4. Datentypen

- Beschreibung von elementaren Datentypen
  boolean, byte, short, int, long, float, double, char
- Auswertung von Ausdrücken
- Arrays

Bisher

- Algorithmus benötigt als Eingabe Daten und liefert als Ergebnis wieder Daten zurück.

- Der Algorithmus macht dabei gewisse Annahmen über die Eingabedaten.
  - Voraussetzung für die Korrektheit des Algorithmus (Vorbedingung)
- Der Algorithmus liefert als Ergebnis nur bestimmte Daten aus einer Wertemenge zurück. (Nachbedingung)
- Beispiel
  - ggt
    - Liefert zu zwei ganzen Zahlen größer Null wieder eine ganze Zahl größer Null
  - getPi
    - Liefert zu einer Gleitpunktzahl (Abbruchbedingung) eine Gleitpunktzahl zurück.

Datentypen

- Datentypen
  - Repräsentation einer Menge von Datenelementen
  - und die zugehörigen Operationen, die auf den Datenelementen ausgeführt werden können.
- Beispiel
  - Datentyp **int**
    - Repräsentation aller ganzen Zahlen in dem Intervall
      -2.147.483.648 ... 2.147.483.647
    - Operationen +, -, /, *, %
- Verwendung von Datentypen bei der Deklarationen von Variablen

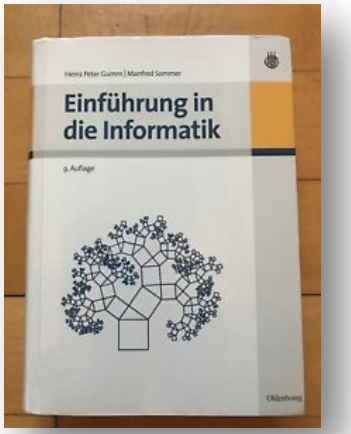
```
double x;          // Variable x ist vom Typ double
int a;             // Variable a ist vom Typ int
```

Datentypen in Java

- Java bietet nur wenige vorgefertigte Datentypen an.
  - Boolesche Werte: boolean
  - Ganzzahlige Datentypen: byte, char, short, int, long
  - Gleitpunktzahlen: float, double
- Zusätzlich wird noch ein Datentyp String unterstützt.
  - Zeichenketten: String
- Java bietet Techniken zur Konstruktion eigener Datentypen
  - Arrays
  - Aufzählungstypen
  - Klassen

Repräsentation von Information

- Viele Abstraktionsebenen



Interpretation

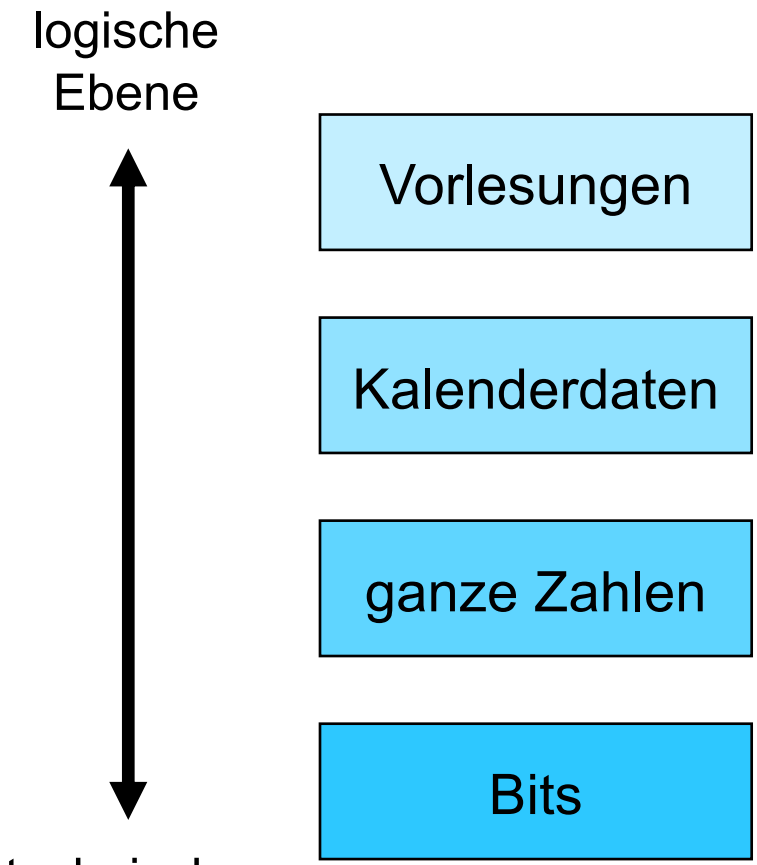
Repräsentation

ISBN 978-3-486-70641-3

```
01001001 01010011 01000010 01001110
00100000 00111001 00110111 00111000
00101101 00110011 00101101 00110100
00111000 00110110 00101101 00110111
00110000 00110110 00110100 00110001
00101101 00110011
```

Motivation

mögliche Abstraktionshierarchie in einem Informationssystem der Universität:

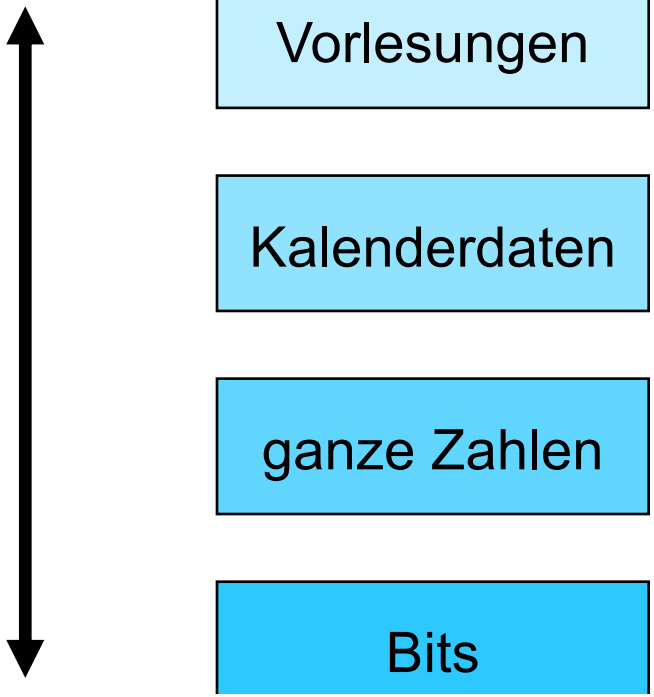


Termin hinzufügen, ...	Termine, Räume, Teilnehmer, ...
Tage addieren, ...	Tag, Monat, Jahr, ...
addieren, ...	-2.147.483.648, 0, 1, 1024
and, or, ...	00011001,00001100, 00000010, ...

Ein Rechner verfügt zunächst nur über wenige, primitive Operationen!

Motivation

mögliche Abstraktionshierarchie in einem Informationssystem der Universität:



Termin hinzufügen, ...	Termine, Räume, Teilnehmer, ...
Tage addieren, ...	Tag, Monat, Jahr, ...
addieren, ...	-2.147.483.648, 0, 1, 1024
and, or, ...	00011001,00001100, 00000010, ...

Bemerkung:

- Der Benutzer einer Abstraktionsebene hat i.A. keine Kenntnisse über die Repräsentation der Daten, sondern kennt nur die Bedeutung der Operationen!

4.1 Beschreibung von Datentypen

- Ein Datentyp besteht aus einer Grundmenge (Sorte, Trägermenge) und einer Menge von Operationen. Jede der zugehörigen Operationen besitzt als Parameter oder als Ergebnis zumindest einmal die Trägermenge.
- Beziehen sich die Parameter und Ergebnisse der Operationen nur auf die Trägermenge, so sprechen wir von einsortigen Datentypen. Ansonsten von mehrsortigen oder heterogenen Datentypen.

Klassen sind auch Datentypen und die sind mehrsortig

Beispiel eines einsortigen Datentyps (boolean)

SORT	boolean		
OPS	true:		→ boolean
	false:		→ boolean
	not:	boolean	→ boolean
	and:	boolean × boolean	→ boolean
	or:	boolean × boolean	→ boolean
	xor:	boolean × boolean	→ boolean

„Signaturen“ der Operationen.

Definieren nur die Syntax nicht die Semantik!

boolean ist ein Datentyp
seine Wertemenge = {true,false}
auf diese Sorte definieren wir diese Opertaionen:
not
and
or
xor

Begriffe

Einsortige Datentypen: Arbeitet nur mit einer Art von Daten. Beispiel: int, float, boolean.
Mehrsortige Datentypen: Kombiniert verschiedene Datentypen oder enthält Daten unterschiedlicher Typen.
Beispiele: Object[]-Arrays, benutzerdefinierte Klassen wie Person

Einsortige Datentypen sind solche, bei denen alle Werte und Operationen nur auf einer einzigen Art von Datentyp operieren. In Java sind dies die primitiven Datentypen wie int, float, double, char, und boolean. Jeder dieser Typen bezieht sich auf eine einzige Sorte von Daten.
Beispiel: int zahl1=5 , int zahl2 =10 , int ergebnis =zahl1+zahl2; hier arbeiten wir mit int werten und die Addition bezieht sich nur auf Ganzzahlen daher sprechen wir von einem einsortigen Datentyp

- Die Stelligkeit einer Operation bezeichnet die Anzahl ihrer Parameter.

- Die Signatur einer Operation besteht aus der Folge der zu den Parametern zugeordneten Datentypen und dem Datentyp des Resultats.
  - In Java ist bei einer Methode der Name der Methode Teil der Signatur, aber nicht der Ausgabetyp.
    (Es können keine zwei Methoden, die sich nur im Rückgabetyp unterscheiden, existieren.)

- Operationen mit Stelligkeit 0 wie z.B. true liefern nur einen Wert zurück; sie können somit als Konstanten aufgefasst werden.

Mehrsortige Datentypen:
Mehrsortige oder heterogene Datentypen sind solche, die verschiedene Typen von Daten zusammenhalten oder verarbeiten können. In Java können wir diese durch Klassen, Arrays von Objekten, oder spezifische Datenstrukturen wie List und Map erreichen.

```
Beispiel:Object[] gemischteDaten = new Object[3];
        gemischteDaten[0] = 42;          // int
        gemischteDaten[1] = "Hallo";     // String
        gemischteDaten[2] = 3.14;        // double

        for (Object element : gemischteDaten) {
            System.out.println(element);
```

Erklärung: In diesem Beispiel verwenden wir ein Object[]-Array, das Werte verschiedener Typen (int, String, double) speichern kann. Das Array selbst ist mehrsortig, da es unterschiedliche Datentypen enthält.

Schreibweisen

Stelligkeit bezeichnet die Anzahl der Parameter, die eine Operation benötigt.
Man kann sich dies als die Anzahl der Eingaben vorstellen, die eine Funktion oder Methode benötigt, um ein Ergebnis zu produzieren.

Beispiel:
Eine Operation, die zwei Zahlen addiert (z.B. a + b), hat die Stelligkeit 2, weil sie zwei Parameter benötigt.
Eine Methode zum Berechnen der Quadratwurzel einer Zahl hat die Stelligkeit 1, weil sie nur einen Parameter benötigt.

Signatur einer Operation beschreibt nicht nur die Anzahl der Parameter, sondern auch die Typen dieser Parameter und den Datentyp des Ergebnisses. Die Signatur bietet eine vollständige Beschreibung der Operation, sodass man genau weiß, welche Eingaben erforderlich sind und welcher Typ von Ergebnis zurückgegeben wird.

Die Signatur der Methode zur Addition von zwei Ganzzahlen in Java könnte int add(int a, int b), die signatur ist : int add(int a, int b) -> int anderes Beispiel : double sqrt(double x) -> double

- Es gibt verschiedene Schreibweisen für den Aufruf einer Operation f:

f	nullstellige Operation weil f benötigt keinen Parameter
f(a, b, ...)	Methodenaufruf
a f b	Infixform (nur für 2-stellige Operationen)
f a b ...	Präfixform
a b ... f	Postfixform
ā	Spezialnotationen

kurz:
Stelligkeit: Anzahl der Parameter:
Signatur:Typen der Parameter und des Ergebnisses:

HP 12c: Postfix-Taschenrechner

- Beispiele
  - true
  - fibo(5);
  - 2 + 3
  - 2 3 + 5* entspricht (2+3)*5
  - ! b Spezialnotation für die der Negation
- Bisher haben wir noch nicht geklärt, was die einzelnen Operationen leisten!

Semantik der Operationen durch eine Funktionstabelle

- Angabe der Semantik der Operationen durch Funktionstabellen

x	not x
true	false
false	true

x	y	x and y	x or y	x xor y
true	true	true	true	false
true	false	false	true	true
false	true	false	true	true
false	false	false	false	false

- **Bemerkungen**
  - Diese Herangehensweise erfordert eine kleine Trägermenge!
  - Gesetzmäßigkeiten wie z.B.

    not (x and y) == (not x) or (not y) de Morgen Gesetz

    sind anhand von Funktionstabellen nur schwierig erkennbar.
  - Bei diesen komplexen Ausdrücke ist bereits die Reihenfolge der Auswertung wichtig zu wissen.
    - Durch Klammerung kann die Reihenfolge festgelegt werden.

Literale

- Literale sind (unmittelbar vom Programmierer eingebbare) Bezeichnungen für die Werte eines Datentyps.
- Es gibt folgende Arten von Literalen:
  - Die Literale des Datentyps *boolean*
    - false und true
  - *Ganzzahlige* Literale
    - z. B. 2, 17, -3 und 32767
  - Literale für *Gleitpunktzahlen*
    - z. B. 3.14, 1E-6
  - Literale für *Zeichen* und *Zeichenketten*
    - z. B. 'A', "Hallo OOP"

Unterscheidung zwischen Syntax und Semantik
Syntax: Dies sind die Regeln und die Struktur, wie Code geschrieben werden muss, damit er von einem Compiler oder Interpreter erkannt wird. Syntaxfehler treten auf, wenn der Code diese Regeln nicht einhält.

Beispiel: In Java muss eine Anweisung mit einem Semikolon (;) enden. Das Fehlen eines Semikolons führt zu einem Syntaxfehler.
Semantik: Dies beschreibt die Bedeutung des Codes und wie er sich verhält, wenn er ausgeführt wird. Semantikfehler treten auf, wenn der Code zwar syntaktisch korrekt ist, aber nicht das gewünschte oder erwartete Verhalten zeigt.

Beispiel: Wenn du eine Variable deklarierst und versuchst, mit ihr eine Division durch null durchzuführen, ist der Code syntaktisch korrekt, aber semantisch führt er zu einem Laufzeitfehler, weil die Operation mathematisch nicht definiert ist.
Semantik in der Programmierung – Beispiele
Arithmetische Operationen:

Syntax: int result = 10 + 5;
Semantik: Diese Anweisung bedeutet, dass 10 und 5 addiert werden und das Ergebnis (15) in der Variablen result gespeichert wird.
Schleifen und Kontrollstrukturen:

Syntax: for(int i = 0; i < 10; i++) { System.out.println(i); }
Semantik: Die Schleife iteriert von 0 bis 9 und druckt jede Zahl auf die Konsole. Die Semantik beschreibt hier, was die Schleife tut: Sie führt den Codeblock zehnmal aus und erhöht dabei i jedes Mal um 1.
Variablendeklaration und -zuweisung:

Syntax: int x = 5;
Semantik: Dies bedeutet, dass eine Ganzzahlvariable x deklariert und ihr der Wert 5 zugewiesen wird. Die Semantik gibt an, dass x nun den Wert 5 enthält und dieser Wert in späteren Berechnungen verwendet werden kann.
Funktionsaufruf:

Syntax: int result = add(2, 3);
Semantik: Der Funktionsaufruf add(2, 3) bedeutet, dass die Funktion add mit den Argumenten 2 und 3 aufgerufen wird, und das Ergebnis dieser Addition (5) wird in der Variablen result gespeichert.
Semantische Fehler
Semantische Fehler treten auf, wenn der Code zwar syntaktisch korrekt ist, aber nicht das tut, was der Programmierer beabsichtigt hat. Diese Fehler können schwieriger zu finden und zu beheben sein, weil sie oft erst zur Laufzeit auftreten.

Zusammenfassung
Syntax: Wie der Code geschrieben wird (Regeln, Grammatik).
Semantik: Was der Code bedeutet oder tut (Verhalten, Logik).
Semantische Fehler: Fehler, die entstehen, wenn der Code zwar korrekt geschrieben ist, aber nicht das gewünschte Verhalten zeigt.
Semantik in der Programmierung ist entscheidend, um sicherzustellen, dass der Code nicht nur richtig geschrieben ist, sondern auch das tut, was er tun soll.