Einführung in die Informatik: Übung 6

Alexander Waldenmaier

18. Dezember 2020

Aufgabe 1: Verschachteln

Ausgabe:

```
x: 0
y: 0
z: false
x: 5
y: 0
z: false
x: 5
y: 25
z: true
```

Zunächst werden im class body die Variablen `x`, `y`, und `z` initialisiert, die fortan als globale Variablen zur Verfügung stehen.

Beim Aufruf der Funktion `methode1` werden die globalen Variablen `x` und `y` mitgegeben und innerhalb der Funktion nun als lokale Variablen verändert. Diese Änderung überträgt sich allerdings nicht zurück in die `main`, weshalb dort zum Zeitpunkt 1 die Werte `0, 0, false` ausgegeben werden - also die Initialisierungswerte der globalen Variablen.

Nun wird `methode2` mit den Werten von `x` und `y` aufgerufen. Innerhalb der Methode wird der Wert von `x` der lokalen Variable `a` zugeschrieben und der Wert von `y` der lokalen Variable `y`. Da somit kein lokales `x` vorhanden ist, wird bei der Zuweisung `x = a + 3` das globale `x` verändert, das damit den Wert 5 erhält. Somit werden zum Zeitpunkt 2 die globalen Werte `5, 0, false` ausgegeben.

Schlielßlich wird die Funktion `methode3` mit genau diesen Werten aufgerufen. Innerhalb der Methode erhalten die drei Werte jedoch die Bezeichner `a, b, c` - somit referenziert jeder Aufruf von `x, y, z` nun die globalen Variablen. Die Rechnung läuft dann wie folgt ab:

```
a = y;  // a = 0
b = a + 5;  // b = 10
x = a + b;  // x = 10
y = x * 5;  // y = 25
if (y < x)  // false
    c = true;
else
    z = true;  // z = true
```

Folglich werden dann zum Zeitpunkt 3 die Werte `5, 25, true` ausgegeben.

Aufgabe 2: Funktionsüberladung

a)/b) Siehe auch „Konkatenieren.java“. Das Programm wurde um die folgenden beiden Methoden erweitert:

```
private static String concat(String a, String b) {
    String retval = a + b;
    if (retval.contains("Katze")) cat();
    return a + b;
}

private static String concat(char a, String b) {
```

```
            String s = String.valueOf(a);
            return concat(s, b);
        }
```

Beide Methoden tragen den selben Namen, `concat`, allerdings unterscheiden sich ihre Methodensignaturen. Die eine Funktion nimmt zwei Strings an, die andere einen Char an erster Stelle und einen String an zweiter Stelle. Ruft man `concat` mit einem Char und einem String auf, so wird die zweite Methode ausgeführt. Diese konvertiert den Char in einen String und ruft dann erneut `concat` auf, was in diesem Fall aber die erste Methode erreicht (da nun zwei Strings im Aufruf übergeben werden). In der ersten Methode werden die beiden Strings zusammengefügt („konkateniert“) und zurückgegeben. Sollte das Teilwort „Katze“ im String enthalten sein, wird zunächst noch die Funktion `cat` aufgerufen.

Die Überladung ist notwendig, da in der `main`-Metode `concat` sowohl mit der Kombination Char + String, als auch mit der Kombination String + String aufgerufen wird, und dabei jedes Mal ein String als Resultat erwartet wird.

Unter Überladung versteht man also generell die Definition mehrerer Methoden mit dem selben Namen, aber unterschiedlichen Methodensignaturen - also unterschiedlichen zu übergebenden Parametern. Entscheidend ist dabei der Typ der Parameter, die Reihenfolge und die Anzahl, nicht aber deren lokal zugewiesene Variablennamen. Übergibt man also die Parameter Char + String (in dieser Reihenfolge) ist klar, dass damit nur die zweite `concat`-Methode gemeint sein kann, die genau diese Datentypen in dieser Reihenfolge annimmt. Genauso verweist die Übergabe zweier Strings eindeutig auf die erste Definition. Übergibt man hingegen String + Char (in dieser Reihenfolge) würde Java eine Fehlermeldung produzieren, da keine Methode für diesen Funktionsaufruf vorhanden ist.

c) Nein. Da beide Methoden den selben Namen tragen, müssen sich ihre Methodensignaturen unterscheiden. Da beide Funktionen aber exakt einen String akzeptieren (die Variablennamen spielen keine Rolle), kann beim Funktionsaufruf nicht unterschieden werden, ob die eine oder die andere gemeint ist. Der Return-Typ der Funktion spielt bei dieser Beurteilung keinerlei Rolle.

Aufgabe 3: Call-by

Die Ausgaben der Programme sind jeweils hinter den print-Aufrufen kommentiert.

Call by reference

```
public class Program {

    // Erstelle globale Variablen a, b, c
    static String a = "Blumen"
    static int b = 20;
    static boolean c = false;

    // Erhaelt Referenzen zu a, b, c
    static int m (String x, int y, boolean c) {
        // Setze Wert von Referenz x, also globales a,
        // auf "Ich habe 20 Blumen"
        x = "Ich habe " + y + " " + x;
        // Setze Wert von Referenz c, also globales c, auf true
        c = true;
        // Setze Wert von Referenz y, also globales b, auf 30
        y = b + 10;
        return b;  // Rueckgabe der Referenz auf globales b
    }

    // Erhealt Referenz auf b
    static void n(int i) {
        if (c)  // c = true
            // Setze Wert von Referenz i, also globales b, auf 60
```

```
            i = i + i;
        else
            i = i - 2;
    }

    public static void main(String[] args) {
        // Uebergebe Referenzen von a, b, c an m().
        // Uebergebe dann Referenz zu return-Wert von m() an n().
        n(m(a, b, c))

        System.out.println(a);  // Ausgabe: Ich habe 20 Blumen
        System.out.println(b);  // Ausgabe: 60
        System.out.println(c);  // Ausgabe: true
    }
}
```

Call by value

```
public class Program {

    // Erstelle globale Variablen a, b, c
    static String a = "Blumen"
    static int b = 20;
    static boolean c = false;

    // Erhaelt Werte von a, b, c in lokalen Variablen x, y, c
    static int m (String x, int y, boolean c) {
        // Setze lokales x auf "Ich habe 20 Blumen"
        x = "Ich habe " + y + " " + x;
        // Setze lokales c auf true
        c = true;
        // Setze lokales y auf 30
        y = b + 10;
        return b;  // Rueckgabe des globalen Werts 20
    }

    // Erhaelt Wert 20
    static void n(int i) {
        if (c)  // c = false
            i = i + i;
        else
            // Setze lokales i auf 18
            i = i - 2;
    }

    public static void main(String[] args) {
        // Uebergebe Werte der globalen Variablen a, b, c an m()
        // Uebergebe Rueckgabewert von m() an n()
        n(m(a, b, c))

        System.out.println(a);  // Ausgabe: Blumen
        System.out.println(b);  // Ausgabe: 20
        System.out.println(c);  // Ausgabe: false
    }
}
```