Aufgaben Programmiervorkurs
Übungsblatt 1

TECHNISCHE UNIVERSITÄT DARMSTADT

von Uli Fahrer & Dennis Albrecht

Wintersemester 2014/2015

Aufgabe 1 Multiple Choice

Kreuze zu jeder Antwort an, ob sie zutrifft (**w**) oder nicht (**f**).

Aufgabe 1.1 Theoriefragen (Stufe 1)

w	f	
☐	■	Ein Compiler führt ein Programm aus.
■	☐	Ein Compiler wandelt ein Progamm in eine andere Darstellung um.
☐	■	Java ist eine interpretierte Sprache.
☐	■	Java ist eine compilierte Sprache.
■	☐	Java wird sowohl compiliert als auch interpretiert.
☐	■	Compilierte Java-Programme können immer nur auf dem eigenen Rechner ausgeführt werden.

Antwort

*Die zutreffenden Antworten sind die Aussagen 2 und 5. Die Aussagen 3 und 4 sind, je nach Interpretation, falsch oder richtig: Java ist keine **reine** interpretierte oder **reine** compilierte Sprache.*

Aufgabe 1.2 Fehlerarten (Stufe 2)

w	f	
☐	■	Programme mit lexikalischen Fehlern können compiliert werden.
☐	■	Programme mit syntaktischen Fehlern können compiliert werden.
■	☐	Programme mit semantischen Fehlern können compiliert werden.

Überlege auch, ob Fehler, welche sich beim compilieren nicht bemerkbar machen, bei Programmausführung **zwingend** zu Fehlern führen.

Antwort

Die dritte Aussage trifft zu, die ersten beiden nicht. Ob sich ein semantischer Fehler bemerkbar macht, hängt vom Fehler selbst ab. Du kannst nur selbst für eine richtige Semantik sorgen.

Aufgabe 2 Zum warm werden

Aufgabe 2.1 Hello World Reloaded (Stufe 1)

Erweitere das in der Vorlesung kennen gelernte Hello-World-Programm um beliebige Ausgaben deiner Wahl. Mache dabei auch Gebrauch von Kommentaren und beantworte die folgenden Fragen:

a) Was ist das besondere an der Mainfunktion?

b) Ändere den Dateinamen deines Quellprogramms und versuche das Programm erneut zu compilieren. Was passiert und warum?

Antwort

a) Mit der Mainfunktion beginnt die Ausführung des Programms.

b) Das Programm lässt sich nicht mehr compilieren, weil der Klassenname nicht mit dem Dateinamen übereinstimmt.

Aufgabe 2.2 Finde den Fehler (Stufe 1)

Gegeben ist folgendes Listing. Finde alle Fehler und überlege dir, zu welchem Fehlertypen die Fehler gehören!

```
1 public class Error_Listing
2     public static void main(String[] args){
3         system.ou.print(Hello World)
4     }
5 }
```

Antwort

- *Nach dem Klassennamen fehlt eine öffnende Klammer ({).* *syntaktischer Fehler*
- *Hierbei soll etwas ausgegeben werden. Da Java case-sensitiv ist, muss System großgeschrieben werden, ebenso muss es „out“ anstatt „ou“ heißen.* *lexikalischer Fehler*
- *Im Allgemeinen sollte man jede Ausgabe mit einem Zeilenumbruch beenden.* *semantischer Fehler*
- *„Hello World“ ist der Text, welcher ausgegeben werden soll. Solche Zeichenketten müssen in Hochkommata (") gesetzt werden.* *syntaktischer Fehler*
- *Nach dem Ausgabebefehl fehlt das Semikolon.* *syntaktischer Fehler*

Aufgabe 2.3 Textausgaben (Stufe 1)

Es gibt in Java zwei Möglichkeiten (Befehle) um Texte einfach und schnell auszugeben.

a) Was ist der Unterschied/sind die Unterschiede zwischen den beiden Befehlen `System.out.print` und `System.out.println`?

b) Was bedeutet das für die folgenden Befehlsfolgen?

1.
```
System.out.print("To␣be␣")
System.out.print("or␣not␣")
System.out.print("to␣be?")
```

2.
```
System.out.println("To␣be␣")
System.out.println("or␣not␣")
System.out.println("to␣be?")
```

3.
```
System.out.print("To␣be␣")
System.out.print("or␣not␣")
System.out.println("to␣be?")
```

4.
```
System.out.println("To␣be␣")
System.out.print("or␣not␣")
System.out.println("to␣be?")
```

Antwort

a) Es gibt nur zwei Unterschiede zwischen den beiden Befehlen. Der eine ist direkt dem Namen zu entnehmen: Der `println`-Befehl hängt an den Text einen Zeilenumbruch an wohingegen der `print`-Befehl nur den Text ausgibt, den man als Argument übergibt.
Der zweite Unterschied ist deutlich subtiler, aber folgt aus dem ersten und kann einem manchmal das Leben etwas erleichtern: Der `println`-Befehl funktioniert auch ohne Argument und sorgt dann nur dafür, dass ein Zeilenumbruch an die bisherige Ausgabe angehängt wird und erspart einem dann Befehle wie System.out.println("") oder System.out.print("\n"). Der `print`-Befehl braucht immer ein Argument.

b) ¶ steht hier für einen zusätzlichen Zeilenumbruch nach dem Fragezeichen.

1. *To be or not to be?*
2. *To be*
   or not
   to be?¶
3. *To be or not to be?¶*
4. *To be*
   or not to be?¶

Aufgabe 3 Variablen

Aufgabe 3.1 Namenskonvention (Stufe 1)

Welches sind **gültige** Variablennamen?

a) Matrikelnummer
b) ÄhmKeineAhnung
c) _meinAlter
d) &Richtig
e) $Dollar
f) 5teVariable

Antwort

Die Antworten d) und f) sind nicht gültig.
Gültig sind somit: Matrikelnummer, ÄhmKeineAhnung, _meinAlter und $Dollar

Aufgabe 3.2 Variablenwerte (Stufe 2)

Welche Zuweisungen compilieren und werden ausgeführt und welche Werte wurden (wenn möglich) in den Variablen gespeichert? Die Anweisungen werden nacheinander ausgeführt. So bezeichnet *c* in der siebten Zuweisung das Ergebnis der dritten Zuweisung.

- int a; :
- float b = 42; :
- int c = 23; :
- char d = 73; :
- float e = 5.7; :
- int f = 9.81f; :
- int g = c; :
- double h = b; :
- int i = b; :
- int j = (int) b; :
- double k = j/g; :
- int l = d; :

Antwort

a hat keinen Wert
b = 42.0f
c = 23
d = 'I'
e → Fehler
f → Fehler
g = 23
h = 42.0
i → Fehler
j = 42
k=1.0
l=73

Aufgabe 3.3 Case-sensitive (Stufe 1)

Gegeben ist folgendes Listing:

```
1 int meinAlter = 21;
2 MeinAlter = MeinAlter + 1;
3 System.out.println(meinAlter);
```

Beim Compilieren der zweiten Zeile wirft der Compiler einen Fehler. Warum? Verbessern Sie das Listing.

Antwort

*Die Variable in Zeile 2 müsste **m**einAlter heißen und nicht **M**einAlter.*

Aufgabe 3.4 the awesome 42 (Stufe 2)

Deklariere eine Integer Variable und initialsiere sie mit dem Wert 42.5. Warum gibt es Compilerfehler? Erkläre dieses Verhalten.

Antwort

Eine Integer-Variable speichert Ganzzahlen, allerdings ist 42.5 ein Fließkommawert. Folglich findet eine Typverengung statt, die der Compiler nicht ohne explizite Aufforderung durchführt.

Aufgabe 3.5 Zahlentypen (Stufe 3)

a) Deklariere und initialisiere eine double Konstante für PI (Wert 3,14159...).

b) Deklariere eine float Variable und initalisiere sie mit dem Wert der Pi-Konstanten. Denke daran, dass hier eine Typverengung stattfindet, die der Compiler nicht ohne weiteres akzeptiert.

c) Deklariere eine Variable vom Typ Byte und initialisiere diese mit dem Dezimalwert 127. Inkrementiere anschließend die Variable. Gib nun den Wert der Variable auf der Konsole aus. Was stellst du fest? Erkläre was passiert ist. Halte das Ergebnisse in einem Kommentar fest.

Antwort

```
1  public class Zahlentypen {
2      public static void main(String[] args) {
3          // a)
4          final double PI = 3.14159;
5
6          //b) Man braucht einen Cast um double in float umzuwandeln
7          float float_PI = (float)PI;
8
9          //c) Dies erzeugt ein Ueberlauf und b hat dann den Wert -128
10         // Bei der Bearbeitung musste man beachten, dass der Compiler
11         // <byte> + 1 als <byte> + <int> versteht und das Ergebnis
12         // vom Typ int ist. Nach einer expliziten Typkonvertierung
13         // konnte man das gewuenschte Verhalten beobachten.
14         byte b = 127;
15         b = (byte)(b + 1);
16         System.out.println(b);
17     }
18 }
```

Aufgabe 3.6 Ein- und Ausgabe (Stufe 2)

Im Rahmen der Vorlesung hast du die Klasse Input kennengelernt, welche es dir erlaubt, auf einfache Art und Weise Benutzereingaben von der Konsole einzulesen. Schreibe jetzt ein Programm, welches mit Hilfe der Input-Klasse nacheinander folgende Eingaben einließt und in Variablen mit jeweils passendem Datentyp speichert.

- deinen Namen als Zeichenkette
- dein Geburtsdatum als drei ganze Zahlen: Tag, Monat, Jahr.

Insgesamt sollen also vier Eingaben verarbeiten werden. Achte dabei darauf dass dein Programm die erwarteten Eingaben textuell klar strukturiert (Stichwort Bedienerfreundlichkeit).
Nachdem die letzte Eingabe verarbeitet ist, soll folgende Zeichenkette ausgegeben werden (wobei natürlich die Platzhalter ersetzt werden sollen):
Ausgabe: "<dein Name> hat am <tt.mm.jjjj> Geburtstag."
Hinweis: Die Verarbeitung von Eingaben wird im Foliensatz 'Variablen.pdf' erläutert.

Antwort

```
1  public class InputOutput {
2      public static void main(String[] args) {
3          String name;
4          int tag,monat,jahr;
5
6          System.out.print("Bitte gib deinen Namen ein: ");
7          name = Input.readString();
8
9          System.out.print("Gib den Tag deiner Geburt ein: ");
10         tag = Input.readInt();
11
12         System.out.print("Gib den Monat deiner Geburt ein: ");
13         monat = Input.readInt();
14
15         System.out.print("Gib das Jahr deiner Geburt ein: ");
16         jahr = Input.readInt();
17
18         System.out.println(name + " hat am " + tag + "."
19             + monat + "." + jahr + " Geburtstag.");
20     }
21 }
```

Aufgabe 4 Arithmetische Grundlagen

Aufgabe 4.1 Quadratische Funktionen (Stufe 3)

Gegeben sind die Funktionen $f(x) = x^2 - 6x + 5$ und $g(x) = -x^2 + 2x + 3$.
Berechne die Nullstellen der polynomiellen Funktionen $f(x)$ und $g(x)$ und speichere die Ergebnisse in einer geeigneten Variable.
Hinweise:

- Sei $f(x) = x^2 + px + q$. PQ-Formel $x_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}$
- Wurzelfunktion: $Math.sqrt(operand)$
- Potenzieren: $Math.pow(base, exponent)$

Antwort

```
1  public class PQ_Formel {
2      public static void main(String[] args) {
3          double f1 = -(-6.0 / 2) + Math.sqrt(Math.pow(-6.0/2, 2)-5);
4          double f2 = -(-6.0 / 2) - Math.sqrt(Math.pow(-6.0/2, 2)-5);
5          System.out.println(f1 + "," + f2);
6
7          double g1 = -(-2.0 / 2) + Math.sqrt(Math.pow(-2.0/2, 2)+3);
8          double g2 = -(-2.0 / 2) - Math.sqrt(Math.pow(-2.0/2, 2)+3);
9          System.out.println(g1 + "," + g2);
10     }
11 }
```

Aufgabe 4.2 Gaußsche Summenformel (Stufe 3)

Die **gaußsche Summenformel** ist eine Formel für die Summe der ersten n aufeinander folgenden natürlichen Zahlen $\mathbb{N}$. Sie ist gegeben durch:

$$1 + 2 + 3 + \ldots + n = \sum_{k=1}^{n} k = \frac{n \cdot (n+1)}{2}$$

a) Berechne die Summe der ersten **100** natürlichen Zahlen und speichere das Ergebnis in einer Variable **sum**. Wähle dafür einen geeigneten Datentyp!

b) Gib dieses Ergebnis anschließend auf der Konsole aus.

Hinweis: Verwende die geschlossene Form der Summe (den Bruch rechts des Gleichheitszeichens).

Antwort

```
1  public class Summenformel {
2      public static void main(String[] args) {
3          int sum = 100 * (100 + 1) / 2;
4          System.out.println("Die Summe der ersten 100 Zahlen ist " + sum);
5      }
6  }
```