

Übungsstunde 3

Einführung in die Programmierung

Probleme bei Übung 1

Lösung Übung 1, Aufgabe 4

Erstellen Sie eine Beschreibung $\langle x2ygemischt \rangle$, die als legale Symbole genau jene Wörter zulässt, in denen für jedes "X" zwei "Y" als Paar auftreten. Beispiele sind $XY Y$, $Y Y X$, $X Y Y Y Y X$.

Interpretation: ...X...YY...



...YY...X...



$n \times X$ und $n \times YY$
(auch $n=0$)

$\langle g \rangle \leq [\langle g \rangle X \langle g \rangle YY \langle g \rangle \mid \langle g \rangle YY \langle g \rangle X \langle g \rangle]$

$\langle x2ygemischt \rangle \leq \langle g \rangle$

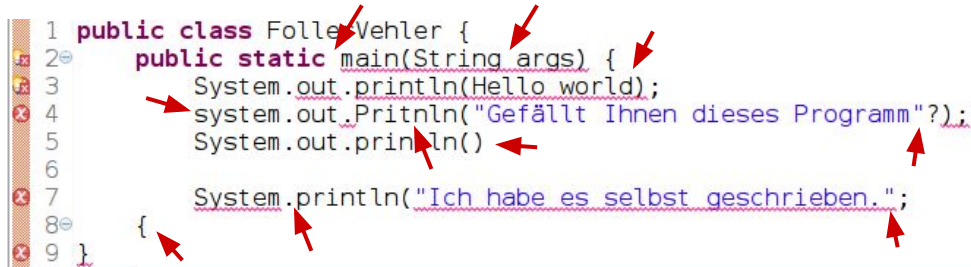
- EBNF ist case sensitiv.
- Das leere Wort häufig auch erlaubt.
- Bei der Rekursion aufpassen, dass es auch eine Option zum Ausstieg / Beenden gibt.

Eclipse Git

Nachbesprechung Übung 2

Aufgabe 1: Fehlerbehebung

```
1 public class FolleVehler {  
2     public static main(String args) {  
3         System.out.println(Hello world);  
4         system.out.Pritnln("Gefällt Ihnen dieses Programm?");  
5         System.out.println()  
6  
7         System.println("Ich habe es selbst geschrieben.");  
8     }  
9 }
```



```
public class OhneFehler {  
    public static void main(String[] args) {  
        System.out.println("Hello world");  
        System.out.println("Gefällt Ihnen dieses Programm?");  
        System.out.println();  
  
        System.out.println("Ich habe es selbst geschrieben.");  
    }  
}
```

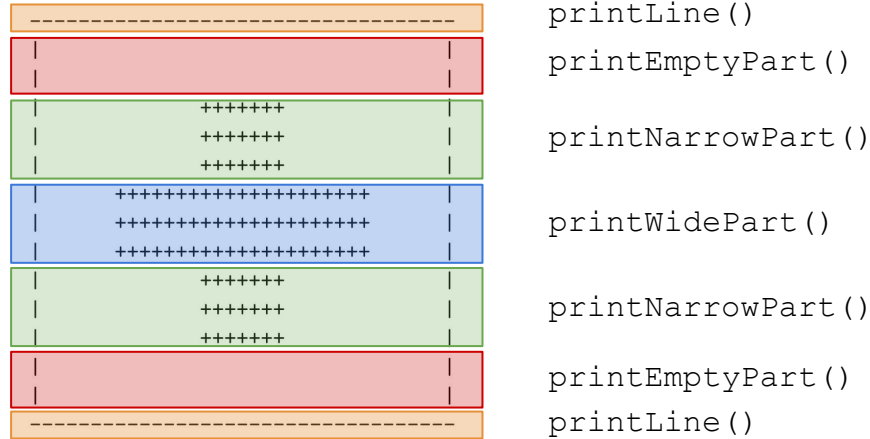
Aufgabe 2: Vorwärts- und Rückwärtsschrägstriche

Ergänzen Sie das Programm in “ForwardAndBackward.java” so, dass folgendes Muster als Text ausgegeben wird:

```
  \/  
  \\  
 \\\/  
 \\\///  
 ///\\\  
  ///\  
   /\
```

Das Muster muss im Ausgabefenster nicht zentriert sein, aber die Form sollte in sich so aussehen.

Aufgabe 3: Schweizerfahne



Teilen Sie das Programm in mehrere Methoden auf, welche von der `main`-Methode aufgerufen werden. Damit sorgen Sie dafür, dass weniger Wiederholungen von Code-Stücken vorkommen, was das Ändern des Programms deutlich einfacher macht.

Aufgabe 4: Berechnungen

1. Vervollständigen Sie "SharedDigit.java". In der Main-Methode sind zwei int Variablen a und b deklariert und mit einem Wert zwischen 10 und 99 (einschliesslich) initialisiert. Das Programm soll einer int Variablen r einen bestimmten Wert zuweisen. Wenn a und b eine Ziffer gemeinsam haben, dann wird r die gemeinsame Ziffer zugewiesen (wenn a und b beide Ziffern gemeinsam haben, dann kann eine beliebige Ziffer zugewiesen werden). Wenn es keine gemeinsame Ziffer gibt, dann soll -1 zugewiesen. Sie brauchen für dieses Programm keine Schleife.

Beispiele:

- Wenn a: 34 und b: 53, dann ist r: 3
- Wenn a: 10 und b: 22, dann ist r: -1
- Wenn a: 66 und b: 66, dann ist r: 6
- Wenn a: 34 und b: 34, dann ist r: 3 oder 4

Testen Sie Ihre Loesung mit a gleich 34 und b gleich 43. Was liefert Ihr Programm?

2. Vervollständigen Sie "SumPattern.java". In der Main-Methode sind drei int Variablen a, b, und c deklariert und mit irgendwelchen Werten initialisiert. Wenn die Summe von zwei der Variablen die dritte ergibt, nehmen wir an dass $a + c == b$, so soll die Methode "Moeglich. $a + c == b$ " ausgeben (wobei die Werte für a, b, und c einzusetzen sind). Wenn das nicht der Fall ist, dann soll die Methode "Unmoeglich." ausgeben.

Beispiele:

- Wenn a: 4, b: 10, c: 6, dann wird "Moeglich. $4 + 6 == 10$ " oder "Moeglich. $6 + 4 == 10$ " ausgegeben.
- Wenn a: 2, b: 12, c: 0, dann wird "Unmoeglich." ausgegeben.

3. Vervollständigen Sie "AbsoluteMax.java". In der Main-Methode sind drei int Variablen a, b, und c deklariert und mit irgendwelchen Werten initialisiert. Das Programm soll einer int Variable r den grössten absoluten Wert von a, b, und c zuweisen.

Aufgabe 5: Postconditions

Geben Sie für die folgenden Programmsegmente die Postcondition an, welche nach der Ausführung gilt, wenn zuvor die angegebene Precondition gilt. Verwenden Sie Java Syntax. Alle Anweisungen sind Teil einer Java Methode. Alle Variablen sind vom Typ `int` und es gibt keinen Overflow.

1.

P: { `x > 10` }

S: `y = x + 5;`

Q: { `x > 10 && y > 15` }

2.

P: { `x == 2` }

S: `y = x + 5;`

Q: { `x == 2 && y == 7` }

3.

P: { `x > 0 && z > 0` }

S: `y = z * x`

Q: { `x > 0 && z > 0 && y > 0` }

Theorie

Übersicht über den Vorlesungsstoff

■ 2.4 Verzweigungen

■ 2.4.1 «if»-Anweisungen

Vergleichsoperatoren

■ 2.4.2 Typ boolean

■ 2.4.3 Bedingte («short-circuit») Auswertung

■ 2.4.4 Pre- und Postconditions

■ 2.4.5 «Schwächste» Vorbedingung

Übersicht über den Vorlesungsstoff

- **2.5 Schleifen (Loops)**

- 2.5.1 «for» Loops
- 2.5.2 Verschachtelte Schleifen
- 2.5.3 «while» Loops

- **2.6 Methoden**

- **2.7 Strings**

- **2.8 Nochmals Schleifen**

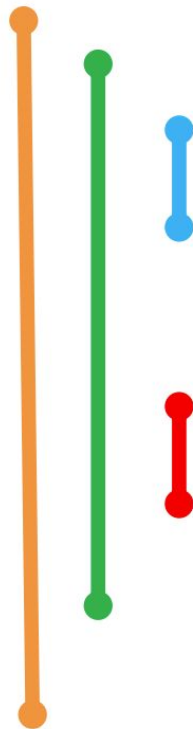
Post und Preincrement

Value Semantics

Scope

Scope (Sichtbarkeitsbereich)

```
public static void example(int x) {  
    for (int i = 1; i <= 10; i++) {  
        for (int j = i; j<=10; j++) {  
            System.out.print(x + i + j + " ");  
        } // j no longer exists here  
        System.out.println(i);  
        if (i%2==0) {  
            int j = 4;  
            System.out.print(i*j + " ");  
        } // j no longer exists here  
    } // i no longer exists here  
    System.out.println(x);  
} // x no longer exists here
```



Strings

Scanner

```

1  import java.util.Scanner;
2
3  /*
4   * Author: Maximiliana Muster
5   * für Einfuehrung in die Programmierung I, HS 2016
6   *
7   * Dieses Programm addiert zwei ganze Zahlen miteinander.
8   */
9  public class Adder {
10
11     public static void main(String[] args) {
12         int zahl1, zahl2, summe;
13
14         // erstelle Scanner Objekt
15         Scanner console = new Scanner(System.in);
16
17         // frage nach den beiden Zahlen
18         System.out.print("Geben Sie Zahl 1 ein: ");
19         zahl1 = console.nextInt();
20
21         System.out.print("Geben Sie Zahl 2 ein: ");
22         zahl2 = console.nextInt();
23
24         // Berechne und drucke die Summe
25         summe = zahl1 + zahl2;
26         System.out.println(zahl1 + " + " + zahl2 + " = " + summe);
27     }
28 }
29

```

← auch java.util.*

← Kommentare nicht vergessen!

← Variablen werden oft alle am Anfang deklariert

← Erstellt den Scanner

← print vs. println

← Leerzeichen um Operatoren erhöht Leserlichkeit

Vorbesprechung Übung 3

Aufgabe 1: Folgen und Reihen

Schreiben Sie ein Programm “Reihe.java”, das eine Zahl N von der Konsole einliest und dann die folgende Summe berechnet:

$$\frac{1}{1^2} + \frac{1}{2^2} + \cdots + \frac{1}{N^2}$$

Beispiel: Für $N = 4$ sollte Ihr Programm ca. 1.42 ausgeben. Wie verhält sich diese Summe für grosse N ?

Aufgabe 2: Binärdarstellung

Schreiben Sie ein Programm "Binaer.java", das eine positive Zahl Z einliest und dann die Binärdarstellung druckt. (Hinweis: Finden Sie zuerst die grösste Zahl k , so dass die Zweierpotenz $K = 2^k$ kleiner als Z ist.)

Beispiel: Für $Z = 13$ sollte Ihr Programm 1101 ausgeben.

Erklärung Binärdarstellung

Aufgabe 3: Grösster gemeinsamer Teiler

Schreiben Sie ein Programm “GGT.java”, das den grössten gemeinsamen Teiler (ggT) zweier ganzer Zahlen mithilfe des Euklidischen Algorithmus berechnet. Hierbei handelt es sich um eine iterative Berechnung, die auf folgender Beobachtung basiert:

1. Wenn x grösser als y ist, dann ist — sofern sich x durch y teilen lässt — der ggT von x und y gleich y ;
2. andernfalls ist der ggT von x und y der gleiche wie der ggT von y und $x \% y$.

Üben Sie diesen Algorithmus zuerst von Hand an ein paar Beispielen und schreiben Sie dann das Java Programm.

Beispiel: Für $x = 36$ und $y = 44$:

$$\text{ggT}(\underbrace{36}_x, \underbrace{44}_y) \stackrel{2.}{=} \text{ggT}(44, \underbrace{36 \% 44}_{36}) \stackrel{2.}{=} \text{ggT}(36, \underbrace{44 \% 36}_8) \stackrel{2.}{=} \text{ggT}(8, \underbrace{36 \% 8}_4) \stackrel{1.}{=} 4$$

Erklärung Euklidischer Algorithmus

Formal description of the Euclidean algorithm

- **Input** Two positive integers, a and b .
- **Output** The greatest common divisor, g , of a and b .
- **Internal computation**
 1. If $a < b$, exchange a and b .
 2. Divide a by b and get the remainder, r . If $r=0$, report b as the GCD of a and b .
 3. Replace a by b and replace b by r . Return to the previous step.

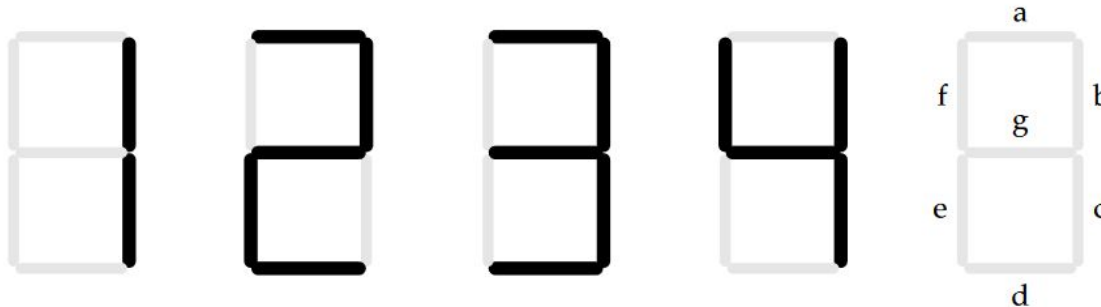
<https://sites.math.rutgers.edu/~greenfie/qs2004/euclid.html>

Aufgabe 4: Zahlenerkennung

Für diese Aufgabe verwenden wir einen String um die erleuchtenden Segmente einer **Sieben-segmentanzeige** zu kodieren. Die Segmente sind, wie im Bild gezeigt, von a bis g nummeriert. Die Kodierung einer möglichen Anzeige ist ein String, in welchem der Buchstabe 'x' genau dann vorkommt, wenn das 'x'te Segment der Anzeige erleuchtet ist. Zum Beispiel wird die Zahl 2 kodiert durch 'abged'. Zur Einfachheit darf angenommen werden, dass kein Buchstabe mehr als einmal in der Kodierung vorkommt und dass nur die Zahlen 0 bis 9 kodiert werden.

Schreiben Sie ein Programm "Zahlen.java", das einen String, der eine Anzeige kodiert, einliest und die kodierte Zahl als Integer ausgibt. Überlegen Sie wie viele IF Blöcke benötigt werden um jede Zahl zu erkennen.

Tipp: Sie können `str.contains("a")` verwenden, um zu überprüfen, ob ein String `str` den Buchstaben 'a' enthält.



Aufgabe 5: Scrabble

In dieser Aufgabe sollen Sie Scrabble-Steine legen, mittels ASCII-Art auf der Konsole. Vervollständigen Sie die Methode `drawNameSquare` in der Klasse `Scrabble`. Diese Methode nimmt einen Namen als String-Parameter und soll den Namen als in einem Quadrat angeordnete Scrabble-Steine auf der Konsole (`System.out`) ausgeben. Wenn z.B. der String `Alfred` übergeben wird, sollte folgendes Bild ausgegeben werden:

```
+---+---+---+---+---+
| A | L | F | R | E | D |
+---+---+---+---+---+
| L |           | E |
+---+           +---+
| F |           | R |
+---+           +---+
| R |           | F |
+---+           +---+
| E |           | L |
+---+---+---+---+---+
| D | E | R | F | L | A |
+---+---+---+---+---+
```

Aufgabe 5: Scrabble

Ihr Code braucht nur Namen der Länge 3 oder länger unterstützen. Für einen Namen mit Länge 3, z.B. Jim, sollte die Ausgabe so aussehen:

```
+---+---+---+
| J | I | M |
+---+---+---+
| I |   | I |
+---+---+---+
| M | I | J |
+---+---+---+
```

Beachten Sie, dass Ihr Programm keinerlei andere Ausgabe als das Scrabble-Quadrat machen darf.

Aufgabe 6: Weakest Precondition

Bitte geben Sie für die folgenden Programmsegmente die schwächste Vorbedingung (weakest precondition) an. Bitte verwenden Sie Java Syntax (die Klammern { und } können Sie weglassen). Alle Anweisungen sind Teil einer Java Methode. Alle Variablen sind vom Typ `int` und es gibt keinen Overflow.

1.

P: { ?? }

S: `m = n * 4; k = m - 2;`

Q: { `n > 0 && k > 5` }

2.

P: { ?? }

S: `m = n * n; k = m * 2;`

Q: { `k > 0` }

3.

P: { ?? }

S: `if (x > y) {`
 `z = x - y;`
 `} else {`
 `z = y - x;`
 `}`

Q: { `z > 0` }

Beispiel

P: { ?? }

S: `a = b * 3; c = a + 1;`

Q: { `a > 0 && c < 5` }

Aufgabe 7: Berechnungen

In dieser Aufgabe implementieren Sie verschiedene Berechnungen.

1. Implementieren Sie die Methode `Calculations.checksum(int x)`, das heisst die Methode `checksum` in der Klasse `Calculations`. Die Methode nimmt einen Integer `x` als Argument, welcher einen nicht-negativen Wert hat. Die Methode soll die Quersumme von `x` zurückgeben. Sie sollen für diese Aufgabe **keine** Schleife verwenden.

Beispiele

- `checksum(258)` gibt 15 zurück.
- `checksum(49)` gibt 13 zurück.
- `checksum(12)` gibt 3 zurück.

Hinweis: Für einen Integer `a` ist `a % 10` die letzte Ziffer und `a / 10` entfernt die letzte Ziffer. Zum Beispiel `258 % 10` ist 8 und `258 / 10` ist 25.

2. Implementieren Sie die Methode `Calculations.magic7(int a, int b)`. Die Methode gibt einen Boolean zurück. Die Methode soll `true` zurückgeben, wenn einer der Parameter 7 ist oder wenn die Summe oder Differenz der Parameter 7 ist. Ansonsten soll die Methode `false` zurückgeben.

Beispiele

- `magic7(2,5)` gibt `true` zurück.
- `magic7(7,9)` gibt `true` zurück.
- `magic7(5,6)` gibt `false` zurück.

Hinweis: Mit der Funktion `Math.abs(num)` können Sie den absoluten Wert einer Zahl `num` erhalten.

3. Implementieren Sie die Methode `Calculations.fast12(int z)`. Das Argument `z` ist nicht negativ. Die Methode gibt einen Boolean zurück. Die Methode soll `true` zurückgeben, wenn `z` nahe einem Vielfachen von 12 ist. Eine Zahl `x` ist nahe an einer Zahl `y`, wenn eine der Zahlen um maximal 2 grösser oder kleiner ist als die andere Zahl. Ansonsten soll die Methode `false` zurückgeben.

Beispiele

- `fast12(12)` gibt `true` zurück.
- `fast12(14)` gibt `true` zurück.
- `fast12(10)` gibt `true` zurück.
- `fast12(15)` gibt `false` zurück.

Zusatzübungen

Arithmetische Ausdrücke

Was wird berechnet?

- $4 / 3 + 16 \% 8 + 20 / 6 + 3.0 / 2 * 5 / 2$
- $19 \% 16 * 9 / 5 * 2.0$

Arithmetische Ausdrücke

Was wird berechnet?

- $4 / 3 + 16 \% 8 + 20 / 6 + 3.0 / 2 * 5 / 2$ **7.75**
- $19 \% 16 * 9 / 5 * 2.0$ **10.0**

Schleifen

Was gibt diese Methode aus?

```
public static void main (String[] args) {  
    int f = 0; int g = 1;  
  
    for (int i = 0; i < 15; i++ ) {  
        System.out.print(" " + f);  
        f = f + g;  
        g = f - g;  
    }  
    System.out.println();  
}
```

Schleifen

Was gibt diese Methode aus?

```
public static void main (String[] args) {  
    int f = 0; int g = 1;  
  
    for (int i = 0; i < 15; i++ ) {  
        System.out.print(" " + f);  
        f = f + g;  
        g = f - g;  
    }  
    System.out.println();  
}
```

i	f	g
	0	1
0	0	1
0	0	1
0	1	1
0	1	0
1	1	0
...

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Potenzieren

Schreiben Sie ein Programm, das zwei ganze Zahlen n , $k > 0$ einliest und n^k berechnet und ausgibt.

Potenzieren

Schreiben Sie ein Programm, das zwei ganze Zahlen n , $k > 0$ einliest und n^k berechnet und ausgibt.

```
public static void main(String[] args) {
    int n, k;

    Scanner console = new Scanner(System.in);

    System.out.print("Geben Sie Zahl 1 und 2 ein: ");
    n = console.nextInt();
    k = console.nextInt();

    int pot = 1;
    for (int i = 1; i <= k; i++) {
        pot = pot * n;
    }

    System.out.println(n + " hoch " + k + " = " + pot);
}
```

While-Schleife

Welche Werte haben `m` und `n` nach Ausführung des folgenden Codes?

```
int m = 0;
int n = 123456789;

while (n != 0) {
    m = (10 * m) + (n % 10);
    n = n / 10;
}
```

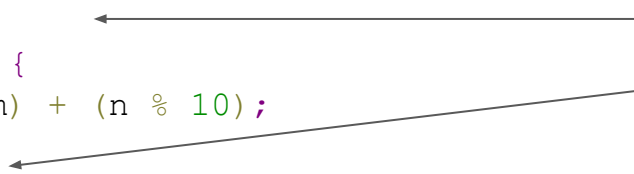
While-Schleife

Welche Werte haben `m` und `n` nach Ausführung des folgenden Codes?

```
int m = 0;
int n = 123456789;

while (n != 0) {
    m = (10 * m) + (n % 10);
    n = n / 10;
}
```

m	n
0	123456789
9	12345678
98	1234567
987	123456
...	...
987654321	0



Sortieren

Schreiben Sie ein Programm, das drei `int` Zahlen `a`, `b`, `c` von der Konsole liest und der Grösse nach druckt, angefangen mit der kleinsten Zahl.

- a) Mithilfe verschiedener Entscheidungsbäume
- b) Unterteilung in Teilprobleme

Inkrement und Dekrement

Inkrement und Dekrement

Kurzform

variable~~++~~;

variable~~--~~;

Beispiele

```
int x = 2;
```

```
x++;
```

```
double note = 4.5;
```

```
note--;
```

Äquivalente ausführlichere Version

variable^② = variable^① + 1; //increment

variable = variable - 1; //decrement

② ① -1

// x = x + 1;

// x now stores 3

// note = note - 1;

// note now stores 3.5

Inkrement und Dekrement

Kurzform Äquivalente ausführlichere Version

`variable++; variable = variable + 1; //increment`

`variable--; variable = variable - 1; //decrement`

Beispiele

`int x = 2;`

`System.out.println(x++); // x = x + 1; x now stores 3`

`System.out.println(x++); // x = x + 1; x now stores 4`

Output:

2

3

Aktualisierung

```
for (int i = start ; i < bound; i++) {  
    // Statement  
}
```

Aktualisierung: i wird um 1 erhöht

```
for (int i = start ; i > bound; i--) {  
    // Statement  
}
```

Aktualisierung: i wird um 1 reduziert

++ (und --) oft in Aktualisierungen des Loop Counters

Inkrement und Dekrement

Kurzform Äquivalente ausführlichere Version

`variable++;` `variable = variable + 1;`

`variable--;` `variable = variable - 1;`

Variable wird *verwendet* und dann *verändert*

Dies gilt auch in Ausdrücken und Zuweisungen

Beispiel

```
int x = 2;
```

```
int y;
```

```
y = x++;
```

Inkrement und Dekrement

Kurzform Äquivalente ausführlichere Version

`variable++;` `variable = variable + 1;`

`variable--;` `variable = variable - 1;`

Variable wird verwendet und dann verändert

Dies gilt auch in Ausdrücken und Zuweisungen

Beispiel

`int x = 2;`

`int y;`

`y = x++;`

`int temp = x;`

`x++;`

`y = temp;`

//x:

//y:

Inkrement und Dekrement

Kurzform Äquivalente ausführlichere Version

`variable++;` `variable = variable + 1;`

`variable--;` `variable = variable - 1;`

Variable wird verwendet und dann verändert

Dies gilt auch in Ausdrücken und Zuweisungen

Beispiel

`int x = 2;`

`int y;`

`y = x++;`

`int temp = x;`

`x = x + 1;`

`y = temp;`

//x:

//y:

Inkrement und Dekrement

Kurzform

`variable++;`

`variable--;`

Äquivalente ausführlichere Version

`variable = variable + 1;` ←

`variable = variable - 1;`

⇒ Variable wird verwendet und dann verändert

Dies gilt auch in Ausdrücken und Zuweisungen

Beispiel

`int x = 2;`

`int y;`

`y = x++;`

① `int temp = x;`
② `x = x + 1;` *Veränderung ++*
`y = temp;`
= 2 //x: 3
//y: 2

Inkrement und Dekrement Puzzles

Poll

```
int x = 1;
int y = 0;
int z = 0;
y = x++;
z = x++ + x++;
```

```
int a = 1;
a = a++;
```

■ Wert a?

```
int i = 10;
int j = i-- - i--;
```

■ Wert i?

■ Wert j?

- Wert x?
- Wert y?
- Wert z?

0	1	2	3	4	5	6

Zuweisungen (Assignment Statement)

LHS = RHS;

LHS: Eine Variable (Typ int, long, boolean, double – später mehr)

RHS: Ein Ausdruck

Ablauf:

- 1.** Rechte Seite (RHS) wird berechnet
- 2.** Resultat(Wert) wird in Variable (LHS) gespeichert

Beispiele für RHS mit Resultat:

```
int i = 3;  
int j = 7;
```

```
9          9
```

```
3+5        8
```

```
i+2        5
```

```
i++        3
```

```
// i: 4
```

```
j-- + j%4  9
```

```
// j: 6
```

Inkrement und Dekrement Puzzles

```
int a = 1;  
a = a++;  
// in Zeitlupe
```

```
int a = 1;  
int temp = a; // 1  
a = a + 1;     // 2  
a = temp;     // 1
```

- Wert a? 1

```
int i = 10;  
int j = i-- - i--;  
// in Zeitlupe  
int i = 10;  
int temp1 = i; // 10  
i = i - 1;     // 9  
int temp2 = i; // 9  
i = i - 1;     // 8  
j = temp1 - temp2; // 1
```

- Wert i? 8
- Wert j? 1

Inkrement und Dekrement **Puzzles**

- Unser Ziel ist es *verständliche* Programme zu schreiben

Inkrement und Dekrement **Puzzles**

- Unser Ziel ist es *verständliche* Programme zu schreiben
- ... und nicht Puzzles zu konstruieren!
- Sie sollten ++ und -- (er)kennen
 - Auch in komplexen Ausdrücken
 - Ihre Entscheidung ob sie es verwenden (aber wenn dann richtig)
- Diese Operatoren sind *nicht* so effizient dass wir dafür die Klarheit eines Programmes opfern wollen.

Weitere Kurzformen

- Erlauben Verwendung des Wertes einer Variable gefolgt von einer Modifikation (Zuweisung)

Kurzform Äquivalente ausführlichere Version

`variable += value ;` `variable = variable + value ;`

`variable -= value ;` `variable = variable - value ;`

`variable *= value ;` `variable = variable * value ;`

`variable /= value ;` `variable = variable / value ;`

`variable %= value ;` `variable = variable % value ;`

- Modifikation mit beliebigen Werten (nicht nur 1)

Weitere Kurzformen

Beispiele

```
x += 3;    // x = x + 3;
```

```
note -= 0.5; // note = note - 0.5;
```

```
number *= 2; // number = number * 2;
```

Warnung:

```
x += 1;    // x = x + 1;
```

```
x =+ 1;    // x = + 1;
```

Kahoot