

Einführung in die Programmierung mit Java - Wiederholung

Dozent: Prof. Dr. Michael Eichberg

Kontakt: michael.eichberg@dhbw.de, Raum 149B

Version: 1.1

Kontrollfragen

1 ■ Welche primitiven Datentypen kennen wir?

2 ■ Was sind Literale?

3 ■ Welche der folgenden Bezeichner sind für Variablen gültig?

1. fooBar
2. BarFoo
3. _fooBar
4. 1fooBar
5. fooBar1
6. fooBar!
7. \$fooBar
8. \$_BarFoo

4 ■ Welche der folgenden Bezeichner sollte man für eine Variable verwenden?

1. gewinn
2. Gewinn
3. _private_i
4. i
5. \$i
6. _i

5 ■ Sie definieren eine Konstante, welchen Namen würden Sie verwenden?

1. ISOLAENDERCODE
2. ISO_LÄNDERCODE
3. ISO_LAENDERCODE
4. ISO_Ländercode
5. ISO_Laendercode
6. iso_Laendercode

6 ■ Welchen Typ hat die Variable x in folgendem Code?

1. `var x = 1;`
2. `var x = 1.0;`
3. `var x = '1';`
4. `var x = 1f;`
5. `var x = 2F;`

6. `var x = "x";`

7 ■ Wieviele Bits hat ein int?

1. 8
2. 16
3. 24
4. 32
5. 40
6. 48

8 ■ Wie ist der Wertebereich von byte?

1. 0 bis 255
2. -128 bis 128
3. -128 bis 127
4. -127 bis 127
5. -127 bis 128

9 ■ Was passiert bei den folgenden Typumwandlungen?

1. `int i = 42; byte b = (byte) i;`
2. `int i = 255; byte b = (byte) i;`
3. `int i = 256; byte b = (byte) i;`

10 ■ Warum ist der folgende Ausdruck wahr obwohl dieser mathematisch falsch ist?

`(long) ((float) (Long.MAX_VALUE - Integer.MAX_VALUE)) == Long.MAX_VALUE;`

11 ■ Ist die Länge eines Strings gleich der Anzahl *sichtbarer* Zeichen?

12 ■ Sie möchten in einem String ein Anführungszeichen verwenden. Wie machen Sie das?

13 ■ Muss ich bei der Variablendeklaration den Typ explizit angeben?

14 ■ Wie deklariert man eine Konstante? Sollte man Werte, die man nicht ändern möchte immer als Konstanten deklarieren?

15 ■ Wie ist der Operator für die Modularechnung (d. h. Restwertberechnung) in Java?

16 ■ Wie sieht der ternäre Operator in Java aus?

17 ■ Welchen Wert haben die folgenden Ausdrücke, wenn x vor der jeweiligen

Auswertung den Wert 5 hat?

1. `x++`
2. `++x`
3. `x += 1`
4. `x = (x = x - 2) + 3 * 4`
5. `x = x = x - 2 + 3 * 4`
6. `(x = (x = x - 2) + 3) * 4`
7. `x >= 5 || 2 / (x - 5) == 0`
8. `x >= 5 | 2 / (x - 5) == 0`
9. `x << 1`
10. `x << 1 >> 2`

18 ■ Was stellt ein Block in Hinblick auf eine Variable dar?

19 ■ Eine while-Schleife und eine for-Schleife können immer ineinander umgewandelt werden?

20 ■ In welcher Weise unterscheidet sich eine do-while-Schleife von einer while-Schleife?

21 ■ Schleifen und Variablen - wie ist die Ausgabe auf der *JShell*?

```
int i = 0;
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break;
    }
}
System.out.println(i);
```

22 ■ Schleife mit break - wie ist die Ausgabe?

```
int i = 0;
for (; i < 10; i++) {
    if (i == 5) {
        break;
    }
}
System.out.println(i);
```

23 ■ Ganz einfache Schleife - wie ist die Ausgabe?

```
int i = 10;
for (; i < 10; i++) {
    System.out.println(i);
}
```

```
}
```

24 ■ Schleife mit continue - wie ist die Ausgabe?

```
int i = 0;
for (; i < 10; i++) {
    if (i % 2 == 0) {
        continue;
    }
    System.out.println(i);
}
```

25 ■ Verschachteltet Schleifen - wie ist die Ausgabe?

```
int i = 0;
outer : for (; i < 10; i++) {
    if (i % 2 == 0) {
        continue;
    }
    System.out.println(i);
    for (int j = 1; j < 10; j++) {
        if (j % 3 == 0) {
            continue outer;
        }
        System.out.println(i + " " + j);
    }
}
System.out.println(i);
```

26 ■ Verschachteltet Schleifen - wie ist die Ausgabe?

```
outer : for (int i = 0; i < 10; i++) {
    if (i % 2 == 0) {
        i = 10;
        continue outer;
    }
    System.out.println(i);
    for (int j = 1; j < 10; j++) {
        if (j % 3 != i % 5) {
            break;
        }
        System.out.println(i + " " + j);
    }
}
```

27 ■ Rekursive Funktion

- Was berechnet diese Funktion?
- Ist diese Funktion effizient?

- Ist eine Lösung mit for-Schleife besser?

```
int f(int n) {  
    if (n == 0) return 0; return n + f(n-1);  
}
```

28 ■ Funktion mit "Tail-Call"

- Was berechnet diese Funktion?
- Ist diese Funktion effizient(er)?

```
/* private */ int f(int n, int sum) {  
    if (n == 0) return sum; return f(n-1, n+sum);  
}  
int f(int n) { return f(n, 0); }
```

29 ■ Wie werden Parameter übergeben?

30 ■ Wie bewerten Sie folgende Kommentierung?

```
/**  
 * Testet ob eine Zahl eine Primzahl ist.  
 *  
 * Die Laufzeit ist O(n/4).  
 *  
 * @param n Eine positive ganze Zahl.  
 * @return true, wenn n eine Primzahl ist, sonst false.  
 */  
boolean isPrim(int n) {  
    ...  
}
```

31 ■ Ist der Kommentar ausreichend?

```
/**  
 * Computes the absolute value of the argument.  
 *  
 * @param a - the argument whose absolute value is to be determined  
 * @return the absolute value of the argument.  
 */  
double abs(double a) { ... }
```

32 ■ Ist die Kommentierung hier ausreichend?

```
/**  
 * Returns the absolute value of an int value.  
 * If the argument is not negative, the argument is returned.  
 * If the argument is negative, the negation of the argument is returned.  
 *  
 * @param a - the argument whose absolute value is to be determined
```

```
* @return the absolute value of the argument.  
*/  
long abs(long a) { ... }
```

33 ■ Sind Java Assertions (:java: `assert`) in Java immer aktiv?

34 ■ Wofür sollten Assertions verwendet werden?

1. Zur Validierung von Eingabeparametern?
2. Zur Validierung von Rückgabewerten?
3. Zur Validierung von internen Invarianten?

35 ■ Beschreiben Sie die Ausgabe:

```
int width = 20;  
int height = 10;  
for (int i = 0; i < width; i++) print("-");  
println("");  
for (int i = 0; i < height - 2; i++) {  
    print("|");  
    for (int j = 0; j < width - 2; j++) print(" ");  
    println("");  
}
```

■ Welche primitiven Datentypen kennen wir?

■ Was sind Literale?

■ Welche der folgenden Bezeichner sind für Variablen gültig?

1. fooBar
2. BarFoo
3. _fooBar
4. 1fooBar
5. fooBar1
6. fooBar!
7. \$fooBar
8. \$_BarFoo

■ Welche der folgenden Bezeichner sollte man für eine Variable verwenden?

1. gewinn
2. Gewinn
3. _private_i
4. i
5. \$i
6. _i

■ Sie definieren eine Konstante, welchen Namen würden Sie verwenden?

1. ISOLAENDERCODE
2. ISO_LÄNDERCODE
3. ISO_LAENDERCODE
4. ISO_Ländercode
5. ISO_Laendercode
6. iso_Laendercode

■ Welchen Typ hat die Variable x in folgendem Code?

1. `var x = 1;`
2. `var x = 1.0;`
3. `var x = '1';`
4. `var x = 1f;`
5. `var x = 2F;`
6. `var x = "x";`

■ Wieviele Bits hat ein int?

1. 8
2. 16
3. 24
4. 32
5. 40
6. 48

■ Wie ist der Wertebereich von byte?

1. 0 bis 255
2. -128 bis 128
3. -128 bis 127
4. -127 bis 127
5. -127 bis 128

■ Was passiert bei den folgenden Typumwandlungen?

1. `int i = 42; byte b = (byte) i;`
2. `int i = 255; byte b = (byte) i;`
3. `int i = 256; byte b = (byte) i;`

■ Warum ist der folgende Ausdruck wahr obwohl dieser mathematisch falsch ist?

```
(long) ((float) (Long.MAX_VALUE - Integer.MAX_VALUE)) == Long.MAX_VALUE;
```

■ Ist die Länge eines Strings gleich der Anzahl *sichtbarer* Zeichen?

■ Sie möchten in einem String ein Anführungszeichen verwenden. Wie machen Sie das?

■ Muss ich bei der Variablendeklaration den Typ explizit angeben?

■ Wie deklariert man eine Konstante? Sollte man Werte, die man nicht ändern möchte immer als Konstanten deklarieren?

■ Wie ist der Operator für die Modularechnung (d. h. Restwertberechnung) in Java?

■ Wie sieht der ternäre Operator in Java aus?

■ Welchen Wert haben die folgenden Ausdrücke, wenn x vor der jeweiligen Auswertung den Wert 5 hat?

1. `x++`
2. `++x`
3. `x += 1`
4. `x = (x = x - 2) + 3 * 4`
5. `x = x = x - 2 + 3 * 4`
6. `(x = (x = x - 2) + 3) * 4`
7. `x >= 5 || 2 / (x - 5) == 0`
8. `x >= 5 | 2 / (x - 5) == 0`
9. `x << 1`
10. `x << 1 >> 2`

■ Was stellt ein Block in Hinblick auf eine Variable dar?

■ Eine while-Schleife und eine for-Schleife können immer ineinander umgewandelt werden?

■ In welcher Weise unterscheidet sich eine do-while-Schleife von einer while-Schleife?

■ Schleifen und Variablen - wie ist die Ausgabe auf der *JShell*?

```
int i = 0;
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break;
    }
}
System.out.println(i);
```

Schleife mit break - wie ist die Ausgabe?

```
int i = 0;
for (; i < 10; i++) {
    if (i == 5) {
        break;
    }
}
System.out.println(i);
```

■ Ganz einfache Schleife - wie ist die Ausgabe?

```
int i = 10;  
for (; i < 10; i++) {  
    System.out.println(i);  
}
```

Schleife mit continue - wie ist die Ausgabe?

```
int i = 0;
for (; i < 10; i++) {
    if (i % 2 == 0) {
        continue;
    }
    System.out.println(i);
}
```

■ Verschachtelte Schleifen - wie ist die Ausgabe?

```
int i = 0;
outer : for (; i < 10; i++) {
    if (i % 2 == 0) {
        continue;
    }
    System.out.println(i);
    for (int j = 1; j < 10; j++) {
        if (j % 3 == 0) {
            continue outer;
        }
        System.out.println(i + " " + j);
    }
}
System.out.println(i);
```


■ Verschachtelte Schleifen - wie ist die Ausgabe?

```
outer : for (int i = 0; i < 10; i++) {  
    if (i % 2 == 0) {  
        i = 10;  
        continue outer;  
    }  
    System.out.println(i);  
    for (int j = 1; j < 10; j++) {  
        if (j % 3 != i % 5) {  
            break;  
        }  
        System.out.println(i + " " + j);  
    }  
}
```

■ Rekursive Funktion

- Was berechnet diese Funktion?
- Ist diese Funktion effizient?
- Ist eine Lösung mit for-Schleife besser?

```
int f(int n) {  
    if (n == 0) return 0; return n + f(n-1);  
}
```

■ Funktion mit "Tail-Call"

- Was berechnet diese Funktion?
- Ist diese Funktion effizient(er)?

```
/* private */ int f(int n, int sum) {  
    if (n == 0) return sum; return f(n-1,n+sum);  
}  
int f(int n) { return f(n,0); }
```

■ Wie werden Parameter übergeben?

■ Wie bewerten Sie folgende Kommentierung?

```
/**
 * Testet ob eine Zahl eine Primzahl ist.
 *
 * Die Laufzeit ist  $O(n/4)$ .
 *
 * @param n Eine positive ganze Zahl.
 * @return true, wenn n eine Primzahl ist, sonst false.
 */
boolean isPrim(int n) {
    ...
}
```

Ist der Kommentar ausreichend?

```
/**
 * Computes the absolute value of the argument.
 *
 * @param a - the argument whose absolute value is to be determined
 * @return the absolute value of the argument.
 */
double abs(double a) { ... }
```

Ist die Kommentierung hier ausreichend?

```
/**
 * Returns the absolute value of an int value.
 * If the argument is not negative, the argument is returned.
 * If the argument is negative, the negation of the argument is returned.
 *
 * @param a – the argument whose absolute value is to be determined
 * @return the absolute value of the argument.
 */
long abs(long a) { ... }
```

■ Sind Java Assertions (:java: `assert`) in Java immer aktiv?

■ Wofür sollten Assertions verwendet werden?

1. Zur Validierung von Eingabeparametern?
2. Zur Validierung von Rückgabewerten?
3. Zur Validierung von internen Invarianten?

■ Beschreiben Sie die Ausgabe:

```
int width = 20;
int height = 10;
for (int i = 0; i < width; i++) print("-");
println("");
for (int i = 0; i < height - 2; i++) {
    print("|");
    for (int j = 0; j < width - 2; j++) print(" ");
    println("|");
}
for (int i = 0; i < width; i++) {
    print("-");
}
println("");
```