

Einführung

Mit Arrays konnten wir mehrere gleichartige Datenelemente (z.B. `int`) zu einer größeren Einheit zusammen fassen und unter einem Namen ansprechen. *Klassen* erfüllen einen ähnlichen Zweck: Sie erlauben die Zusammenfassung mehrerer Datenelemente unter einem gemeinsamen Namen, wobei aber die Datenelemente im Gegensatz zu denen der Arrays verschiedenartig sein können. Die Klasse gibt an, welche Datentypen vorkommen. Mit Hilfe der Klasse kann dann ein konkretes *Objekt* mit den in der Klasse vorgegebenen Datentypen angelegt werden.

Bei der Arbeit mit den Datenbanken haben wir ähnlich gearbeitet: Wir haben eine Tabelle (ein Array) bestehend aus Zeilen bzw. Datensätzen (Objekten) erstellt, wobei es für jede Zeile eine Menge von Attributen bzw. Feldnamen gab. Die Felder einer Zeile hatten verschiedene Datentypen, die wir festgelegt hatten. Die Festlegung der Datentypen einer Datenbanktabelle entspricht in etwa der Deklaration einer Klasse, sozusagen als Schablone. Erst beim Ausfüllen einer Tabellenzeile wird dann ein konkretes Objekt einer Klasse angelegt. Dementsprechend ist eine Datenbanktabelle ein Array aus Objekten, die zu einer Klasse gehören.

Zur Erinnerung: Es wurde schon im Zusammenhang mit Arrays erwähnt, dass auch diese letztlich nur Objekte sind, die über einen Pointer oder Zeiger angesprochen werden. Die Klasse `Array` ist allerdings in Java schon ohne weiteres Zutun deklariert.

Deklaration und Verwendung von Klassen und Objekten

Deklaration von Klassen

Als Beispiel soll eine Klasse `Date` deklariert werden, die verschiedene Datentypen eines Datums zusammen fasst:

```
class Date
{ int tag;
  String monat;
  int jahr;
}
```

Die Variablen einer Klasse nennt man *Felder* oder *Attribute* (siehe Datenbanktabellen). Klassen werden in einer eigenen Datei oder auf der äußersten Ebene einer Datei deklariert. Beispiel:

```
class Date {...}
class Time {...}
class MeinProgramm {...}
```

Beim Kompilieren werden dann die separaten Dateien `Date.class` und `Time.class` angelegt.

Deklaration von Objekten

Durch Voranstellen des Klassennamens können Objekte dieser Klasse deklariert werden:

```
Date datum1, datum2, datum3;
```

Im Beispiel ist nun Speicherplatz für drei Zeiger geschaffen worden, die auf Datumsobjekte zeigen sollen. Der Platz für die Objekte selbst ist damit noch nicht reserviert.

Erzeugung von Objekten

Wie Arrayobjekte müssen auch Objekte einer Klasse mittels `new` **erzeugt** werden.

```
datum1 = new Date();  
datum2 = new Date();  
datum3 = new Date();
```

Nun ist für die drei Objekte der Speicher reserviert worden und die Zeiger zeigen auf die jeweiligen Speicherstellen. Man beachte das Klammerpaar `()` hinter `Date`. Hier können später diverse Parameter schon bei der Erzeugung eines Objektes übergeben werden.

Zugriff auf Objekte

Mit z.B.

```
datum1.tag    = 19;  
datum1.monat  = "Mai";  
datum1.jahr   = 1956;
```

werden Werte den einzelnen Variablen des Objektes zugewiesen.

Freigabe von Objekten

Objekte einer Klasse werden wie Arrayobjekte automatisch durch den in Java eingebauten *Garbage Collector* „entsorgt“, wenn sie nicht mehr durch einen Zeiger referenziert werden. Man muss sich also im Gegensatz zu älteren Sprachen wie Pascal oder C++ um nichts kümmern.

Zuweisungen von Objekten

Objektvariable können einander zugewiesen werden, falls ihre zugehörigen Objekte den gleichen Typ haben. Mit der Zuweisung

```
datum2 = datum1;
```

zeigt die Objektvariable `datum2` auf das selbe Objekt wie `datum1`. Es handelt sich also um die Zuweisung von Zeigern und nicht um die Zuweisung von Inhalten von Objekten (Achtung!). Inhalte müssen einzeln zugewiesen werden:

```
datum2.tag    = datum1.tag;  
datum2.monat  = datum1.monat;  
datum2.jahr   = datum1.jahr;
```

Die Zuweisung `datum2 = null;` besagt, dass `datum2` auf kein Objekt mehr zeigt.

Vergleiche von Objekten

Ähnlich wie bei den Zuweisungen verhält es sich bei Vergleichen. Mit

```
if (datum1 == datum2) ...
```

wird lediglich überprüft, ob die beiden Zeiger auf die gleiche Speicherstelle zeigen. Das kennen wir bereits von den Arrays. Zum Vergleich der Inhalte der Objekte muss man sich eine Methode schreiben:

```
boolean sindGleich(Date x, Date y)
{ return (x.tag == y.tag) &&
        x.monat.equals(y.monat) && // String-Vergleich
        (x.jahr == y.jahr);
}
```

Objekte als Rückgabe von Methoden

Methoden können in Java ohne Weiteres nur einen einfachen Wert zurück liefern. Möchte man mehrere Werte zurück geben, muss man sie in einer Klasse verpacken und ein Objekt dieser Klasse zurück geben. Beispiel:

```
static Date datumseingabe()
{ Date datum = new Date();
  ...
  return datum;
}
```

Aufgaben:

1. Schreibe ein Hauptprogramm für die Konsole (mit den Klassen `In` und `Out`) mit einer Klasse `Date` in der gleichen Java-Datei, deklariere drei Objektvariablen `datum1`, `datum2` und `datum3`, erzeuge die drei zugehörigen Objekte und weise ihnen Werte zu.
2. Formuliere eine Methode `static void datumsausgabe(Date x)`.
3. Formuliere eine Methode `static Date datumseingabe()`, in der auch der Monat als Zahl eingegeben wird.
4. Teste die Methoden mit geeigneten Daten.
5. Eine Datenbanktabelle *Geburtstagsmerkzettel* besitzt die Attribute *Name* und *Geburtsdatum*. Dies soll in Java nachgebildet werden.

Deklariere dazu eine neue Klasse `Merkzettel` mit einem Namen und einem Datum (benutze dazu die schon bestehende Klasse `Date`).

Nun lege ein Array aus `Merkzetteln` an und gib einige Daten ein und aus.