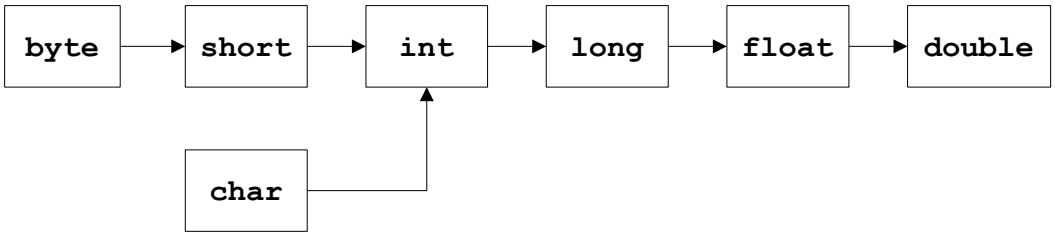
gds2 Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

Implizite und explizite Typkonvertierungen bei Rechenausdrücken in Java

Werden bei einer Zuweisung, bei der Auswertung eines arithmetischen Ausdrucks oder beim Aufruf von Methoden unterschiedliche Datentypen verwendet, so müssen diese zunächst einander angeglichen werden. Java unterscheidet zwischen erweiternder und einschränkender Typkonvertierung.

1 Erweiternde Typkonvertierung

Die erweiternde Typkonvertierung wird automatisch (implizit) vom Compiler vorgenommen, wenn ein kurzer Datentyp einem längeren zugewiesen wird. In diesem Fall kann keine Datenverfälschung auftreten.



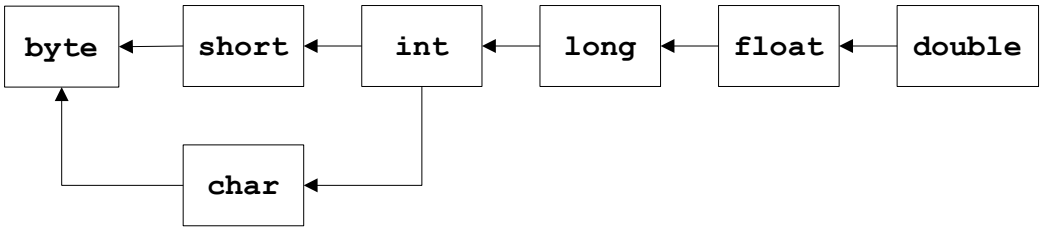
Beispiel:

```
int   iZahl = 7;
float fZahl = 12.5f;
fZahl = iZahl;         // OK
iZahl = fZahl;         // Compilerfehler: "type mismatch"
```

Achtung: Ein float kann zwar mindestens genauso große Werte aufnehmen wie ein long, aber seine Genauigkeit ist auf 8 Stellen beschränkt. Werden also größere int- oder long-Werte einem float zugewiesen, so können sie nicht mit voller Genauigkeit konvertiert werden. Dieser Verlust an Genauigkeit wird allerdings vom Compiler nicht angezeigt.

2 Einschränkende Typkonvertierung

Wenn ein langer Datentyp einem kürzeren zugewiesen wird, so wird das vom Compiler verhindert, da Datenverlust eintreten kann. Die Fehlermeldung heißt „type mismatch".
Soll die Zuweisung trotzdem vorgenommen werden, so kann der Programmierer diese einschränkende Konvertierung mit Hilfe des ***Typecast-Operators*** manuell (explizit) erzwingen.



Beispiel:

```
int   iZahl = 7;
float fZahl = 12.5f;
iZahl = (int) fZahl; // jetzt OK, aber Datenverlust, in iZahl steht 12
```

ALLGEMEIN: zielWert = (datentyp) quellWert
datentyp muss zum Datentyp von zielWert zuweisungskompatibel sein.

3 Regeln zur Typkonvertierung bei arithmetischen Ausdrücken

Auch bei der Berechnung von arithmetischen Ausdrücken nimmt Java Typkonvertierungen vor.

Dabei gelten folgende Regeln:

1. Sind an einer Berechnung *nur ganzahlige* Operanden beteiligt, so ist der Ergebnistyp:
   - long wenn ein long-Operand beteiligt ist
   - int in allen anderen Fällen (auch wenn nur short- oder byte-Typen beteiligt sind)

2. Sind an einer Berechnung *float oder double*-Operanden beteiligt, so ist der Ergebnistyp:
   - double wenn ein double-Operand beteiligt ist
   - float in allen anderen Fällen

Beispiele :

```
short sZahl1 = 10,
      sZahl2 = 5;
int   iZahl;

sZahl1 = sZahl1 + sZahl2;  // Kompilerfehler: „type mismatch”,
                           // da short + short -> int (s. Regel 1)
iZahl1 = sZahl1 + sZahl2;  // OK, oder...
sZahl1 = (short)(sZahl1 + sZahl2);

double dVerbrauch100km = 7.5, dVerbrauch;
int    iKm  = 320;

dVerbrauch  = iKm / 100 * dVerbrauch100km;
/*
 * kein Compilerfehler, aber falsches Ergebnis
 */
dVerbrauch  = iKm / 100.0 * dVerbrauch100km;
/*
 * richtiges Ergebnis
 */

dVerbrauch  = iKm * dVerbrauch100km / 100;
/* richtiges Ergebnis, da von links nach
 * rechts ausgewertet wird
 */
```