

Informatik II

Zu jeder Aufgabe ist eine Datei abzugeben, deren Name rechts in der Aufgabenüberschrift steht. Stellen Sie die insgesamt drei Dateien in ein extra Verzeichnis (mit beliebigem Namen) und packen Sie dieses zu einem ZIP-Archiv. Geben Sie dieses, wie üblich, per UniWorx ab.

Aufgabe 2-1 Programmzustände (Zustand.java, 4 Punkte)

Wir stellen den Keller, in dem die Namen und Werte der lokalen Variablen während der Ausführung eines Programms gespeichert werden, durch eine Liste von Name/Wert Paaren dar. Hierbei steht dasjenige Name/Wert Paar am weitesten rechts, das zuletzt zum Keller hinzugefügt wurde.

So ist z.B. $[(x, 12), (y, \text{true})]$ eine Darstellung eines Kellers, in dem die Werte der Variablen x und y gespeichert werden. Wird eine neue Variable z , etwa durch `double z = 4.3` vereinbart, ändert sich obiger Keller zu $[(x, 12), (y, \text{true}), (z, 4.3)]$.

Betrachten Sie folgendes Java-Programm:

```
1  class Zustand
2  { public static void main(String[] args)
3    { int m = 2;
4      double d = 2.0;
5      d = m * d;
6      int n = 15 + (int) d;
7      { int k = 8;
8        d = d + k;
9      }
10     d = n / 4;
11     System.out.println(d);
12   }
13 }
```

Geben Sie die Zustände des Programms nach Ausführung von Zeile 3, 4, 5, 6, 7, 8 und 10 in der oben beschriebenen Notation an.

Aufgabe 2-2 Quadratzahltest (Quadratzahl.java, 4 Punkte)

Eine ganze Zahl $q \geq 0$ ist ein Quadratzahl, wenn es eine ganze Zahl n gibt mit $q = n * n$. So ist z.B. $16 = 4 * 4$ eine Quadratzahl, nicht aber 17.

Vervollständigen Sie das Programmgerüst

```
class Quadratzahl
{ public static void main(String[] args)
  { int q = 17;
    // Ihr Code
  }
}
```

zu einem Java-Programm, das für jeden gültigen Wert von q ausgibt, ob es sich bei q um eine Quadratzahl handelt.

Hinweis: Prüfen Sie (mit geeigneter Abbruchbedingung), ob $q = n * n$ für $n = 1, 2, \dots$

Aufgabe 2-3**Fakultätsfunktion**

(Fakultaet.txt, 8 Punkte)

Betrachten Sie folgendes Programm-Fragment zur Berechnung der Fakultätsfunktion:

```
public static int fakultaet(int n) {  
    // Anfang  
    int a = 1;  
    int k = n;  
    while (k > 0) {  
        a = a * k;  
        k--;  
    }  
    // Ende  
    return a;  
}
```

Sei c der Programmabschnitt zwischen den Kommentaren **Anfang** und **Ende**. Leiten Sie im Hoare-Kalkül das Hoare-Tripel $\{n \geq 0\} c \{a = n!\}$ – mithin die Korrektheitsaussage für die Methode **fakultaet** – ab. Geben Sie dazu immer die benutzte Regel an sowie die Ableitungen für die Prämissen. Sie können “top-down” oder “bottom-up” vorgehen, also von dem gewünschten Tripel her oder von den ersten Prämissen her (die Sie vermutlich in einem ersten Durchgang auf Papier herausfinden).

Hinweis: Die Schleifen-Invariante enthält unter Anderem die Aussage $a \cdot k! = n!$. Beachten Sie aber den Definitionsbereich von “!”.

Abgabe: Per UniWorx, bis spätestens 15.5.2006, 9:00 Uhr.