

10. Klassen

10.1 Klassen und Objekte

10.2 Beispiele

10.3 Klassen und Arrays

Motivation



Wie würde man ein Datum speichern (z.B. 13. November 2018)?

3 Variablen

```
int day;  
String month;  
int year;
```

Unbequem, wenn man mehrere Exemplare davon braucht:

```
int day1;  
String month1;  
int year1;  
int day2;  
String month2;  
int year2;  
...
```

Idee: die 3 Variablen zu einem eigenen Datentyp zusammenfassen

Datentyp Klasse

Speicherung *verschiedenartiger* Werte unter einem gemeinsamen Namen
(Arrays dienen der Speicherung *gleichartiger* Elemente)

Deklaration

```
class Date {
    int day;
    String month;
    int year;
}
```

} Felder der Klasse *Date*

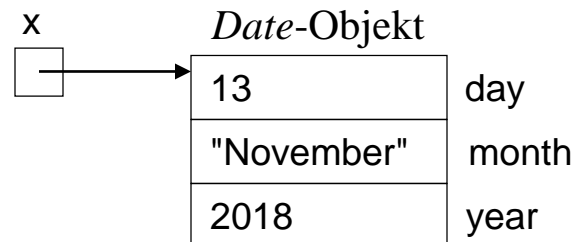
Verwendung als Typ

```
Date x, y;
```

Zugriff

```
x.day = 13;
x.month = "November";
x.year = 2018;
```

Felder sind wie Variablen
benutzbar





x ist eine Variable vom Typ *Date*
Sie enthält (genauer: zeigt auf) ein *Date*-Objekt

Objekte einer Klasse müssen vor ihrer ersten Benutzung erzeugt werden

```
Date x, y;
```

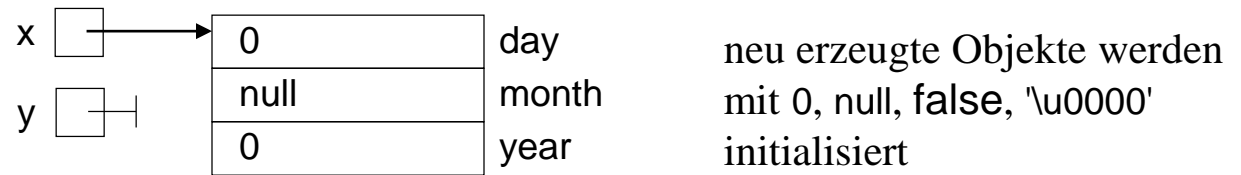
reserviert nur Speicher für die Zeigervariablen

x  y  haben anfangs den Wert *null*

Erzeugung

```
x = new Date();
```

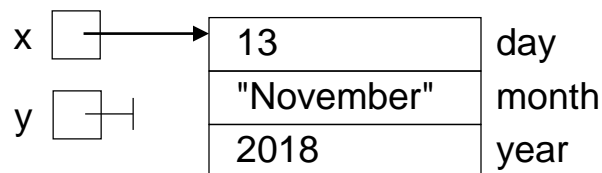
erzeugt ein *Date*-Objekt und weist seine Adresse *x* zu



Eine Klasse ist wie eine Schablone, von der beliebig viele Objekte erzeugt werden können.

Benutzung

```
x.day = 13;  
x.month = "November";  
x.year = 2018;
```



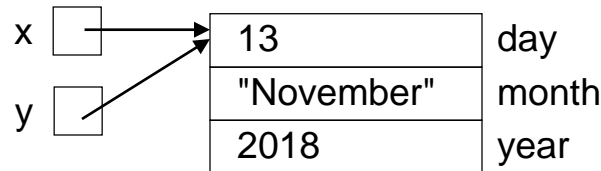
Freigabe von Objekten

durch den Garbage Collector

Zuweisungen

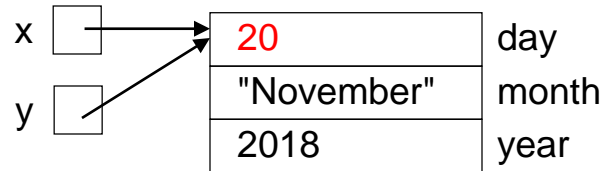


```
y = x;
```



Zeigerzuweisung!

```
y.day = 20;
```



ändert auch `x.day`!
(Alias-Effekt)

Zuweisungen sind erlaubt, wenn die Typen gleich sind

```
class Date {  
    int day;  
    String month;  
    int year;  
}
```

```
class Address {  
    int number;  
    String street;  
    int zipCode;  
}
```

```
Date d1, d2;  
Address a1, a2;
```

```
d1 = d2;           // ok, gleiche Typen
```

```
a1 = a2;           // ok, gleiche Typen
```

```
d1 = a2;           // verboten: verschiedene Typen trotz gleicher Struktur!
```

Vergleiche



Zeigervergleich

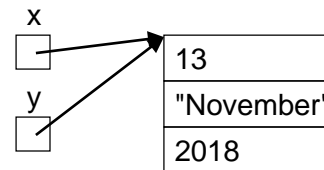
`x == y`
`x != y`

vergleicht nur Zeiger

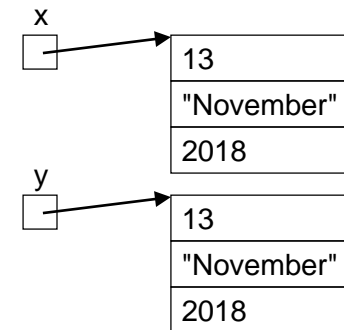
`x < y`
`x <= y`
`x > y`
`x >= y`

nicht erlaubt

`x == y` liefert *true*



`x == y` liefert *false*



Wertvergleich muss mittels Vergleichsmethode selbst implementiert werden

```
static boolean equalDate (Date x, Date y) {  
    return  
        x.day == y.day &&  
        x.month.equals(y.month) &&  
        x.year == y.year;  
}
```

```
if (equalDate(x, y)) ...
```

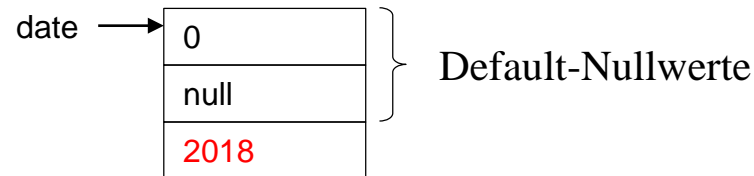
Initialisieren von Feldern



```
class Date {  
    int    day;  
    String month;  
    int    year = 2018;  
}
```

wie bei Variablen

```
Date date = new Date();
```



Initialisierte Felder dürfen natürlich nachträglich verändert werden

```
date.day = 13;  
date.month = "May";  
date.year = 2019;
```

... außer wenn sie *final* deklariert sind

```
class Date {  
    int    day;  
    String month;  
    final int year = 2018;  
}
```

behält Initialwert bei

Wo werden Klassen deklariert

In einer einzigen Datei (auf äußerster Ebene)

MainProgram.java

```
class C1 {
    ...
}
```

```
class C2 {
    ...
}
```

```
class MainProgram {
    public static void main (String[] arg) {
        ...
    }
}
```

In getrennten Dateien

C1.java

```
class C1 {
    ...
}
```

C2.java

```
class C2 {
    ...
}
```

MainProgram.java

```
class MainProgram {
    public static void main (String[] arg) {
        ...
    }
}
```

Übersetzung

```
javac MainProgram.java
```

```
javac MainProgram.java C1.java C2.java
```


10. Klassen

10.1 Klassen und Objekte

10.2 Beispiele

10.3 Klassen und Arrays

Beispiel: Verknüpfung von Objekten



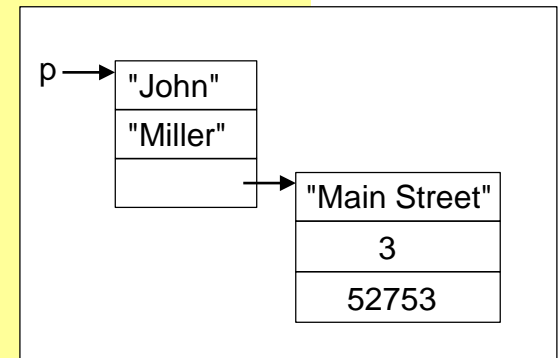
Personen mit Adressen

```
class Person {  
    String  firstName;  
    String  lastName;  
    Address addr;  
}
```

```
class Address {  
    String  street;  
    int     nr;  
    int     zip;  
}
```

Erzeugen einer Person mit Adresse

```
static Person createPerson (String fn, String ln, String street, int nr, int zip) {  
    Person p = new Person();  
    p.firstName = fn;  
    p.lastName = ln;  
    p.addr = new Address();  
    p.addr.street = street;  
    p.addr.nr = nr;  
    p.addr.zip = zip;  
    return p;  
}
```



```
Person p = createPerson("John", "Miller", "Main Street", 3, 52753);
```

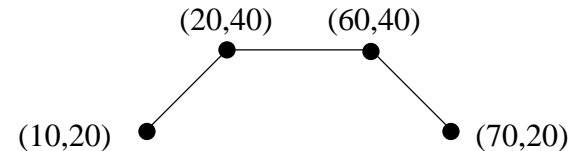
Mit Konstruktoren geht das wieder einfacher (s. später)

Beispiel: Polygone

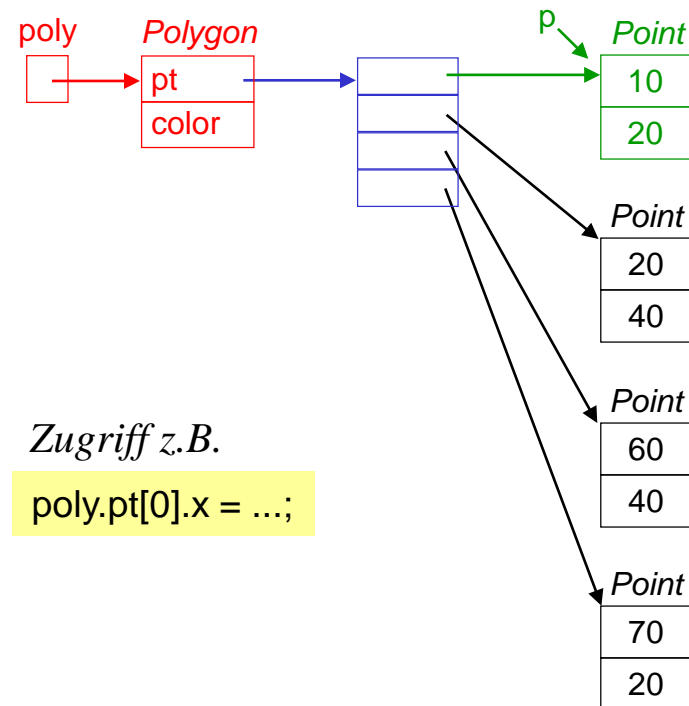


```
class Point {  
    int x, y;  
}
```

```
class Polygon {  
    Point[] pt;  
    int color;  
}
```



```
...  
Polygon poly = new Polygon();  
poly.pt = new Point[4];  
poly.color = RED;  
Point p = new Point(); p.x = 10; p.y = 20;  
poly.pt[0] = p;  
p = new Point(); p.x = 20; p.y = 40;  
poly.pt[1] = p;  
p = new Point(); p.x = 60; p.y = 40;  
poly.pt[2] = p;  
p = new Point(); p.x = 70; p.y = 20;  
poly.pt[3] = p;  
...
```



Geht mit Konstruktoren wesentlich eleganter (s. später)

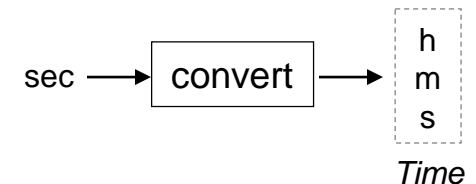
Methoden mit mehreren Rückgabewerten



Java-Funktionen haben nur 1 Rückgabewert

Will man mehrere Rückgabewerte, muss man sie zu einer Klasse zusammenfassen

Beispiel: Umrechnung von Sekunden auf Std, Min, Sek



```
class Time {  
    int h, m, s;  
}
```

```
class Program {  
    static Time convert (int sec) {  
        Time t = new Time();  
        t.h = sec / 3600; t.m = (sec % 3600) / 60; t.s = sec % 60;  
        return t;  
    }  
    public static void main (String[] arg) {  
        Time t = convert(10000);  
        Out.println(t.h + ":" + t.m + ":" + t.s); // ergibt 2:46:40  
    }  
}
```

10. Klassen

10.1 Klassen und Objekte

10.2 Beispiele

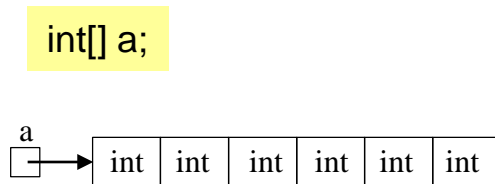
10.3 Klassen und Arrays

Klassen versus Arrays

Beides sind **strukturierte Datentypen**
d.h. sie bestehen aus mehreren Elementen

Arrays

Folge gleichartiger Elemente



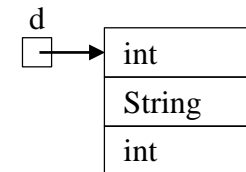
Elemente über Index angesprochen

`a[3] = ...;`

Klassen

Folge unterschiedlicher Elemente

```
class Date {
    int day;
    String month;
    int year;
}
```



`Date d;`

Elemente über Namen angesprochen

`d.month = ...;`

Kombination von Klassen mit Arrays

Beispiel: Telefonbuch

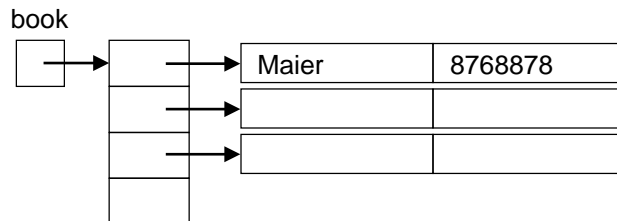
	name	phone
0	Maier	8768878
	Mayr	5432343
	Meier	6562332
99		

zweidimensionales Array
kann hier nicht verwendet werden

Array von Objekten

```
class Person {
    String name;
    int phone;
}

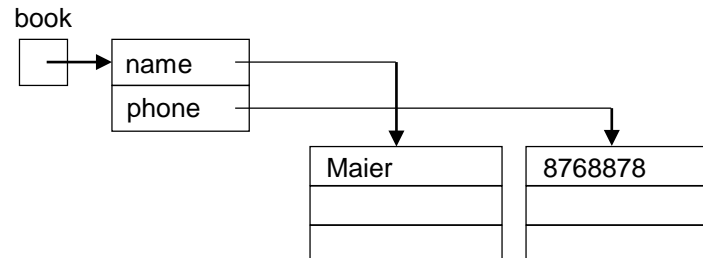
Person[] book = new Person[100];
book[0] = new Person(); ...
...
```



Objekt bestehend aus 2 Arrays

```
class PhoneBook {
    String[] name;
    int[] phone;
}

PhoneBook book = new PhoneBook();
book.name = new String[100];
book.phone = new int[100];
```



Diese Lösung ist aus logischer Sicht besser

Implementierung

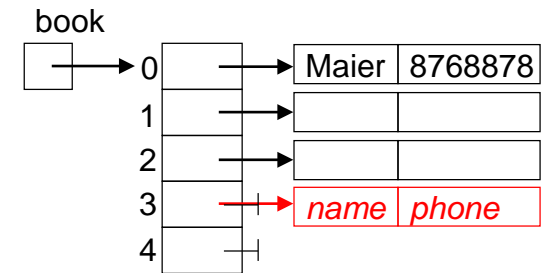


```
class Person {  
    String name;  
    int phone;  
}
```

```
class PhoneBookSample {  
    static Person[] book = new Person[1000];  
    static int nEntries = 0; // current number of entries in book
```

```
    static void enter (String name, int phone) {  
        if (nEntries >= book.length) {  
            Out.println("--- phone book full");  
        } else {  
            Person p = new Person();  
            p.name = name; p.phone = phone;  
            book[nEntries] = p;  
            nEntries++;  
        }  
    }  
}
```

```
    static int lookup (String name) {  
        int i = 0;  
        while (i < nEntries && !name.equals(book[i].name)) i++;  
        // i >= nEntries || name.equals(book[i].name)  
        if (i < nEntries) return book[i].phone; else return -1;  
    }  
}
```



nEntries = ~~3~~
4

Implementierung (Fortsetzung)



...

```
public static void main (String[] arg) {
    //----- read the phone book from a file
    In.open("phonebook.txt");
    String name = In.readWord();
    while (In.done()) {
        int phone = In.readInt();
        enter(name, phone);
        name = In.readWord();
    }
    In.close();

    //----- search in the phone book
    for (;;) {
        Out.print("Name: "); name = In.readWord();
        if (!In.done()) break;
        int phone = lookup(name);
        if (phone >= 0) {
            Out.println("phone number = " + phone);
        } else {
            Out.println(name + " unknown");
        }
    }
}
```

```
} // end PhoneBookSample
```