Programmieraufgaben zu Kapitel 3: Datentypen und Operationen

Aufgabe 1: Ergebnisdatentyp

Der Datentyp eines Ausdrucks hängt oftmals von den Datentypen der Operanden ab. Schreibe daher ein Java-Programm, das dir mit Hilfe des Befehls `System.out.println(…)` die Werte der folgenden Ausdrücke ausgibt..
Erkläre!

- 3/2
- 3.0/2.0
- 3.0/2
- ‘A‘
- ‘A‘ + 0
- ‘A‘ + 7
- ‘A‘ + ‘B‘
- “A“ + “B“

Aufgabe 2: Standarddatentyp int und Wertebereichüberlauf

In den meisten Fällen arbeitet man mit dem Typ `int` (für „integer“). Für die Speicherung von `int`-Zahlen sind 32 Bit vorgesehen, d.h. sie umfassen den Zahlenraum von -2147483648 bis 2147483647. Der Typ `int` ist also groß genug, um die häufigsten Zahlenwerte aufzunehmen. Außerdem sind (bzw. waren früher) viele Rechner auf 32-Bit-Zahlen ausgelegt, d.h. sie rechnen mit `int`-Zahlen am schnellsten.

Schreibe ein Java-Programm `Wertebereichueberlauf.java`, in dem einer `int`-Variablen `i` der Wert $2^{31}-5$ zugewiesen wird und anschließend die Ausgabe der Ergebnisse der Ausdrücke `i, i+1, i+2, i+3, …, i+10` erfolgt. Erkläre!

Hinweis: Die Potenz 2^{31} lässt sich in Java berechnen durch den Aufruf `Math.pow(2,31)`, der jedoch einen `double`-Wert liefert. Um das Ergebnis der `int`-Variablen `i` zuweisen zu können, muss eine Typkonvertierung stattfinden, und zwar durch `(int)(Math.pow(2,31))`. Dadurch wird das `double`-Ergebnis in `int` umgewandelt. Setze also zu Beginn `i = (int)(Math.pow(2,31)) – 5`.

Aufgabe 3: Weitere Standarddatentypen für ganze Zahlen:

Java bietet aber auch andere Datentypen für ganze Zahlen an:

`byte`	8-Bit-Zahl	$-2^7 \dots 2^7-1$	(-128 … 127)
`short`	16-Bit-Zahl	$-2^{15} \dots 2^{15}-1$	(-32768 … 32767)
`int`	32-Bit-Zal	$-2^{31} \dots 2^{31}-1$	(-2147483648 … 2147483647)
`long`	64-Bit-Zahl	$-2^{63} \dots 2^{63}-1$	

So kann man z.B. in einer `short`-Variablen nicht den Wert 100000 speichern, wohl aber in einer `int`-Variablen. Auch kann man einen `short`-Wert einer `int`- oder `long`-Variablen zuweisen, nicht aber einer `byte`-Variablen.

Um den Wertebereich des Datentyps long zu ermitteln, müsste man in den Taschenrechner 2^{63} eingeben. Was gibt der Taschenrechner aus?

Schreibe ein Java-Programm zur Ermittlung des Wertebereichs des Datentyps `long` unter Verwendung von `Math.pow` und der Typkonversion (siehe Aufgabe 2).

Wie lautet die größte durch `long` darzustellende Zahl?

Aufgabe 4: Zuweisungskompatibilität

Untersuche das folgende Programm: An welcher Stelle meldet der Compiler Fehler? Warum? Wie lauten die Fehlermeldungen? Erst denken, dann ausprobieren!

```
short s = 5;
int i,j;
byte b = 1;
long x = 12;
boolean s = true;

i = j;
i = s;
b = 1234;
s = x;
x = true;
```

Aufgabe 5: Datentypen `float` und `double`

Für Kommazahlen gibt es in Java die Datentypen **`float`** und **`double`**, die sich nur in ihrer Größe unterscheiden: `float`-Zahlen werden mit 32 Bits, `double`-Zahlen mit 64 Bits dargestellt. Eine `double`-Zahl hat sowohl mehr Nachkommastellen aus auch einen größeren Exponentenbereich als eine `float`-Zahl.

	kleinste pos. Zahl	größte neg. Zahl
`float`	1.4E-45	3.4E38
`double`	4.9E-324	1.8E308

Mit Gleitkommazahlen kann man wie mit ganzen Zahlen rechnen, wobei die Modulo-Operation nicht verwendet wird. Die **Genauigkeit** ist aber bei beiden Datentypen begrenzt, da alle reellen Zahlen im Rechner durch eine begrenzte Anzahl von Nachkommastellen dargestellt werden. Daher treten jeweils geringfügige Rundungsfehler auf, die sich bei einzelnen Operationen wie z.B. Multiplikationen jedoch summieren.

Schreibe ein Java-Programm `Rundungsfehler.java`, das die Differenz der Zahlen 4.3 und 2.5 berechnet und ausgibt.