Einführung in die Programmierung mit Java - Wiederholung

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de, Raum 149B

Version: 1.2

1. Grundlagen

Kontrollfragen

1.1. Welche primitiven Datentypen kennen wir?

1.2. Was sind Literale?

1.3. Welche der folgenden Bezeichner sind für Variablen gültig?

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

1.4. Welche der folgenden Bezeichner sollte man verwenden?

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

1.5. Welchen Namen würden Sie für eine Konstante verwenden?

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

1.6. 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";`

1.7. Wie viele Bits hat ein int?

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

1.8. 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

1.9. 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;`

1.10. Warum ist der folgende Ausdruck wahr?

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

Bemerkung

Rein mathematisch betrachtet - d. h. ohne Betrachtung von Typen und Typumwandlungen - wäre dieser natürlich falsch.

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

1.12. Wie fügen Sie in einen String ein Anführungszeichen ein?

1.13. Muss ich bei der Variablendeklaration den Typ explizit angeben?

1.14. Wie deklariert man eine Konstante?

Sollte man Werte, die man nicht ändern möchte immer als Konstanten deklarieren?

1.15. Wie ist der Operator für die Modulorechnung in Java?

(D. h. wenn Sie eine Restwertberechnung in Java durchführen wollen.)

1.16. Wie sieht der ternäre Operator in Java aus?

1.17. Welchen Wert haben die folgenden Ausdrücke?

`x` hat vor der jeweiligen Auswertung den Wert 5.

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 >> 2`

1.18. Was stellt ein Block in Hinblick auf eine Variable dar?

2. Schleifen

Kontrollfragen

2.1. Können `while`- und `for`-Schleifen ineinander umgewandelt werden?

2.2. Wie unterscheidet sich eine `do-while`- von einer `while`-Schleife?

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

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

2.4. Schleife mit break - wie ist die Ausgabe?

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

2.5. Ganz einfache Schleife - wie ist die Ausgabe?

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

2.6. Schleife mit continue - wie ist die Ausgabe?

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

2.7. Verschachteltet Schleifen - wie ist die Ausgabe?

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

```
    }
}
IO.println(i);
```

2.8. Verschachteltet Schleifen - wie ist die Ausgabe?

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

3. Funktionen

Kontrollfragen

3.1. Rekursive Funktion

```
1 int f(int n) {
2    if (n == 0) return 0; return n + f(n-1);
3 }
```

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

3.2. Funktion mit „Tail-Call“

```
1 /* private */ int f(int n, int sum) {
2    if (n == 0) return sum; return f(n-1,n+sum);
3 }
4 int f(int n) { return f(n,0); }
```

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

3.3. Wie werden Parameter übergeben?

3.4. Wie bewerten Sie folgende Kommentierung?

```
1  /**
2   * Testet ob eine Zahl eine Primzahl ist.
3   *
4   * Die Laufzeit ist O(n/4).
5   *
6   * @param n Eine positive ganze Zahl.
7   * @return true, wenn n eine Primzahl ist, sonst false.
8   */
9  boolean isPrim(int n) {
10    ...
11 }
```

3.5. Ist der Kommentar ausreichend?

```
1 /**
2  * Computes the absolute value of the argument.
3  *
4  * @param a - the argument whose absolute value is to be
5  *            determined
6  * @return the absolute value of the argument.
7  */
8 double abs(double a) { ... }
```

3.6. Ist die Kommentierung hier ausreichend?

```
1 /**
```

```
 * 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) { ... }
```

3.7. Sind Java Assertions (assert) in Java immer aktiv?

3.8. Wofür sollten Assertions verwendet werden?

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

3.9. Beschreiben Sie die Ausgabe des Programms

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