Java Programmierung Basis:

* **Lexikalische Elemente**:
  + Java-Programme bestehen aus Unicode-Zeichen, die eine Vielzahl von Symbolen ermöglichen.
  + Kommentare in Java können einzeilig, mehrzeilig oder Dokumentationskommentare sein, jede mit spezifischer Syntax.
* **Bezeichner**:
  + Bezeichner sind Namen für Variablen, Klassen oder Methoden, die bestimmten Namenskonventionen folgen.
* **Variablendeklaration**:
  + Die Syntax zur Deklaration von Variablen umfasst die Angabe des Typs und des Namens.

Kontrollstrukturen

* If-Anweisungen:
  + Die if-Anweisung bewertet eine Bedingung und führt einen Codeblock aus, wenn diese wahr ist.
  + Der else-Zweig bietet einen alternativen Ausführungspfad.
* public class IfExample {
* 2 public static void main(String[] args) {
* 3 int a = 10;
* 4 int b = 20;
* 5
* 6 // Beispiel einer if-Anweisung
* 7 if (a < b) {
* 8 System.out.println("a ist kleiner als b");
* 9 }
* 10
* 11 // Beispiel einer if-else-Anweisung
* 12 if (a > b) {
* 13 System.out.println("a ist größer als b");
* 14 } else {
* 15 System.out.println("a ist nicht größer als b");
* 16 }
* 17
* 18 // Beispiel einer if-else if-else-Anweisung
* 19 if (a < b) {
* 20 System.out.println("a ist kleiner als b");
* 21 } else if (a == b) {
* 22 System.out.println("a ist gleich b");
* 23 } else {
* 24 System.out.println("a ist größer als b");
* 25 }
* 26 }

Switch-Cases:

* **switch-Anweisung**: Eine Mehrfachverzweigung, die Code basierend auf dem Wert einer Variablen ausführt.

int tag = 3;

2

3switch (tag) {

4 case 1:

5 System.out.println("Montag");

6 break;

7 case 2:

8 System.out.println("Dienstag");

9 break;

10 case 3:

11 System.out.println("Mittwoch");

12 break;

13 default:

14 System.out.println("Ungültiger Tag");

15}

27}

Datenrepräsentation

* Alle Datentypen in Java haben eine feste Größe, die in Bits dargestellt wird.
* Java unterstützt vier ganzzahlige Datentypen (byte, short, int, long), die alle vorzeichenbehaftet sind.
* Fließkommatypen umfassen float (einfache Genauigkeit) und double (doppelte Genauigkeit).
* int alter = 25; // Ganzzahlige Variable
* 5 double gehalt = 3500.50; // Fließkommazahl
* 6 String name = "Max Mustermann"; // Zeichenkette
* 7 boolean istStudent = false; // Boolesche Variable
* final double PI = 3.14159; // Konstante für den Wert von Pi

A screenshot of a computer

AI-generated content may be incorrect.

Variablen und Konstanten

* Variablen sind durch einen Namen, eine Adresse und einen Wert definiert und können während der Programmausführung geändert werden.
* Konstanten werden mit dem Schlüsselwort final deklariert, was ihre Werte unveränderlich macht.

Operatoren

* Java umfasst verschiedene Operatoren für Zuweisungen, logische Operationen, relationale Vergleiche und arithmetische Berechnungen.

public class OperatorExample {

2 public static void main(String[] args) {

3 // Arithmetische Operatoren

4 int a = 10;

5 int b = 5;

6

7 int summe = a + b; // Addition

8 int differenz = a - b; // Subtraktion

9 int produkt = a \* b; // Multiplikation

10 double quotient = (double) a / b; // Division

11 int rest = a % b; // Modulo

12

13 System.out.println("Addition: " + summe);

14 System.out.println("Subtraktion: " + differenz);

15 System.out.println("Multiplikation: " + produkt);

16 System.out.println("Division: " + quotient);

17 System.out.println("Modulo: " + rest);

18

19 // Relationale Operatoren

20 boolean istGleich = (a == b); // Gleichheit

21 boolean istUngleich = (a != b); // Ungleichheit

22 boolean istGroesser = (a > b); // Größer

23 boolean istKleiner = (a < b); // Kleiner

24 boolean istGroesserGleich = (a >= b); // Größer oder gleich

25 boolean istKleinerGleich = (a <= b); // Kleiner oder gleich

26

27 System.out.println("Ist a gleich b? " + istGleich);

28 System.out.println("Ist a ungleich b? " + istUngleich);

29 System.out.println("Ist a größer als b? " + istGroesser);

30 System.out.println("Ist a kleiner als b? " + istKleiner);

31 System.out.println("Ist a größer oder gleich b? " + istGroesserGleich);

32 System.out.println("Ist a kleiner oder gleich b? " + istKleinerGleich);

33

34 // Logische Operatoren

35 boolean c = true;

36 boolean d = false;

37

38 boolean und = c && d; // Logisches UND

39 boolean oder = c || d; // Logisches ODER

40 boolean nicht = !c; // Logisches NICHT

41

42 System.out.println("c UND d: " + und);

43 System.out.println("c ODER d: " + oder);

44 System.out.println("NICHT c: " + nicht);

45 }

* 46}

Typumwandlung

* Typumwandlung ist die Konvertierung eines Datentyps in einen anderen, die entweder erweiternd (sicher) oder einschränkend (potenziell unsicher) sein kann.

public class TypeCastingExample {

2 public static void main(String[] args) {

3 // Erweiternde Typumwandlung (Widening)

4 int intValue = 100;

5 double doubleValue = intValue; // int wird automatisch in double umgewandelt

6

7 System.out.println("Ursprünglicher int-Wert: " + intValue);

8 System.out.println("Erweiterte double-Wert: " + doubleValue);

9

10 // Verengende Typumwandlung (Narrowing)

11 double anotherDoubleValue = 9.78;

12 int anotherIntValue = (int) anotherDoubleValue; // double muss explizit in int umgewandelt werden

13

14 System.out.println("Ursprünglicher double-Wert: " + anotherDoubleValue);

15 System.out.println("Verengte int-Wert: " + anotherIntValue); // Nachkommastellen werden abgeschnitten

16

17 // Typumwandlung zwischen char und int

18 char charValue = 'A';

19 int charToInt = (int) charValue; // char wird in int umgewandelt

20

21 System.out.println("Ursprünglicher char-Wert: " + charValue);

22 System.out.println("char in int umgewandelter Wert: " + charToInt); // ASCII-Wert von 'A' ist 65

23

24 // Typumwandlung zwischen int und char

25 int intFromChar = 66; // ASCII-Wert für 'B'

26 char intToChar = (char) intFromChar; // int wird in char umgewandelt

27

28 System.out.println("Ursprünglicher int-Wert: " + intFromChar);

29 System.out.println("int in char umgewandelter Wert: " + intToChar); // 'B'

30 }

31}

Schleifen in Java

* while-Schleife:
  + Vorab-Test-Schleife, die so lange fortgesetzt wird, wie eine Bedingung wahr ist.
* do-while-Schleife:
  + Nach-Test-Schleife, die mindestens einmal ausgeführt wird, bevor die Bedingung überprüft wird.
* for-Schleife:
  + Kombiniert Initialisierung, Bedingungsprüfung und Aktualisierung in einer Zeile.
  + Ermöglicht eine prägnante Iteration über einen Wertebereich.
* Kontrollflussanweisungen:
  + break: Beendet die Schleife sofort.
  + continue: Überspringt die aktuelle Iteration und fährt mit der nächsten fort.

While-Schleife

int i = 0;

2

3while (i < 5) {

4 System.out.println("i ist: " + i);

5 i++; // Erhöht i um 1

6}

Do-while-Schleife:

int j = 0;

2

3do {

4 System.out.println("j ist: " + j);

5 j++; // Erhöht j um 1

6} while (j < 5);

For-Schleife:

for (int k = 0; k < 5; k++) {

2 System.out.println("k ist: " + k);

3}

Verschachtelte Schleifen:

for (int m = 0; m < 3; m++) {

2 for (int n = 0; n < 2; n++) {

3 System.out.println("m ist: " + m + ", n ist: " + n);

4 }

5}

5. Verwendung von break und continue

Die break-Anweisung wird verwendet, um eine Schleife vorzeitig zu beenden.

for (int p = 0; p < 10; p++) {

2 if (p == 5) {

3 break; // Beendet die Schleife, wenn p gleich 5 ist

4 }

5 System.out.println("p ist: " + p);

6}

Die continue-Anweisung wird verwendet, um die aktuelle Iteration zu überspringen und mit der nächsten fortzufahren.

for (int q = 0; q < 10; q++) {

2 if (q % 2 == 0) {

3 continue; // Überspringt die aktuelle Iteration, wenn q gerade ist

4 }

5 System.out.println("q ist: " + q);

6}

Datenstrukturen in Java:

* **Arrays**:
  + Java unterstützt eindimensionale und mehrdimensionale Arrays.
  + Arrays sind dynamisch zugewiesen, d.h. ihre Größe wird zur Laufzeit bestimmt.
  + Syntax zur Deklaration von Arrays umfasst die Verwendung von eckigen Klammern (z.B. int[] a;).
  + Arrays können mit dem new-Operator oder durch literale Initialisierung initialisiert werden.
* // Deklaration eines eindimensionalen Arrays
* 2int[] zahlen = new int[5]; // Array mit 5 Elementen
* 3
* 4// Initialisierung der Array-Elemente
* 5zahlen[0] = 10;
* 6zahlen[1] = 20;
* 7zahlen[2] = 30;
* 8zahlen[3] = 40;
* 9zahlen[4] = 50;
* 10
* 11// Ausgabe der Array-Elemente
* 12for (int i = 0; i < zahlen.length; i++) {
* 13 System.out.println("Element " + i + ": " + zahlen[i]);
* 14}
* // Deklaration und Initialisierung in einer Zeile
* 2String[] farben = {"Rot", "Grün", "Blau"};
* 3
* 4// Ausgabe der Array-Elemente
* 5for (String farbe : farben) {
* 6 System.out.println("Farbe: " + farbe);
* 7}
* **Zugriff auf Array-Elemente**:
  + Elemente werden über ihren Index angesprochen, beginnend bei 0.
  + Die Länge eines Arrays kann mit der length-Eigenschaft abgerufen werden.
* int[] zahlen = {10, 20, 30, 40, 50};
* 2
* 3// Zugriff auf ein bestimmtes Element
* 4int drittesElement = zahlen[2]; // Index 2 entspricht dem dritten Element
* 5System.out.println("Das dritte Element ist: " + drittesElement);

**Mehrdimensionale Arrays**:

* + Werden mit mehreren Paaren von eckigen Klammern deklariert (z.B. int[][] a;).
  + Können nicht-rechteckige Datenstrukturen darstellen.
* // Deklaration eines zweidimensionalen Arrays
* 2int[][] matrix = new int[2][3]; // 2 Zeilen und 3 Spalten
* 3
* 4// Initialisierung der Array-Elemente
* 5matrix[0][0] = 1;
* 6matrix[0][1] = 2;
* 7matrix[0][2] = 3;
* 8matrix[1][0] = 4;
* 9matrix[1][1] = 5;
* 10matrix[1][2] = 6;
* 11
* 12// Ausgabe der Array-Elemente
* 13for (int i = 0; i < matrix.length; i++) {
* 14 for (int j = 0; j < matrix[i].length; j++) {
* 15 System.out.print(matrix[i][j] + " ");
* 16 }
* 17 System.out.println(); // Neue Zeile nach jeder Zeile der Matrix
* 18}
* Schlüsselkonzepte:
  + Klasse: Eine Vorlage zur Erstellung von Objekten mit gemeinsamen Eigenschaften.
  + Objekt: Eine konkrete Instanz einer Klasse.
  + Attribut: Daten, die den Zustand eines Objekts beschreiben.
  + Operation: Eine Funktion, die das Verhalten eines Objekts definiert.
* Beziehung zwischen Klasse und Objekt:
  + Klassen dienen als Vorlagen zur Erstellung von Objekten, die ihre eigenen Attribute und Methoden haben.
  + Beispiel: Eine Klasse "Auto" kann Attribute wie Name, Jahr der Erstzulassung und Leistung haben.
* Syntax für Klassendefinition:
  + Allgemeine Struktur: [Modifikator] class Klassenname [extends Basisklasse] [implements Schnittstellen] { Attributdeklarationen; Operationsdeklarationen; }
  + Beispiel einer einfachen Klasse:

java

􀈅􀉂

1class Auto {

2 String name;

3 int jahrDerErstzulassung;

4 int leistung;

5}

* **Objekterstellung**:
  + Objekte werden mit dem new-Operator instanziiert.
  + Beispiel der Objekterstellung:

java

􀈅􀉂

1Auto meinPkw = new Auto();