Java-Quelltext übersetzen und ausführen

```
# Projektverzeichnis erstellen.
mkdir projekt
# In Projektverzeichnis wechseln.
cd projekt
# Quelltextdatei erzeugen und editieren. (SPEICHERN!)
notepad MyClass.java
# Quelltextdatei mit Java Compiler übersetzen.
# Erfolgt keine Ausgabe, war die Übersetzung erfolgreich.
javac MyClass.java
# Mit dem Java Launcher eine Klasse starten (deren main-Methode).
java MyClass
```

Wozu brauchen wir Datentypen?

- Ein Datentyp definiert, wie ein Bitmuster im Speicher zu interpretieren ist. Beispiel: Was bedeutet die Bitfolge `011001`?
- Ein Datentyp legt fest, wie viel Speicher für eine Variable zu reservieren ist. Beispiel: Eine Variable vom Typ `int` belegt 32 Bits (also 4 Bytes) im Speicher.
- Ein Datentyp definiert den Wertebereich für eine Variable. Beispiel: Der Datentyp `byte` lässt nur Werte im Bereich `-128` bis `+127` zu.
- Ein Datentyp legt fest, welche Operationen mit einer Variablen zulässig sind. Beispiel: Einen `int` kann man multiplizieren, aber einen `String` hingegen nicht.
- Ein Datentyp liefert dem Compiler zusätzliche Informationen, damit er die typkonforme Verwendung der Variablen prüfen kann.
- Datentypen legen die Intention für Variablen fest und fördern damit die Verständlichkeit des Quelltextes.

In Java gibt es zwei Kategorien von Datentypen:

- Primitive Datentypen
- Referenzdatentypen

Hinweis: Für jeden primitiven Datentyp existiert in Java ein korrespondierender Referenzdatentyp - sogenannte Wrapper-Klassen. Beispiel: `byte` und `Byte`, `char` und `Character`, `double` und `Double`.

Um den Wertebereich eines primitiven Datentyps zu ermitteln, verwende dessen zugehörige Wrapper-Klasse. Beispiel:

```
Byte.MIN_VALUE
Byte.MAX_VALUE
Integer.MAX_VALUE
Integer.MIN_VALUE
```

Rundungsfehler bei Datentyp `double` und `float`

Mit den Datentypen `double` und `float` können wir Zahlen mit Nachkommastellen abspeichern. Hier kann es jedoch zu Rundungsfehlern kommen. Für `double` gilt: Ungefähr 15 signifikante Ziffern können exakt dargestellt werden. Bei Datentyp `float` sind es hingegen nur etwa 7.

Die *betragsmäßig* größte Zahl ist bei `double` etwa 1.8E308 und die betragsmäßig kleinste Zahl ist 4.9E-328. (E-328 bedeutet "10 hoch -328").

Achtung: Manche Dezimalzahlen, z.B. 0.1, sind im Binärsystem nicht exakt darstellbar. Diese können nur gerundet abgespeichert werden.

Faustregeln bei Typkonvertierungen

Merke:

- Ganze Zahlen haben den Datentyp `int`.
- Zahlen mit Nachkommastellen bzw. Dezimaltrenner (`.`) haben den Datentyp `double`.
- Verwendet man die wissenschaftliche Notation (z.B. `2e-3`) so hat dieser Wert den Datentyp `double`.
- Um eine Gleitkommazahl als Float anzugeben, verwende den Suffix `f`. Beispiel: `2.5f`.

Regeln:

- Wenn man zwei `int` Werte miteinander verrechnet, entsteht wieder ein Ergebnis vom Typ `int`. Nachkommastellen werden abgeschnitten.
- Verrechnet man zwei Werte miteinander, wobei mindestens ein Wert ein `double` ist, dann ist das Gesamtergebnis vom Typ `double`.
- Verrechnet man zwei `byte` Werte miteinander, ist das Ergebnis vom Typ `int`.
- Verrechnet man zwei `short` Werte miteinander, ist das Ergebnis vom Typ `int`.
- Verrechnet man zwei `float` Werte miteinander, ist das Ergebnis vom Typ `float`.

Zahlen angeben in verschiedenen Zahlensystemen

In Java können Zahlen im Binär-, Oktal-, Dezimal- und Hexadezimalsystem angegeben werden.

Binäre Zahlen beginnen mit Präfix `0b`, oktale Zahlen beginnen mit Präfix `0` und hexadezimale Zahlen beginnen mit Präfix `0x`. Ohne Präfix werden Zahlen als Dezimalzahlen interpretiert.