JAVA BASICS

2. EINFÜHRUNG IN DIE SPRACHE

Agenda

- Variablen, Datentypen
- Arrays
- Steuerstrukturen (Bedingungen, switch, Schleifen)
- Operatoren
- Methoden
- Klassen

Variablen

- Eine Variable:
  - hält einen Wert
  - hat einen Datentyp
  - wird in einer Deklaration erstellt
- Namenskonvention:
  - beginnt mit einem Kleinbuchstaben
  - Verwendung von CamelCase

Motivation

- Welche Erfahrungen haben Sie mit anderen Programmiersprachen? Welche Datentypen kennen Sie?
- Was ist der Unterschied zwischen Java und Javascript hinsichtlich Datentypen?
- Was sollte Sie bei Berechnungen mit großen Ganzzahlen beachten?
- Was sollte bei Berechnungen mit Dezimalzahlen (z. B. Geld) beachtet werden?

Primitive Datentypen in Java

```
type variableName = value;
```

- **byte**: Ganzzahlen von -128 bis 127
- **short**: Ganzzahlen von -32,768 bis 32,767
- **int**: Ganzzahlen von -2,147,483,648 bis 2,147,483,647
- **long**: Ganzzahlen von -9,223,372,036,854,775,808 bis 9,223,372,036,854,775,807
- **float**: Gleitkommazahlen. Geeignet für 6-7 Dezimalstellen
- **double**: Gleitkommazahlen. Geeignet für 15-16 Dezimalstellen
- **boolean**: Wahrheitswerte (true/false)
- **char**: Ein einzelnes Zeichen oder ASCII-Werte

Nicht-primitve, vordefinierte Datentypen

- **String**: Speichert Text
- **BigDecimal**: Speichert Dezimalzahlen mit definierter Genauigkeit
- **BigInteger**

Beispiel:

```
int myNum = 5;               // Ganzzahl
float myFloatNum = 5.99f;    // Gleitkommazahl
char myLetter = 'D';         // Zeichen
boolean myBool = true;       // Boolean
String myText = "Hallo";     // String

String x = "10";
int y = 20;
String z = x + y;  // z wird "1020" (als String)
```

Arrays

Arrays können eine oder mehrere Dimensionen haben:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
int[][] coordinates = {{1, 2}, {2, 3}};
```

> Weitere Informationen finden Sie hier.

Operatoren

Für eine detaillierte Liste der Operatoren besuchen Sie w3schools - Java Operatoren.

Arithmetische Operatoren:

- `+` Addition: Addiert zwei Werte `x + y`
- `-` Subtraktion: Subtrahiert einen Wert vom anderen `x - y`
- `*` Multiplikation: Multipliziert zwei Werte `x * y`
- `/` Division: Teilt einen Wert durch einen anderen `x / y`
- `%` Modulo: Gibt den Rest der Division zurück `x % y`
- `++` Inkrement: Erhöht den Wert einer Variablen um 1 `++x`
- `--` Dekrement: Verringert den Wert einer Variablen um 1 `--x`

Zuweisungsoperatoren

Operator	Beispiel	Gleichwertig
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3

Vergleichsoperatoren

Operator	Beispiel	Gleichwertig
==	Gleich	x == y
!=	Ungleich	x != y
>	Größer als	x > y
<	Kleiner als	x < y
>=	Größer oder gleich	x >= y
<=	Kleiner oder gleich	x <= y

Logische Operatoren

- **&&** Logisches UND: Gibt `true` zurück, wenn beide Ausdrücke wahr sind `x < 5 && x < 10`
- **||** Logisches ODER: Gibt `true` zurück, wenn einer der Ausdrücke wahr ist `x < 5 || x < 4`
- **!** Logisches NICHT: Kehrt das Ergebnis um, gibt `false` zurück, wenn der Ausdruck wahr ist `!(x < 5 && x < 10)`

Steuerfluss - Bedingung

```
if (Bedingung1) {
  // Codeblock, wenn Bedingung1 wahr ist
} else if (Bedingung2) {
  // Codeblock, wenn Bedingung1 falsch und Bedingung2 wahr ist
} else {
  // Codeblock, wenn sowohl Bedingung1 als auch Bedingung2 falsch sind
}
```

Steuerfluss - Switch

```
switch(Ausdruck) {
  case x:
    // Codeblock
    break;
  case y:
    // Codeblock
    break;
  default:
    // Codeblock
}
```

Steuerfluss - Schleifen (For-each Beispiel)

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

Methoden, Parameter und Strukturierung des Codes

- Methoden:
  - sind Codeblöcke, die nur ausgeführt werden, wenn sie aufgerufen werden
  - verwenden Parameter, um Daten in die Methode zu übergeben
  - können einen Rückgabewert oder `void` haben
  - werden in einer Klasse oder Struktur deklariert
- Verwenden Sie Methoden, um Code wiederzuverwenden

Beispiel:

```
class SimpleMathExtension {
    public int divideTwoNumbers(int number1, int number2) {
        return number1 / number2;
    }
}
```

Main Methode

Die Entry-Methode, die nach dem Starten der App aufgerufen wird:

```
class SimpleMathExtension {
    public static void Main(String[] args) {
      int result = divideTwoNumbers(9, 3);
      System.out.println(result);
    }

    public int divideTwoNumbers(int number1, int number2) {
        return number1 / number2;
    }
}
```

Method Overloading

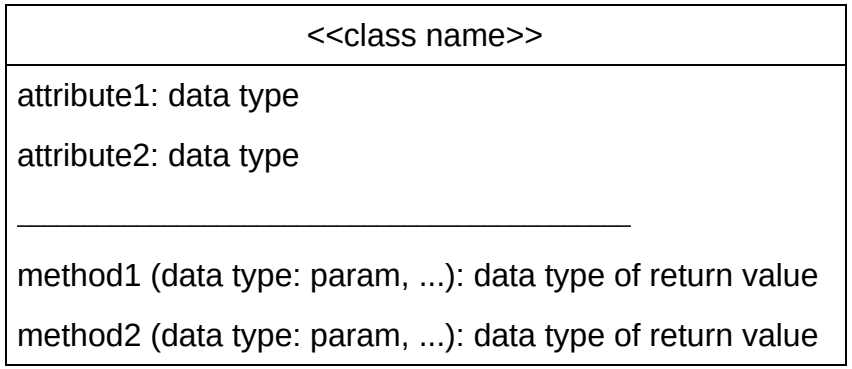
```
/**
* Methoden haben denselben Namen, aber unterschiedliche Parameter
**/
class MethodOverloadExample {
    public int addNumbers(int number1, int number2) {
        return number1 + number2;
    }

    public int addNumbers(int number1, int number2, int number3) {
        return number1 + number2 + number3;
    }

    public double addNumbers(double number1, double number2) {
        return number1 + number2;
    }
}
```

Lokale Variablen

```
public int divideTwoNumbers(int number1, int number2) {
    // Lokale Variable, nur innerhalb der Methode verfügbar
    int returnValue = number1 / number2;
    return returnValue;
}
```

2. EINFÜHRUNG IN DIE SPRACHE

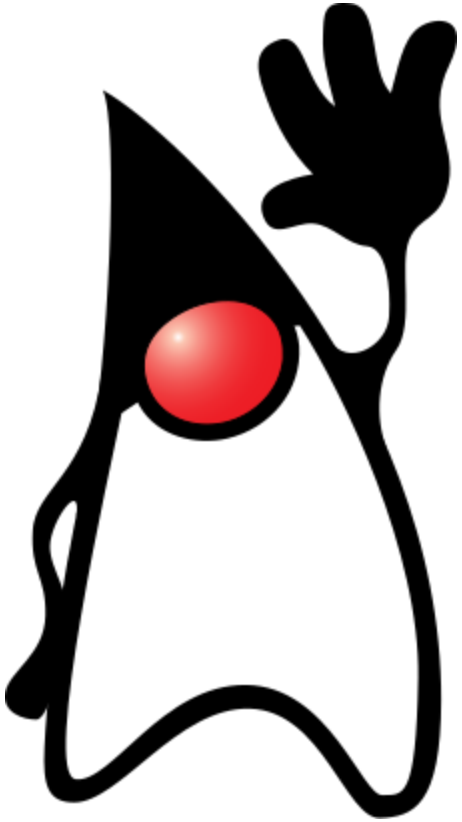
Klassendefinition



Quiz

- Nenne jeweils einen Datentyp für: ganze Zahlen, Dezimalzahlen, Text, Zeichen.
- Nenne arithmetische Operatoren. Welche Arten von Operatoren gibt es noch?
- Wie heißt die Methode, die beim Starten eines Projektes ausgeführt wird?
- Was sind lokale Variablen?
- Aus welchen drei Elementen besteht die Signatur einer Methode?
- Wie kann eine Klasse verwendet werden? Was muss man dafür tun?

Weitere Aufgaben findest du in der README.



```
This is the raw source code for the README file. You can copy and paste it into a `README.md` file and use it as needed. Let me know if you need further changes or adjustments!
```