Java lernen
kurz & gut

2 Schnelleinstieg

2.1 Hallo Welt (Hello World)

Wir beginnen den Einstieg in Java mit einem simplen Programm, nämlich wie traditionell in den allermeisten Büchern zum Programmierenlernen mit HelloWorld, der Ausgabe eines Grußes auf der Konsole. Das haben wir schon in etwas komplizierterer Form in der Einleitung gesehen. Wir wollen es weiter vereinfachen und damit unsere Entdeckungsreise starten.

Java bietet mit der JShell einen interaktiven Kommandozeileninterpreter, auch Read-Eval-Print-Loop (REPL) genannt. Das erleichtert das schrittweise Erlernen von Java, weil wir nicht gleich zu Beginn eine Vielzahl an Konzepten verstehen oder zumindest hinnehmen müssen. Als weiterer Vorteil wird uns eine Menge Tipparbeit erspart.

Die interaktive JShell erlaubt es uns, kleinere Java-Codeschnipsel auszuprobieren. Der Start erfolgt mit dem Kommando `jshell`:

```
$ jshell
|  Willkommen bei JShell - Version 21
|  Geben Sie für eine Einführung Folgendes ein: /help intro

jshell>
```

Neben dem Hinweis auf die Version zeigt der Prompt `jshell>` an, dass wir nun Java-Befehle eingeben können. Probieren wir es gleich einmal aus:

```
jshell> System.out.println("Hello World from JShell!")
Hello World from JShell!
```

Herzlichen Glückwunsch zur ersten Ausgabe mit Java!

Wir verwenden dabei die später in Kapitel 5 vorgestellten Klassen und Methoden. Hier wird über `System.out` eine Ausprägung der Klasse `java.io.PrintStream` referenziert. Diese ermöglicht Ausgaben auf der Konsole. Funktionalität wird durch sogenannte Methoden, eine Folge von Befehlen, bereitgestellt, in diesem Fall durch die Methode mit dem Namen `println()`, der man in runden Klammern einen Text eingerahmt von doppelten Anführungszeichen übergibt. Aber keine Sorge, Sie müssen noch nicht alle Details verstehen – wir werden diese im Verlaufe dieses Buchs ausführlich besprechen. Jetzt geht es zunächst um die ersten Schritte und ein initiales Gespür. Laden Sie die Begleitsourcen herunter oder tippen Sie die einfachen Beispiele ab, und wenn Sie sich sicher fühlen, dann variieren Sie auch gerne ein wenig.

Wir haben bereits einen netten Gruß auf der Konsole ausgeben können. Das war schon ein guter Anfang. Allerdings wäre es etwas eintönig, nur Texte ohne Variation auszugeben. Um den Gruß anzupassen, lernen wir nun Variablen kennen.

2.2 Variablen und Datentypen

Stellen wir uns vor, wir wollten einige Eigenschaften einer Person mit Java verwalten, etwa Name, Alter, Gewicht, Wohnort, Familienstand usw. Dabei helfen uns sogenannte Variablen. Diese dienen zum Speichern und Verwalten von Werten. Dabei gibt es unterschiedliche Typen (auch Datentypen genannt), etwa für Texte, Zahlen und Wahrheitswerte. Wir schauen uns dies zunächst einmal einführend an und erweitern dann unser Wissen Schritt für Schritt, beispielsweise um ergänzende Infos zu Wertebereichen oder Aktionen mit den Variablen. Zum Einstieg betrachten und nutzen wir folgende Typen:

- `String` – Speichert textuelle Informationen, z. B. "Hallo". In Java werden diese in doppelte Anführungszeichen eingeschlossen und Strings genannt.

- `char` – Speichert einzelne Zeichen, wie z. B. 'a' oder 'B'. Solche Zeichenwerte sind von einfachen Anführungszeichen eingeschlossen.
- `int` und `long` – Speichern Ganzzahlen, wie 123 oder -4711. Für `int` geht der Wertebereich von etwa -2 Milliarden bis 2 Milliarden, für `long` ist er noch viel größer.
- `float` und `double` – Speichern Gleitkommazahlen (mit Vor- und Nachkommastellen), wie 72,71 oder -1,357. Achtung: Weil Java dem amerikanischen Sprachraum entstammt, werden als Dezimaltrenner Punkte statt Kommata verwendet. Somit muss es im Java-Programm `72.71` und `-1.357` heißen. Der Typ `float` besitzt eine geringere Genauigkeit und kann somit weniger Nachkommastellen als der Typ `double` verwalten.
- `boolean` – Speichert Wahrheitswerte als wahr oder falsch. In Java sind diese durch `true` oder `false` repräsentiert.

2.2.1 Definition von Variablen

In Java erzeugen wir eine Variable, indem wir zunächst den Typ, dann einen Namen sowie eine Wertzuweisung nach folgendem Muster (Syntax genannt) angeben:

```
typ variablenname = wert;
```

Man spricht in diesem Zusammenhang von der Definition. Eine solche Definition wie auch andere Anweisungen müssen in Java normalerweise mit einem Semikolon beendet werden. In der JShell ist dies oft optional und man kann das Ganze wie folgt schreiben:

```
typ variablenname = wert
```

Lassen Sie uns etwas experimentieren:

```
jshell> int theAnswer = 41
theAnswer ==> 41

jshell> theAnswer = 42
theAnswer ==> 42

jshell> int oneMillion = 1_000_000
oneMillion ==> 1000000

jshell> double PI = 3.1415
PI ==> 3.1415
```

Die JShell bestätigt jede Definition einer Variablen mit einer Ausgabe, wobei zunächst der Variablenname, dann ==> und schließlich der Wert folgt. Für das Beispiel mit einer Million sehen wir die Angabe von Unterstrichen, die sich zur Separierung, hier von Tausendersegmenten, eignet – allerdings nur bei der Angabe, nicht bei der internen Repräsentation.

Den aktuellen Wert einer Variablen können wir durch Eingabe das Namens abfragen bzw. auslesen:

```
jshell> theAnswer
theAnswer ==> 42
```

Wie oben für `theAnswer` angedeutet, kann einer Variablen durchaus mehrmals im Programmverlauf an unterschiedlichen Stellen ein anderer Wert zugewiesen werden. Denken Sie etwa an einen Punktestand in einem Spiel, einen Temperaturverlauf pro Tag und eine Variable `temperature` oder aber an die Modellierung der Tageszeit von morgens, mittags, nachmittags, abends und nachts. Für derart in sich abgeschlossene Wertebereiche existieren Aufzählungen, die wir in Abschnitt 7.4 kennenlernen werden.

Eine solche Variable wird auch bei dem kürzest möglichen Hello-World-Programm automatisch als unbenannte Variable `$1` erzeugt:

```
jshell> "Shortest hello world ever"
$1 ==> "Shortest hello world ever"
```

Besonderer Typ `String`

Wie Zahlen kann man auch Strings mit + verbinden. Man spricht hier von Konkatenation. Als Folge entsteht ein neuer String.

```
jshell> String vorname = "Michael"
vorname ==> "Michael"

jshell> String name = vorname + " " + "Inden"
name ==> "Michael Inden"
```

Wie gezeigt, sind problemlos mehrere Konkatenationen möglich, ebenso eine Verknüpfung von Zahlen mit einem String:

```
jshell> int value = 4711
value ==> 4711

jshell> "Wert: " + value
$3 ==> "Wert: 4711"
```

Der Typ `String` unterscheidet sich von den anderen vorgestellten Typen dadurch, dass er neben + noch einige weitere spezifische Funktionalitäten bereitstellt. Dies werden wir uns genauer in Kapitel 3 anschauen.

Besonderheit beim Typ `char`

Der Typ `char` repräsentiert ein einzelnes Zeichen:

```
jshell> char character = 'A'
character ==> 'A'
```

Interessanterweise lassen sich mathematische Aktionen wie + und - ausführen, allerdings entsteht dann ein Zahlenwert vom Typ `int`. Mithilfe eines später vorgestellten sogenannten ***Casts*** kann man durch die Angabe von `(char)` vor einem Ausdruck diesen wieder in ein Zeichen wandeln (vgl. Abschnitt 7.2.2):

```
jshell> character + 5
$34 ==> 70

jshell> (char)(character + 5)
$35 ==> 'F'
```

Wie gezeigt, kann man sich dann per Addition vorwärts durch das Alphabet bewegen. Eine Subtraktion ist ebenfalls möglich, um eine Anzahl an Buchstaben in Richtung A zu springen.

Kurzschreibweise `var`

Als Kurzschreibweise zur Definition von Variablen gibt es die sogenannte Local Variable Type Inference mit dem reservierten Wort `var`. Was bedeutet das genauer?

Zuvor haben wir zur Definition von Variablen folgende Syntax genutzt:

```
typ variablenname = wert
```

Statt einer konkreten Typangabe können wir kürzer `var` schreiben, um vor allem bei längeren Typangaben etwas Tipparbeit zu sparen.

```
var variablenname = wert
```

Java selbst ist so schlau, festzustellen, was der genaue Typ ist. Insbesondere bei längeren Typnamen wird der Sourcecode durch Nutzung von `var` etwas kürzer und eventuell auch lesbarer:

```
jshell> var theAnswer = 42
theAnswer ==> 42

jshell> var PI = 3.1415;
PI ==> 3.1415

jshell> var name = "Michael"
name ==> "Michael"
```

Eine derart definierte Variable hat dann genau den von Java ermittelten Typ, also ist `theAnswer` vom Typ `int` und `name` vom Typ `String`.

An dem Beispiel möchte ich noch einmal explizit auf zwei Dinge eingehen: Zum einen ist es in der JShell oftmals möglich, auf ein Semikolon am Ende der Anweisungen zu verzichten. Hier wird zur Demonstration nur bei der Definition von `PI` (π)

ein solches verwendet, nicht aber für die beiden anderen Definitionen. Zum anderen kann man zwar `var` statt `int` nutzen, aber dies ist nicht sinnvoll, da es genauso viele Zeichen sind und man somit nichts gewinnt.

Konsolenausgeben – `System.out.println()`

Einleitend hatten wir schon `System.out.println()` genutzt, um Texte auf der Konsole ausgeben zu können. Das lässt sich auch mit Variablen kombinieren:

```
jshell> var name = "Michael"
name ==> "Michael"

jshell> System.out.println("Hello " + name)
Hello Michael
```

Auf ein Detail möchte ich nochmals hinweisen: Bei `println()` handelt es sich um eine in Java integrierte sogenannte Methode. Diese dienen dazu, gewisse Aktionen standardisiert bereitzustellen. Methoden schauen wir uns in Abschnitt 2.5 genauer an.

Besonderheit Deklaration

Manchmal weiß man zwar bereits, dass man das Ergebnis einer Berechnung in einer Variablen speichern möchte, hat diese Berechnung aber noch nicht ausgeführt. Für solche Fälle ist es möglich, Variablen ohne Wertzuweisung zu notieren, man spricht dabei von Deklarieren. Die Wertzuweisung erfolgt somit nicht direkt, sondern erst später im Ablauf – hier durch eine Konsolenausgabe als zwischenzeitliche Aktion angedeutet:

```
jshell> int age
age ==> 0

jshell> System.out.println("Something in between!")
Something in between!

jshell> age = 15
age ==> 15
```

Oftmals ist die bereits besprochene Definition, also die Kombination aus Deklaration und direkter Wertzuweisung, zu bevorzugen, da sich dies klarer liest und Fehlverwendungen vorbeugt. Außerdem kann man bei einer Deklaration `var` nicht verwenden, weil kein Wert angegeben wird und somit der Typ nicht ermittelt werden kann.

Konstanten – finale Variablen

Mithilfe des Schlüsselworts `final` vor der Definition einer Variablen machen wir diese unveränderlich und erzeugen somit eine Konstante:

```
final double PI = 3.1415
PI = 3.14;  // error: cannot assign a value to a final variable
```

Keine Konstanten in der JShell

Bitte beachten Sie, dass die JShell `final` nicht so unterstützt wie ein Java-Programm. Zwar kann man `final` angeben, aber es entsteht keine Konstante, sondern eine Variable. Dadurch ist eine zweite Zuweisung in der JShell möglich und verändert einfach den Wert:

```
jshell> final double PI = 3.1415
PI ==> 3.1415

jshell> PI = 3
PI ==> 3.0
```

2.2.2 Bezeichner (Variablennamen)

Ohne es explizit zu erwähnen, haben wir uns in den Beispielen bei der Benennung von Variablen an ein paar Regeln gehalten. Zunächst einmal muss jede Variable (und auch die später vorgestellten Methoden und Klassen) durch einen eindeutigen Namen, auch Identifier oder Bezeichner genannt, gekennzeichnet werden.

Bei der Benennung sollte man Folgendes beachten:

- Namen sollten mit einem Buchstaben beginnen[1] – Ziffern sind als erstes Zeichen eines Variablennamens nicht erlaubt.
- Namen können danach aus einem beliebigen Mix aus Buchstaben (auch Umlauten), Ziffern, $ und _ bestehen, dürfen aber keine Leerzeichen enthalten.
- Bestehen Namen aus mehreren Wörtern, dann ist es guter Stil, die sogenannte CamelCase-Schreibweise zu verwenden. Bei dieser beginnt das erste Wort mit einem Kleinbuchstaben und danach startet jedes neue Wort mit einem Großbuchstaben, etwa `arrivalTime`, `estimatedDuration`, `shortDescription`.
- Die Groß- und Kleinschreibung von Namen spielt eine Rolle: `arrivalTime` und `arrivaltime` bezeichnen unterschiedliche Variablen.
- Namen dürfen nicht mit den in Java vordefinierten und in Anhang A aufgelisteten Schlüsselwörtern übereinstimmen, demzufolge sind z. B. `public`, `class` oder `int` keine gültigen Namen für Variablen. Die Begriffe `PublicTransport`, `classic` oder `Winter` hingegen sind erlaubt.

Übrigens spielt es für Java keine Rolle, ob Sie sehr kurze (`i`, `m`, `p`), kryptische (`dpy`, `mtr`) oder aber gut gewählte Namen (`daysPerYear`, `maxTableRows`) nutzen. Für Sie und das Verständnis beim Nachvollziehen eines Java-Programms macht das aber einen himmelweiten Unterschied. Natürlich sind kurze Variablennamen wie etwa `x` und `y` zur Bezeichnung von Koordinaten vollkommen verständlich, aber was sagt beispielsweise ein einzelner Buchstabe `m` aus?

[1] Namen dürfen auch mit $ oder _ beginnen, jedoch ist das eher für fortgeschrittene Themen nützlich. Da wir in diesem Buch auf Verständlichkeit setzen, werden wir das nicht nutzen.

Betrachten wir zur Verdeutlichung ein Beispiel:

```
// Gut verständliche Namen
int minutesPerHour = 60
int daysPerYear = 365
int maxTableRows = 25

// NAJA, oft schwieriger verständlich, was die Abkürzungen bedeuten
int m = 60;
int dpy = 365
int mtr = 25

// ACHTUNG: ziemlich schwierig unterscheidbar
String arrivalTime = "16:50"
double arrivaltime = 16.50
```

Die letzten beiden Variablendefinitionen sind ungünstig, da sie kaum zu unterscheiden sind. Besäßen diese denselben Typ, so käme es leicht zu Fehlverwendungen.

Im Listing sehen wir Kommentare. Damit kann man erklärende Zusatzinformationen hinterlegen. In diesem Fall werden mit // einzeilige Kommentare eingeleitet. Alles, was nach dem // bis zum Ende der Zeile folgt, wird bei der Ausführung von Java ignoriert. Weitere Varianten von Kommentaren lernen wir in Abschnitt 2.7 kennen.

2.3 Operatoren im Überblick

Operatoren dienen dazu, Operationen zwischen Variablen und / oder Werten auszuführen. Selbst eine banale Addition ist ein Beispiel, nämlich für den Operator +:

```
jshell> int sum1 = 200 + 50;
sum1 ==> 250

jshell> int sum2 = sum1 + 500;
sum2 ==> 750

jshell> int sum3 = sum2 + 750;
sum3 ==> 1500
```