technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2013/14

Übungsblatt 6

Bearbeitungszeit:
09.12.–13.12.2013

Aufgabe 6.1 – Methoden und Sichtbarkeit von Variablen

Methoden sind wiederverwendbare Unterprogramme. Betrachten Sie Folie 10 aus Kapitel 4, auf der die einzelnen Elemente einer Methode erläutert werden. Betrachten Sie dann den folgenden Programmtext. Das Programm enthält eine selbstdefinierte Methode mit dem *Methodennamen* `pluszehn` (auf der Folie 10 wird der Methodenname „Funktionsbezeichner“ genannt). Die Methode ist als `static` deklariert; dies bedeutet, dass sie unabhängig von einem Objekt ausgeführt werden kann – darauf werden wir später noch genauer eingehen. Die Methode hat einen *Rückgabewert* vom Typ `int`. Des Weiteren hat die Methode einen *Eingabeparameter* vom Typ `int` mit dem Namen `c`.

```
1   class Sichtbarkeit {
2    public static void main (String[] args){
3      int a = 10;
4      int b = pluszehn(a);
5      System.out.println("Wert von a in main: " + a);
6      System.out.println("Wert von b in main: " + b);
7    }
8
9    public static int pluszehn(int c) {
10     int a = c + 10;
11     System.out.println("Wert von a in pluszehn: " + a);
12     return a;
13   }
14  }
```

Bearbeiten Sie nun folgende Teilaufgaben:

- Die Methode `pluszehn` wird innerhalb der Methode `main` in einer Zuweisung aufgerufen; geben Sie die Zeilennummer der Zuweisung an. _____
- Welchen Wert für `a` wird das Programm in Zeile 11 ausgeben? _____ Welchen Wert für `a` wird das Programm in Zeile 5 ausgeben? _____ Welchen Wert für `b` wird das Programm in Zeile 6 ausgeben? _____
- Das Programm enthält lokale Variablen, die nicht von überall aus zugreifbar sind. Markieren Sie die Sichtbarkeit der lokalen Variablen `a` und `b` sowie des Parameters `c` im Programmtext.

Aufgabe 6.2 – Eine eigene Methode

Schreiben Sie ein Programm mit einer von Ihnen selbst definierten Methode in folgenden Schritten:

- Schreiben Sie zunächst das Grundgerüst eines Java-Programms mit der `main`-Methode, wie es Ihnen bereits bekannt ist (siehe Aufgabe 2.3).
- Ergänzen Sie dann außerhalb der `main`-Methode, aber noch innerhalb der Klassendefinition, eine eigene Methode mit folgenden Eigenschaften:
  a) Da wir noch nicht mit Objekten arbeiten, muss Ihre Methode als `public static` (wie in Aufgabe 6.1) deklariert sein.

b) Der Datentyp des Rückgabewertes ist `int`.

c) Der Methodenname ist `plus`.

d) Die Methode hat zwei Eingabeparameter des Typs `int`; beachten Sie, dass die beiden Eingabeparameter durch ein Komma getrennt werden.

e) Die Methode hat eine lokale Variable vom Typ `int`, die das Ergebnis der Addition der beiden Eingabeparameter zugewiesen bekommt.

f) Die Methode gibt dieses Ergebnis als Rückgabewert zurück.

- Rufen Sie Ihre Methode innerhalb der `main`-Methode mit zwei beliebigen Eingabewerten auf. Weisen Sie dabei den Rückgabewert des Methodenaufrufs einer neuen `int`-Variable zu.
- Geben Sie in der `main`-Methode den Wert dieser neuen Variable auf dem Bildschirm aus.

Aufgabe 6.3 – Einfache Methoden

- Schreiben Sie ein Programm mit einer Methode mit dem Namen `max2`, die das Maximum zweier ganzer Zahlen bestimmt und dieses zurückgibt.
- Ändern Sie die Methode `max2` so ab, dass diese mit Fließkommazahlen arbeitet. Programmieren Sie eine weitere Methode mit dem Namen `max3`, die das Maximum dreier Fließkommazahlen mithilfe der Methode `max2` bestimmt und dieses zurückgibt.

Aufgabe 6.4 – Rekursion: Fakultätsberechnung

In Kapitel 4 der Vorlesung wurde das Konzept der Rekursion eingeführt. Schreiben Sie ein Programm mit einer Methode `public static int fakultaet_rek(int n)`, die die Fakultät einer Zahl *rekursiv* berechnet. Zur Erinnerung: die *Fakultät* $n!$ einer Zahl n ist folgendermaßen definiert:

$$n! := \begin{cases} 1 & \text{für } n = 0 \\ n \cdot (n-1)! & \text{für } n \geq 1 \end{cases}$$

Zum Beispiel ist

$$5! = 5 \cdot 4! = 5 \cdot 4 \cdot 3! = 5 \cdot 4 \cdot 3 \cdot 2! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120.$$

Wenn Sie möchten, können Sie die Zahl von der Tastatur einlesen lassen.

Aufgabe 6.5 – Rekursion: Fibonacci-Zahlen

Zur Erinnerung: die *Fibonacci-Folge* $(f_1, f_2, \ldots)$ ist folgendermaßen definiert:

$$f_n := \begin{cases} 1 & \text{für } n = 1 \\ 1 & \text{für } n = 2 \\ f_{n-1} + f_{n-2} & \text{für } n > 2 \end{cases}$$

Zum Beispiel ist das vierte Glied der Fibonacci-Folge

$$f_4 = f_3 + f_2 = (f_2 + f_1) + f_2 = (1 + 1) + 1 = 3.$$

Schreiben Sie ein Programm mit einer Methode `public static int fibonacci_rek(int n)`, die die Fibonacci-Zahlen *rekursiv* berechnet und geben Sie das Ergebnis in der `main` Methode aus.

Vergleichen Sie die rekursive Implementierung der Fibonacci-Zahlen mit der iterativen Implementierung aus der Aufgabe 4.12. Welche Lösung ist effizienter? Warum?

Aufgabe 6.6 – Potenzberechnung

Programmieren Sie eine Klasse mit einer Methode `public static int potenz(int b, int n)`, die die Potenz b^n einer ganzzahligen Basis b zu einem ganzzahligen, positiven Exponenten n berechnet. Verwenden Sie dazu eine der Schleifen.

Aufgabe 6.7 – Potenzberechnung rekursiv

Schreiben Sie nun eine zweite Methode `public static int potenz_rek(int b, int n)`, die wie eben beschrieben funktioniert, aber die die Potenz rekursiv berechnet. Benutzen Sie keine Schleifen!