(args[i]);
        }
    }
}
```



Wenn beim Programmstart Zahlen übergeben werden,
landen diese in dem String-Array args als Strings.

Es gibt Möglichkeiten den String in eine Zahl zu konvertieren.

Primitive Datentypen und Klassen

- Zu jedem primitiven Datentyp gibt es eine entsprechende Klasse:

primitiver Datentyp	dazugehörige Klasse
int	Integer
float	Float
double	Double
char	Character
boolean	Boolean
short	Short
byte	Byte

Die Klassen verfügen über bestimmte Eigenschaften und Methoden.

Zahlenbereiche:

Minimalwerte und Maximalwerte


Eigenschaft	Wert
Byte. <i>MAX_VALUE</i>	127
Byte. <i>MIN_VALUE</i>	-128
Short. <i>MAX_VALUE</i>	32767
Short. <i>MIN_VALUE</i>	-32768
Integer. <i>MAX_VALUE</i>	2147483647
Integer. <i>MIN_VALUE</i>	-2147483648
Long. <i>MAX_VALUE</i>	9223372036854775807
Long. <i>MIN_VALUE</i>	-9223372036854775808
Character. <i>MAX_VALUE</i>	65535
Character. <i>MIN_VALUE</i>	0
Float. <i>MAX_VALUE</i>	3.4028235E38
Float. <i>MIN_VALUE</i>	1.4E-45
Double. <i>MAX_VALUE</i>	1.7976931348623157E308
Double. <i>MIN_VALUE</i>	4.9E-324

negativen Wert erhält man
durch Voranstellen eines
Minus
(wenn überhaupt benötigt)

Umwandlung von Strings in andere Datentypen

- Jede dieser Klassen hat die statische Methode `valueOf(String)`, die einen übergebenen String in ein Objekt der entsprechenden Klasse umwandelt.

primitiver Datentyp	dazugehörige Klasse
int	Integer
float	Float
double	Double
char	Character
boolean	Boolean
short	Short
byte	Byte

`Integer.valueOf("25")` gibt eine 25 zurück.

String-Objekt Integer-Objekt

Autoboxing und Unboxing

- Ein primitiver Datentyp und Objekte von dessen Klasse waren vor Java 5 nicht zuweisungskompatibel.
- Seit Java 5 ist dies aber wie folgt möglich:
- Normale Zuweisung:
`Integer i = new Integer(12);`
- (Automatisches) Autoboxing:
`Integer i = 12; /*12 wird automatisch in ein Integer-Objekt "gepackt" und dann erst zugewiesen.*/`
- Automatisches Unboxing:
`System.out.println(10 * i); /*Aus dem Integer-Objekt i wird der int-Wert "entpackt" und mit 10 multipliziert*/`

Übersicht: Nützliche String Methoden

String Methoden

- Ein String-Objekt stellt einige wichtige Methoden bereit:

Methode	Beschreibung
length()	gibt die Länge des Strings zurück. Bsp.: "Hallo".length() gibt 5 zurück
charAt(int i)	gibt den char an der Stelle i zurück. Bsp.: "Welt".charAt(0) gibt 'W' zurück, "Welt".charAt(1) gibt 'e' zurück.
substring(int beginIndex, int endIndex)	gibt eine Kopie des Strings aus, die bei dem Buchstaben charAt(beginIndex) beginnt und bei inklusive dem Buchstaben charAt(endIndex - 1) endet. Bsp.: "HalloWelt".substring(1,6); gibt "alloW" zurück

"HalloWelt".substring(1,6); gibt "alloW" zurück

↑↑↑↑↑↑↑↑
012345678
└───┘
alloW

detailliertere Informationen im Java-Doc!!

String Methoden (ctd.)

Methode	Beschreibung
compareTo(String s)	vergleicht den String buchstabenweise und gibt eine negative Zahl zurück, wenn das String-Objekt vor dem übergebenen String im Alphabet kommt.
equalsIgnoreCase(String s)	vergleicht das String-Objekt mit dem übergebenen String ohne Groß- und Kleinbuchstaben zu berücksichtigen
indexOf(String s)	gibt den Index des ersten Vorkommens des Strings s zurück.
replace(old, new)	replace() ersetzt in einer Kopie des String-Objekts die Zeichenfolge <i>old</i> mit der Zeichenfolge <i>new</i>
toCharArray()	gibt das char-Array zurück, das dem String hinterlegt ist.
toLowerCase()	gibt eine Kopie des Strings in Kleinbuchstaben zurück
toUpperCase()	gibt eine Kopie des Strings in Großbuchstaben zurück
startsWith(String prefix)	gibt true zurück, wenn die ersten Buchstaben des Strings mit dem String <i>prefix</i> übereinstimmen.
contains(string)	gibt true zurück, wenn der String die übergebene Zeichenfolge beinhaltet.

String: toLowerCase(), toUpperCase(), equalsIgnoreCase()

- toLowerCase() gibt eine Kopie des Strings in Kleinbuchstaben zurück
- toUpperCase() gibt eine Kopie des Strings in Großbuchstaben zurück
- equalsIgnoreCase() vergleicht das String-Objekt mit dem übergebenen String ohne Groß- und Kleinbuchstaben zu berücksichtigen

```
String a = "Hallo";  
System.out.println(a);  
System.out.println(a.toLowerCase());  
System.out.println(a.toUpperCase());  
if (a.equalsIgnoreCase("hallo")) {  
    ...  
}
```

Hallo
hallo
HALLO

true

String: compareTo()

- `compareTo()` vergleicht den String buchstabenweise
- Beim ersten nichtgleichen Buchstaben wird angehalten und ausgegeben, um wie viele Stellen die beiden Buchstaben auseinanderliegen
- Sind alle Buchstaben bis zum Ende eines der beiden Strings gleich, wird die Längendifferenz ausgegeben.
- Ist die ausgegebene Zahl < 0 , kommt String a lexikographisch vor String b.

```
public static void main(String[] args) {  
    String a = "Awkward";  
    String b = "Hallo";  
    String c = "Hallowelt";  
    System.out.println(a.compareTo(b)); -7  
    System.out.println(b.compareTo(a)); 7  
    System.out.println(b.compareTo(c)); -4  
}
```

Was ergibt?

`System.out.println(c.compareTo(b));`

String ausschneiden mit indexOf + substring

- Ausschnitt der Wikipedia-Webseite:

```
<body id="www-wikipedia-org">
```

```
<div class="central-textlogo">
```

```
  
```

```
</body>
```

Wir haben obigen Webseitenausschnitt vorliegen.

Wie können wir mit einem Programm die URL des Bildes ausschneiden, um danach das Bild herunterzuladen?

```
String seite = "<body id=\"www-wikipedia-org\">"
+ "<div class=\"central-textlogo\">"
+ "<img src=\"http://upload.wikimedia.org/wikipedia/commons/thumb"
+ "/b/bb/Wikipedia_wordmark.svg/174px-Wikipedia_wordmark.svg.png\">"
+ "</div>"
+ "</body>";
```

Seitenausschnitt wird als String benötigt.
(Auch aus Datei oder aus dem Internet mgl.)

```
int startIndex = seite.indexOf("<img src=\"");
```

Suche nach
Beginn der URL

```
String seite = "<body id=\"www-wikipedia-org\">"
+ "<div class=\"central-textlogo\">"
+ "<img src=\"http://upload.wikimedia.org/wikipedia/commons/thumb"
+ "/b/bb/Wikipedia_wordmark.svg/174px-Wikipedia_wordmark.svg.png\">"
+ "</div>"
+ "</body>";
```

↑
startIndex

```
int startIndex = seite.indexOf("<img src=\"");
```

Suche nach
Beginn der URL

```
String seite = "<body id=\"www-wikipedia-org\">"
+ "<div class=\"central-textlogo\">"
+ "<img src=\"http://upload.wikimedia.org/wikipedia/commons/thumb"
+ "/b/bb/Wikipedia_wordmark.svg/174px-Wikipedia_wordmark.svg.png\">"
+ "</div>"
+ "</body>";
```

↑
startIndex

```
int startIndex = seite.indexOf("<img src=\"");
if (startIndex != -1)
{
    startIndex = startIndex + "<img src=\"".length();
}
}
```

Suche nach
Beginn der URL

```
String seite = "<body id=\"www-wikipedia-org\">"
+ "<div class=\"central-textlogo\">"
+ "<img src=\"http://upload.wikimedia.org/wikipedia/commons/thumb"
+ "/b/bb/Wikipedia_wordmark.svg/174px-Wikipedia_wordmark.svg.png\">"
+ "</div>"
+ "</body>";
```

↑

startIndex

↑

endIndex

```
int startIndex = seite.indexOf("<img src=\"");
if (startIndex != -1)
{
    startIndex = startIndex + "<img src=\"".length();
    int endIndex = seite.indexOf('"', startIndex);
}
}
```


startIndex

```
int startIndex = seite.indexOf("<img src=\"");
if (startIndex != -1)
{
    startIndex = startIndex + "<img src=\"".length();
    int endIndex = seite.indexOf('\"', startIndex);
    String url = seite.substring(startIndex, endIndex);
    System.out.println(url);
}
```

Download des Bildes?
In einem späteren Semester 😊

Komplexe Datentypen und Modellierung

Aus der Übung kennen wir
das Pizzabeispiel.

Eine Pizza bestand aus Durchmesser
und Preis.

Nehmen wir an, wir wollen eine
Speisekarte gestalten.

Dazu brauchen wir auf jeden Fall
zusätzlich den Namen.

Ø 26cm

1.	Pizza Tomaten, Käse ²	4,50 €
2.	Pizza Salami, Käse ^{2,4}	5,50 €
3.	Pizza Käse, Peperoniwurst ^{2,4,8}	5,50 €
4.	Pizza Käse, Schinken ^{2,4,7,8}	5,50 €
5.	Pizza Käse, Salami, Pilze ^{2,4}	6,50 €
6.	Pizza , Käse, Schinken, Pilze ^{2,4,7,8}	6,50 €
7.	Pizza Käse, Salami, Schinken, Pilze, Zwiebeln, Pepperoni ^{2,4,5,7,8}	7,50 €
8.	Pizza Hawaii, Käse, Schinken, Ananas ^{2,4,7,8}	6,50 €
9.	Pizza Käse, Salami, Schinken und Pilze ^{2,4,7,8}	6,50 €
10.	Pizza Käse, Broccoli, Knoblauch, Kapern ²	6,50 €
11.	Pizza à la Chef, Auberginen, Thunfisch, Oliven und Zwiebeln ²	7,50 €
12.	Pizza Käse, Spinat, Kapern und Ei ²	7,00 €
13.	Pizza Käse, Peperoniwurst, Paprika, Zwiebeln ^{2,4,8}	6,50 €
14.	Pizza Napoli , Mozzarella und Basilikum ^{2,5}	5,50 €
15.	Pizza Vegetaria , Spinat, Broccoli, Zucchini ²	7,00 €
16.	Pizza Spezial , Mozzarella, Ruccola, Frische Tomaten und frische Champignons ^{2,5}	7,50 €

Quelle: <http://www.pizzeria-buongiorno.de/speisekarte.html>

```
public class Speisekarte
{
    public static void main(String[] args)
    {
        String namePizza1;
        String namePizza2;
        String namePizza3;
        ...
        int durchmesserPizza1;
        int durchmesserPizza2;
        int durchmesserPizza3;
        ...
        float preisPizza1;
        float preisPizza2;
        float preisPizza3;
        ...
    }
}
```

Unsere bisherige
Vorgehensweise



Lieber Arrays!

```
public class Speisekarte
{
    public static void main(String[] args)
    {
        final int ANZAHL_PIZZEN = 15;
        String[] namePizza = new String[ANZAHL_PIZZEN];
        int[] durchmesserPizza = new int[ANZAHL_PIZZEN];
        float[] preisPizza = new float[ANZAHL_PIZZEN];
        ...
    }
}
```

Schon übersichtlicher!

Aber: Für jede Pizza müssen wir jetzt drei Variablen „verwalten“.



Kann man jede Pizza in **einer** eigenen Variable verwalten?

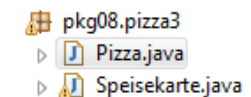
```
//Datei Pizza.java
public class Pizza
{
    String name;
    int durchmesser;
    float preis;
}
```

```
//Datei Speisekarte.java
public class Speisekarte
{
    public static void main(String[] args)
    {
        final int ANZAHL_PIZZEN = 15;
        Pizza[] pizzas = new Pizza[ANZAHL_PIZZEN];
        ...
    }
}
```

Wir haben einen neuen
komplexen Datentypen
definiert.

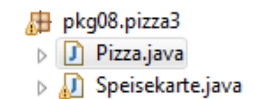
Den können wir ähnlich
wie andere Datentypen
verwenden.

neues Array von Pizzen



```
//Datei Pizza.java
public class Pizza
{
    String name;
    int durchmesser;
    float preis;
}
```

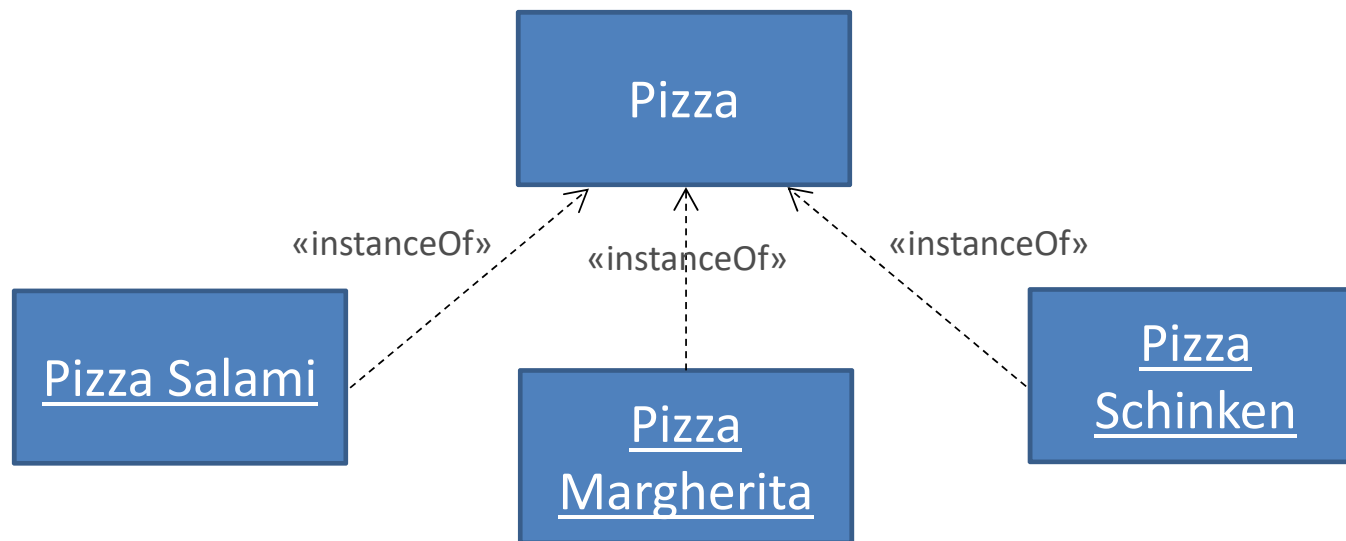
```
//Datei Speisekarte.java
public class Speisekarte
{
    public static void main(String[] args)
    {
        Pizza[] pizzas = new Pizza[15];
        ...
    }
}
```



Um gleichartige Objekte der realen Welt zu beschreiben, bietet sich eine Abstraktion an...

...eine sogenannte **Klasse**.

Klassen dienen als Baupläne/Schablonen, nach denen **Objekte/Instanzen** erstellt werden können.



Der Vorgang, Objekte aus der Realität
(und deren Beziehungen untereinander)
darzustellen, heißt **Modellierung**.

Das Modell betrachtet i.d.R. nur einen
wichtigen Ausschnitt der Realität.

➤ unwichtige Details werden weggelassen

- unwichtige Details werden weggelassen

Die Pizza soll auf einer Speisekarte
dargestellt werden:

Daher interessiert uns:
Name, Größe, Preise

Daher interessiert uns **nicht**:

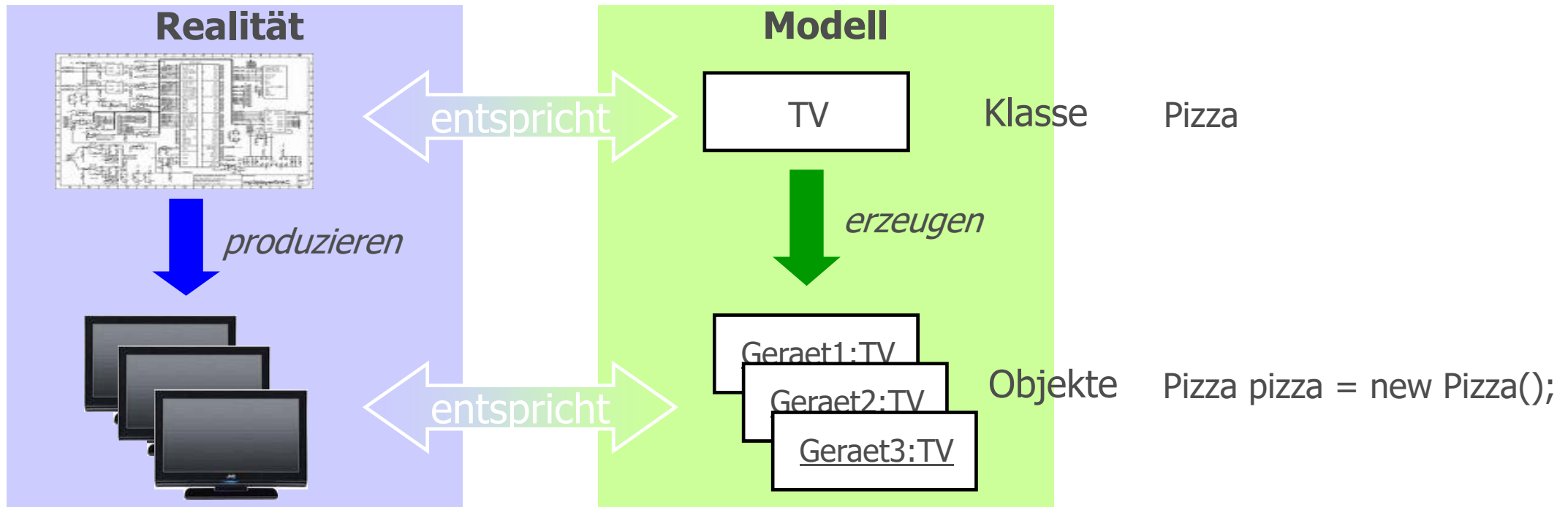
- wie stark ist die Pizza gebacken
- liegt die Salami auf oder unter dem Käse
- ist die Pizza aufgegessen oder nicht
- ...

```
public class Pizza
{
    String name;
    int durchmesser;
    float preis;
}
```

In dieser Form dient uns
die Klasse Pizza als
komplexer Datentyp zur
Haltung zusammengehöriger
Daten.

name, durchmesser und preis werden als **Attribute** oder **Felder (Fields)**
bezeichnet und beschreiben die **Eigenschaften** bzw. den **Zustand** des
Objekts.

Ein Objekt wird nach dem „Bauplan“
einer Klasse gebaut.



Wie modellieren wir eine Person?

```
public class Person
{
    String vorname;
    String nachname;
    String strasse;
    String hausnummer;
    String plz;
    String wohnort;
    int koerpergroesse;
    float gewicht;
    ...
}
```

Wir erstellen eine
eigene Klasse.

Welche Eigenschaften
sollen wir modellieren?

Wie modellieren wir eine Person?

```
public class Person
{
    String vorname;
    String nachname;
    String strasse;
    String hausnummer;
    String plz;
    String wohnort;
    int koerpergroesse;
    float gewicht;
    ...
}
```

Die Attribute hängen
vom Problem ab, das
wir lösen wollen!

Die Attribute hängen
vom Problem ab, das
wir lösen wollen!



Online Shop:
Person benötigt ggf. Bankverbindung, E-Mail-
Adresse, etc.

Die Attribute hängen
vom Problem ab, das
wir lösen wollen!



Login für eine Webseite:
Person benötigt nur einen Username und ein
Passwort, ggf. den Namen

Die Attribute hängen
vom Problem ab, das
wir lösen wollen!



Person beim Einwohnermeldeamt (z. B. für
den Personalausweis):
Körpergröße, Augenfarbe, etc.

Objekte erstellen

Klasse Person Person-Objekt erstellen

```
//Datei Person.java
public class Person
{
    String vorname;
    String nachname;
    String username;
}
```

```
//Datei Hauptprogramm.java
public class Hauptprogramm {
    public static void main (String[] args) {
        //neues Objekt/Instanz einer Person anlegen
        Person joe = new Person();
        joe.vorname = "Joe";
        joe.nachname = "Cool";
        joe.username = "jcool";
    }
}
```

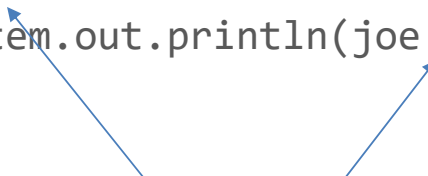
über **new** wird entsprechend dem Bauplan (d. h. der Klasse) Speicher zur Verfügung gestellt, d. h. Speicher für

- die **Referenz** *joe* auf das Personenobjekt
- das Attribut *vorname*
- das Attribut *nachname*
- das Attribut *username*
- Standardattribute, die jedes Objekt in Java hat (*hashCode*, etc.)
- ...

Klasse Person Person-Objekt erstellen

```
//Datei Person.java
public class Person
{
    String vorname;
    String nachname;
    String username;
}
```

```
//Datei Hauptprogramm.java
public class Hauptprogramm {
    public static void main (String[] args) {
        //neues Objekt/Instanz einer Person anlegen
        Person joe = new Person();
        joe.vorname = "Joe";
        joe.nachname = "Cool";
        joe.username = "jcool";
        System.out.println(joe.vorname);
    }
}
```



Über den Punktoperator **.** kann auf die
Attribute
(und auch Methoden)
des Objekts zugegriffen werden.

Über den Punktoperator . kann auf die
Attribute
(und auch Methoden)
des Objekts zugegriffen werden.

Allgemein:
<Name der Referenz>.<Attributname>

Beispiel:
joe.vorname

new-Operator

Über **new** wird Speicher für das zu erstellende Objekt angefordert.

```
new Person();
```

Anschließend gibt der **new**-Operator eine **Referenz** auf den angelegten Speicherbereich zurück.

Die zurückgegebene Referenz wird dann
bspw. in **p** gespeichert:

```
Person p;  
p = new Person();
```


Referenztypen

Variablen (z.B. p) von Referenztypen (z.B. Person) können also Referenzen speichern.

Daher unterscheidet man primitive Datentypen von Referenztypen.

Deklaration
primitiver Datentypen

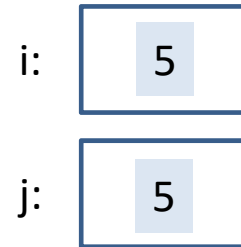
- `int i;`
- `double x;`

Deklaration von
Referenztypen

- `Scanner scanner;`
- `Person p;`
- `Pizza salami;`

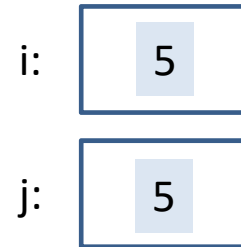
primitive Datentypen

```
int i, j;  
i = 5;  
j = i;
```



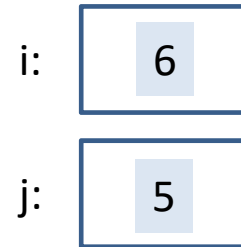
primitive Datentypen

```
int i, j;  
i = 5;  
j = i;  
i = 6;
```



primitive Datentypen

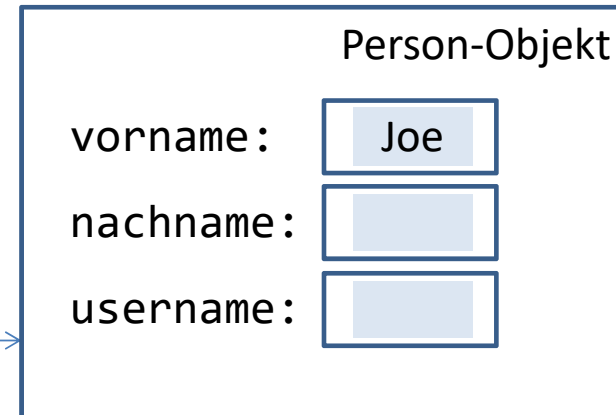
```
int i, j;  
i = 5;  
j = i;  
i = 6;
```



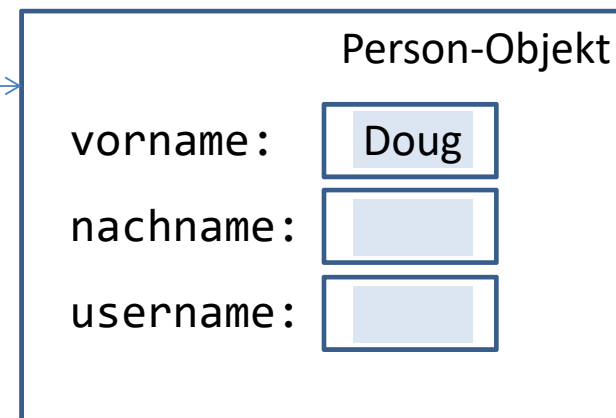
Referenztypen

```
Person p1 = new Person();  
Person p2 = new Person();  
p1.vorname = "Joe";  
p2.vorname = "Doug";
```

p1 →

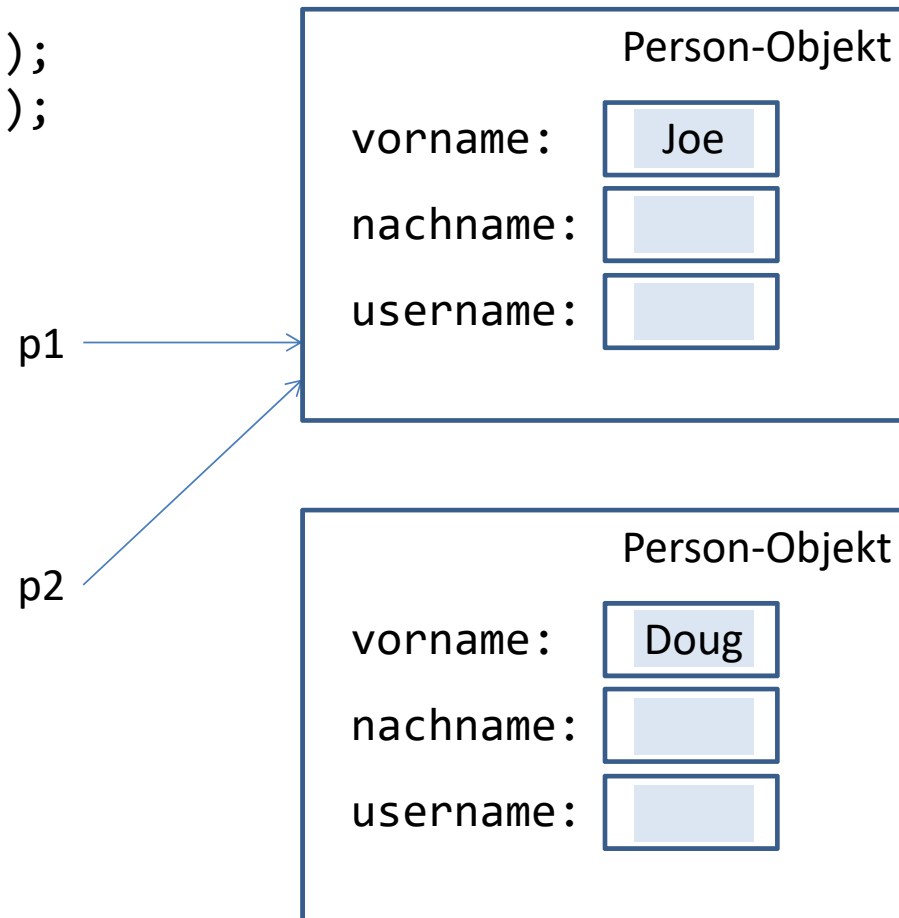


p2 →



Referenztypen

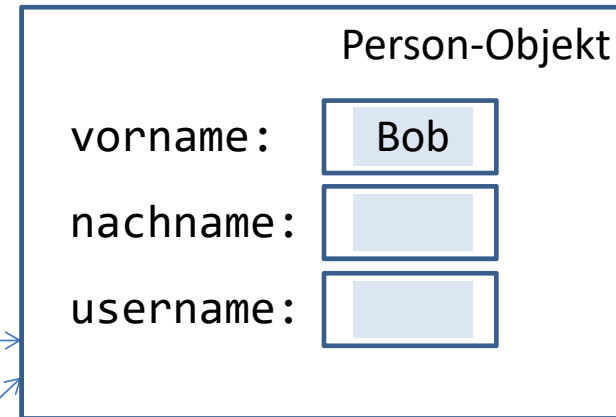
```
Person p1 = new Person();  
Person p2 = new Person();  
p1.vorname = "Joe";  
p2.vorname = "Doug";  
p2 = p1;
```



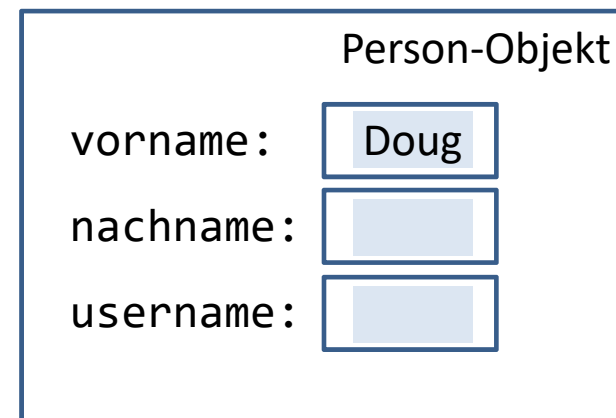
Referenztypen

```
Person p1 = new Person();  
Person p2 = new Person();  
p1.vorname = "Joe";  
p2.vorname = "Doug";  
p2 = p1;  
p1.vorname = "Bob";
```

p1 →

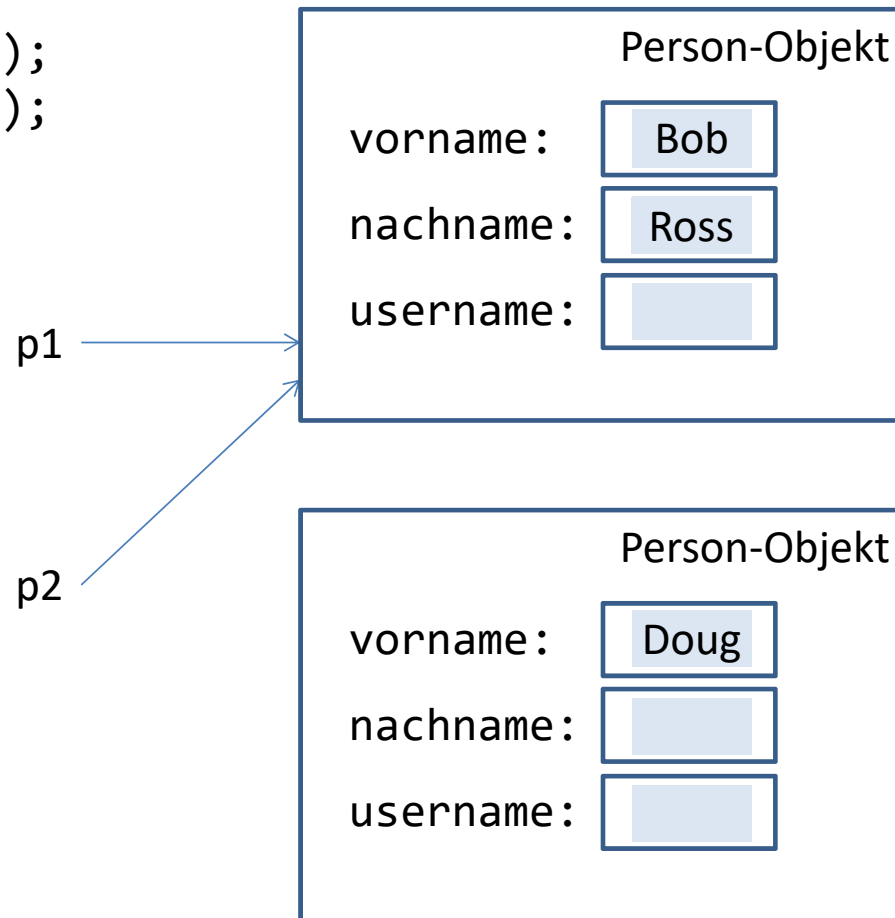


p2 →



Referenztypen

```
Person p1 = new Person();  
Person p2 = new Person();  
p1.vorname = "Joe";  
p2.vorname = "Doug";  
p2 = p1;  
p1.vorname = "Bob";  
p2.nachname = "Ross";
```

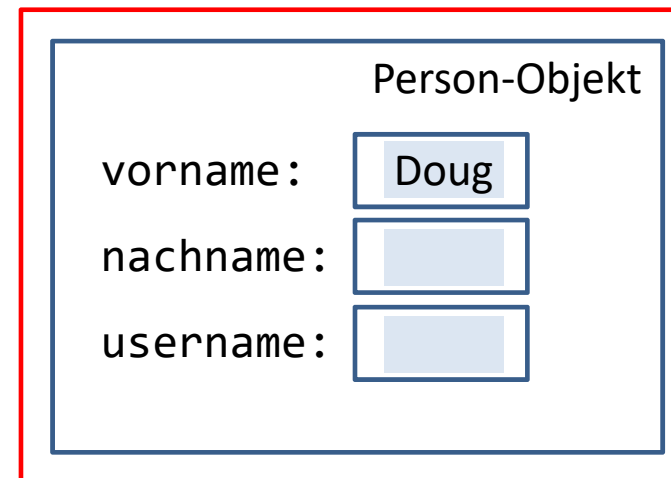
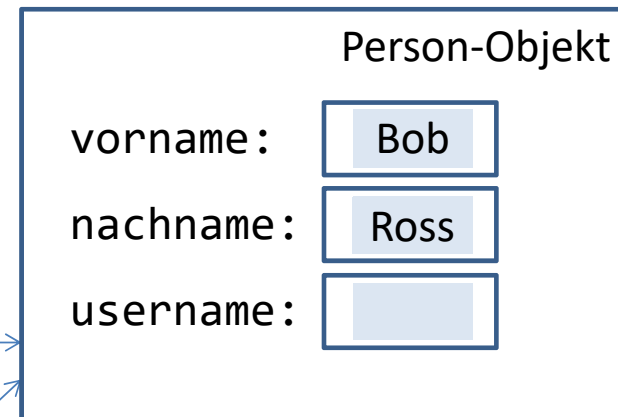


Referenztypen

```
Person p1 = new Person();  
Person p2 = new Person();  
p1.vorname = "Joe";  
p2.vorname = "Doug";  
p2 = p1;  
p1.vorname = "Bob";  
p2.nachname = "Ross";
```

p1

p2



Dieses Objekt ist nicht mehr
erreichbar!