Einstieg in die Informatik mit Java

Klassen als Datenstrukturen

Gerd Bohlender

Institut für Angewandte und Numerische Mathematik

Gliederung

Gliederung

Klassen als Datenstruktur

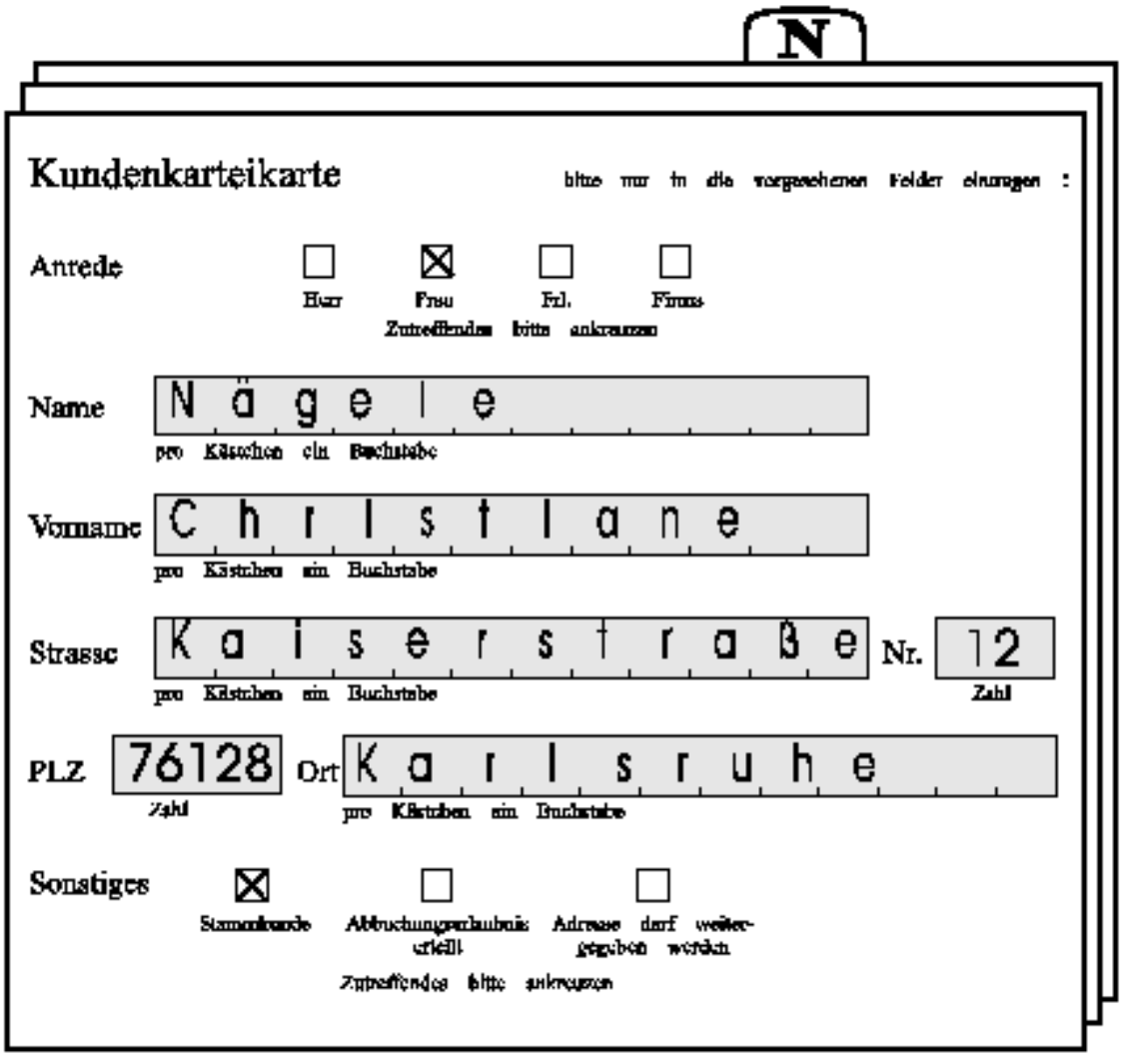
Im Gegensatz zu Feldern können auch verschiedenartige Daten zusammen abgespeichert werden.

Beispiel

Person: Name (String), Vorname (String), Geburtsdatum (3 ganze Zahlen).
Konto: Inhaber (String oder Person), Nummer (ganze Zahl), Saldo (Gleitkommazahl)

- Die Einträge in einer Klasse heißen *Attribute (Eigenschaften)*, *Daten-Komponenten* oder *Instanzvariablen*.
- Jedes Attribut kann einen beliebigen anderen Typ besitzen (*Komponententyp*).
- Variablen vom Typ einer Klasse heißen *Objekte* oder *Instanzen* der Klasse. Diese werden dynamisch erzeugt.

Beispiel: Karteikarte



Gliederung

Vereinbarung von Klassen

Syntax

class Klassenname {
    Komponententyp Bezeichnerliste ;
    ...
}

Achtung

Durch die Deklaration einer Klasse wird noch kein Speicher für die Attribute / Instanzvariablen reserviert; es existieren noch keine Instanzen!

Beispiel

```
class Person {
  String name, vorname;
  int tag, monat, jahr;
}
```

Gliederung

Erzeugen von Objekten - Instanzen einer Klasse

Der Name der Klasse kann als neuer Datentyp verwendet werden.
Damit werden Referenzvariablen vereinbart, also noch *keine* Objekte vom Typ der Klasse!

Beispiel

```
Person clara;
```

Objekte (Instanzen der Klasse) können mit dem Operator `new` erzeugt werden (bei der Initialisierung oder später per Wertzuweisung). Dabei wird der Klassenname mit runden (!) Klammern angegeben.

Beispiel

```
Person dora = new Person();
clara = new Person();
```

Gliederung

Zugriff auf Attribute

Auf ein Attribut einer Instanz kann zugegriffen werden durch die Schreibweise
Instanzname . Attributname

Beispiel

```
clara.name = "Maier";
clara.vorname = "Clara";
System.out.println (clara.vorname + " " + clara.name);
```

Gliederung

Initialisierung

Die Attribute werden automatisch initialisiert (wenn nicht anders angegeben, mit 0, 0.0, null, false, ...)

Beispiel

```
Person claus = new Person();
System.out.println ("Geburtsjahr: " + claus.jahr);
// Ausgabe: Geburtsjahr: 0
int jahr;
System.out.println ("Geburtsjahr: " + jahr);
// Ausgabe: Fehlermeldung
```

Gliederung

Geschachtelte Datentypen

- Klassen dürfen als Komponententyp bei Feldern auftreten.

Beispiel

```
Person[] familie = new Person [4]; // nur 4 Referenzen
for (int i=0; i<4; i++) // 4 Personen erzeugen
    familie[i] = new Person ();
```

- Ebenso dürfen Attribute von Klassen einen beliebigen Datentyp haben, also auch z.B. Felder oder Klassen.

Beispiel

```
class Familie {
  Person vater, mutter;
  Person[] kinder;
  int[] alter;
}
```

Geschachtelte Datentypen

- Der Typ eines Attributs einer Klasse darf auch vom Typ *dieser* Klasse sein. Dies ist erlaubt, weil es sich nur um Referenzen handelt.

Beispiel

```
class Stammbaum {
   String name, vorname;
   Stammbaum vater, mutter;
}
```

Gliederung

Konstruktoren

Bisher

- zuerst ein Objekt erzeugen
- dann seine Attribute per Wertzuweisung definiert.

Dies ist umständlich und fehleranfällig.
Alternative: Definition eines Konstruktors

Syntax Konstruktordefinition

Klassenname (Typ Parametername, ...) {
    Instanzvariable = Parametername;
    ...
}

Syntax Konstruktoraufruf

new Klassenname (Wert, ...);

Beispiel zu Konstruktoren

Definition einer Klasse mit Konstruktor

```
class PersonK {
   String name, vorname;
   PersonK (String n, String v) {
      name = n;
      vorname = v;
   }
}
```

Erzeugen einer Instanz dieser Klasse

```
PersonK nora = new PersonK ("Schneider", "Nora");
```

Bemerkungen zu Konstruktoren

- Die angegebenen Werte werden der Reihe nach an die Parameternamen übergeben.
- Anzahl und Typ muss übereinstimmen.
- Mittels `new` wird der benötigte Speicher beschafft, und die angegebenen Anweisungen im Konstruktor ausgeführt.
- Es dürfen mehrere Konstuktoren für eine Klasse definiert werden. Sie müssen unterschiedliche Argumentlisten haben.
- *Standardkonstruktor:* Konstruktor ohne Argumente, wird vom Compiler automatisch erzeugt – sofern keine Konstruktoren definiert werden.

Gliederung

Referenzen - Wertzuweisung

Im Gegensatz zu den Grundtypen (Zahlen, Zeichen und boolsche Typen) werden Klassen wie alle Objekte als Referenzen behandelt.

Das bedeutet, daß eine Zuweisung über den Zuweisungsoperator = nur eine neue Referenz (einen *Aliasnamen* für das Objekt) erzeugt, jedoch keine echte Kopie.

Achtung

Bei Kopien mit Hilfe des Zuweisungsoperators wird bei Grundtypen der Wert kopiert, bei Referenztypen (Klassen, Felder, Strings) nur die Referenz, nicht der Wert!

Spezielle Referenz null verweist auf kein Objekt!

Referenzen - Zuweisung

Beispiel

```
PersonK nora = new PersonK ("Schneider", "Nora");
PersonK nora2 = nora; // nora2.name = "Schneider"...
nora.name = "Schuster";  // aendert auch nora2.name
```

`nora` und `nora2` sind lediglich zwei unabhängige Referenzen auf das selbe Objekt.

Beispiel

```
nora = null; // Verbindung Referenz/Instanz loesen
nora.name = "Schmidt";  // Fehler!!!
```

Referenzen - Vergleich von Objekten

Mit dem logischen Vergleich `a==b` werden nur die Referenzen verglichen, nicht jedoch der Inhalt der Attribute.

Beispiel

```
PersonK paul1 = new PersonK ("Mac Intosh", "Paul");
PersonK paul2 = new PersonK ("Mac Intosh", "Paul");
if (paul1==paul2) ...
// Bedingung ergibt false, obwohl Inhalt gleich ist
```