Programmieren

Teil - 1

Tim Nätebus

Git : $Fluffy Unicorns$

9. Dezember 2024

Vorwort

Dieses Dokument, soll zur Hilfestellung in dem Modul
"Programmieren-1" dienen.

In Programmieren-1, benutzen wir Java mit der OpenJDK-23.

Dieses Dokument, ist in Kapitel unterteilt, die sich mit den Grundlagen
von Java beschäftigen.

Inhaltsverzeichnis

Variablen

Was Genau sind Variablen?

Variablen sind Speicherplätze, die einen Wert speichern können. Der Wert kann sich während der Laufzeit des Programms ändern.

Welche Datentypen gibt es in Java?

Es gibt folgende Datentypen in **Java**:

- **Primitive Datentypen**
- **Komplexe Datentypen**

Primitive Datentypen

Datentyp	Größe	Kann darstellen
byte	8 Bit	Ganze Zahlen
short	16 Bit	Ganze Zahlen
int	32 Bit	Ganze Zahlen
long	64 Bit	Ganze Zahlen
float	32 Bit	Kommazahlen
double	64 Bit	Kommazahlen
char	16 Bit	Zeichen
boolean	1 Bit	Wahr & Falsch
void	–	Kein Wert

Komplexe Datentypen

Datentyp	Beschreibung
String	Zeichenkette
Array	Liste von Elementen
Klasse	Eigene Datentypen

Was ist ein Literal?

Ein Literal, ist einfach ein Wert der einem Datentypen zugeordnet ist. Zum Beispiel:

> **25** ist ein Literal vom Datentyp **int** o. **long** o. **short** o. **byte**

Wie werden Variablen deklariert & initialisiert?

Eine Deklaration der Variable, legt einen Speicherplatz für die Variable fest. Eine Initialisierung, weist der Variable einen Wert zu.
Man kann eine Variable innerhalb eines schrittes deklarieren und initialisieren.

Beispiele

```
.
    int a;       // Deklaration
    a = 5;       // Initialisierung

    int b = 10; // Deklaration und Initialisierung
.
```

Konstanten

Konstanten sind Variablen, die nach Ihrer Initialisierung nicht mehr verändert werden können.
Zu Konstanten sind wesentliche dinge zu beachten:

a) Der Name einer Konstante wird in Großbuchstaben geschrieben.

b) Der Wert einer Konstante wird mit dem Schlüsselwort **final** deklariert.

c) Der Wert einer Konstante wird bei der Deklaration initialisiert.

d) Der Wert einer Konstante kann nicht mehr verändert werden.

e) Der Wert einer Konstante kann nur einmal initialisiert werden.

Beispiel

```
.
    final int MAX = 100; // Konstante Variable
.
```

Initialisierung von Klassen

Wenn wir Klassen benutzen wollen, müssen wir folgendes Beachten:

- Ist die Klasse in der Standardbibliothek enthalten?
- Hab Ich das Paket indem die Klasse enthalten ist, installiert?
- Muss Ich die Klasse importieren?

Aufgrund dessen, das wir nur mit Klassen Arbeiten, die Standardmäßig enthalten sind, wird hier nur auf Standardbibliotheken und Importe eingegangen.

Standardbibliotheken

Java bietet eine Vielzahl von Standardbibliotheken an, die wir benutzen können. Einige Beispiele sind:

- **java.util** - Für die Eingabe von Daten
- **java.io** - Für die Ausgabe von Daten
- **java.awt** - Für die Erstellung von GUIs

Die Standardbibliotheken sind zu finden unter Standardbibliotheken-Java-23

Importierung von Klassen

Beispiel mit der Scanner Klasse

```java
import java.util.Scanner; // Importieren der Scanner Klasse

public class Main {

    public static void main(String[] args) {

        // Initialisierung der Scanner Klasse
        //              mit System.in als Parameter
        Scanner scIn = new Scanner(System.in);

    }
}
```

Wir initialisieren die Variable **scIn** mit der Klasse Scanner als **komplexen** Datentyp.

Operatoren

Was sind Operatoren?

Operatoren sind Symbole, die auf Variablen und Werte angewendet werden, um eine Operation durchzuführen.

Welche Arten von Operatoren gibt es?

DISCLAIMER:

Die Operatoren, die aufgezählt werden, sind die Operatoren, die wir auch in dem Modul "Programmieren-1" benutzen.

Arten	Beispiele
Arithmetische Operatoren	+ | - | * | / | %
Zuweisungsoperatoren	=
Vergleichsoperatoren	< | > | >= | <=
Logische Operatoren	! | && | || | ^
Bitweise Operatoren	| | & | ^ | ~
Inkrement & Dekrement Operatoren	++ | --

Weitere Operatoren

a) *instanceof*

b) Cast Operator: ( )

c) Klammern ( )

d) Indexierung von Arrays [ ]

e) Ternary Operator ? :

Post & Präfix Operator

Es gibt **Zwei** Arten von Inkrement & Dekrement Operatoren, die auch die Arbeitsweise, des Operators, beeinflussen.

*Es sei I eine Variable von Typ **Int**.*

a) **Präfix Notation:**

`++i` - Variable wird **Sofort** Inkrementiert

`--i` - Variable wird **Sofort** Dekrementiert

und dann der **neue** Wert für die Variable verwendet.

b) **Postfix Notation:**

`i++` - Variable wird erst **nach** der Verwendung Inkrementiert

`i--` - Variable wird erst **nach** der Verwendung Dekrementiert

und der **alte** Wert für die Variable verwendet.

Kontrollstrukturen

Kontrollstrukturen dienen dazu, den Ablauf eines Algorithmus zu steuern. Sie geben an, ob bzw. wie oft Schritte (Anweisungen, statements) ausgeführt werden sollen.

Begriffserklärung:

Algorithmus:

ist eine allgemeingültige Verarbeitungsvorschrift.

Programm:

Ist ein Algorithmus in einer Programmiersprache (+ Dokumentation).

Das Programm wird als **Statisch** bezeichnet.

Prozess:

ist die Ausführung eines Programms.

Der Prozess wird als **Dynamisch** bezeichnet.

Kontrollstrukturen sind selbst Anweisungen!

Merkmale Prozesse

Prozesse können entweder "endlich" oder "unendlich" sein.

Endliche Prozesse: Terminieren.
Unendliche Prozesse: Terminieren nicht.

Anweisungen

Anweisungen sind üblicherweise Zuweisungen o. Kontrollanweisungen (Schleifen und bedingte Anweisungen) und Prozeduren (Funktions und Methodenaufrufe).

Im Gegensatz zu Ausdrücken haben Anweisungen **nicht immer** einen Wert.

1. Anweisungen können Ausdrücke sein, z.B. Zuweisungen, Methodenaufrufe oder Inkrement/Dekrement operatoren.
2. Vergleiche oder Arithmetische Ausdrücke sind syntaktisch keine vollständigen Anweisungen.

Prinzip der Lokalität und Linearität

1. **Lokalität:**

   (a) Der Kontrollfluss verlässt den durch den **Eingang** und **Ausgang** definierten Kontrollbereich, nicht!

2. **Linearität:**

   (a) Betrachtet man, jede Kontrollstruktur *makroskopisch*, dann verläuft der Kontrollfluss **linear** durch einen Algorithmus, *d.h.* streng Sequentiell von Anfang bis Ende.

Jede Kontrollstruktur für sich gesehen bildet eine Einheit und werden Schrittweise ausgeführt. Aus einer Kontrollstruktur **kann nicht herausgesprungen** werden. Sie folgen dem Prinzip der Lokalität. Zudem lassen sich Kontrollstrukturen selbst, beliebig kombinieren und verschachteln, daher werden Sie zur beschreibung von Komplexen Algorithmen verwendet.

Ein Beispiel

Beispiel

```java
// Importieren einer Klasse
import java.util.Scanner;

/** public = Von oeffentlich aufrufbar.
    class  = definiert eine Klasse. */
public class Main {

    // Globale, nur innerhalb dieser Klasse veraenderbare Variable
    // private = nur innerhalb dieser Klasse verwendbar.
    // Variable wird Deklariert & Initialisiert!
    private int globInt = 42;

    // Erstellung der Main Methode, der Einstieg ins Programm.
    public static void main(String[] args){
        // Initialisierung eines Scanners.
        Scanner scIn = new Scanner(System.in);

        // Lokale Variable wird deklariert.
        String rawInput;

        // Lokale Variable wird initialisiert.
        // scIn.next() liest input aus der Konsole ein.
        rawInput = scIn.next();

        // Benutzen einer Anweisung/ Kontrollstruktur
        // Wenn rawInput Nicht "" ist, dann fuehre aus.
        if (rawInput != "") {
            System.out.println(rawInput + globInt);
        }
    }
}
```