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 5

Bearbeitungszeit:
02.–06.12.2013

Aufgabe 5.1 – Aufgaben von Blatt 4

Bitte lösen Sie zunächst die regulären Aufgaben von Blatt 4, die Sie noch nicht bearbeitet haben.

Aufgabe 5.2 – Syntaktische Varianten

Gegeben sei eine ganzzahlige Variable x. Geben Sie vier syntaktisch verschiedene Möglichkeiten an, den Wert der Variablen um 1 zu erhöhen.

Aufgabe 5.3 – Bedingte Anweisung und Schleifen

Bearbeiten Sie diese Aufgabe zunächst auf dem Papier!

Was gibt das folgende Programm aus? Terminiert das Programm? Wie lässt sich das Programm vereinfachen? Hierbei muss die Funktionalität erhalten bleiben. Dies bedeutet hier, dass das Programm nach der Vereinfachung noch dasselbe ausgeben soll wie vorher.

```
1  class SchleifenTest {
2    public static void main(String[] args) {
3      int x = 5;
4      int i = 0;
5      while (i < 5) {
6        for (i = 0; i < 3; i++){
7          System.out.println("i : " + i);
8        }
9        if (i > x) {
10         x = x;
11       } else if (i <= x) {
12         x = x;
13       } else if (x > 4) {
14         x = 14*i - 38*x/(x*3);
15       } else {
16         x = 14*i - 38;
17       }
18     }
19     System.out.println("x : " + x);
20   }
21 }
```

Aufgabe 5.4 – Einrückung

Sie haben schon gelernt, dass Quelltextzeilen mit geschweiften Klammern („{“ und „}“) umgeben werden, um zu markieren, dass sie zusammen gehören (*Blöcke*).

Zum Beispiel wird alles, was zu einer Klasse gehört, mit `{` und `}` umgeben:

```
class Klassenname {
  // hier ist der Inhalt der Klasse
}
```

Gleiches gilt u. a. für if-, while- und for-Anweisungen und Methodendefinitionen (diese werden in der nächsten Woche besprochen). Innerhalb solcher Blöcke können weitere Blöcke auftreten: Sie sind *verschachtelt*. Um zu verdeutlichen, wie die Blöcke ineinander verschachtelt sind, werden die Blöcke im Quelltext *eingerückt*. Dies bedeutet, dass zu Beginn jeder Zeile, die zu einem Block gehört, zwei zusätzliche Leerzeichen gesetzt werden. Wird innerhalb eines Blocks wieder ein Block begonnen, so sind es insgesamt vier Leerzeichen usw. Ein Beispiel (Leerzeichen sind speziell markiert):

```
class␣Einrueck␣{
␣␣public␣static␣void␣main(String[]␣args)␣{
␣␣␣␣while␣(true)␣{
␣␣␣␣␣␣System.out.println("Endlosschleife!");
␣␣␣␣}
␣␣}
}
```

Berücksichtigen Sie noch Folgendes: 1) Die schließende geschweifte Klammer wird nicht eingerückt und steht in einer separaten Zeile (außer vor `else`). 2) In jeder Zeile sollte höchstens eine Anweisung stehen.

Es gibt noch andere Möglichkeiten der Einrückung, aber der Einfachheit halber wollen wir in diesem Praktikum diese Konventionen verwenden. **Halten Sie sich ab jetzt daran!** Bitte berücksichtigen Sie, dass auf den Vorlesungsfolien aus Platzgründen zum Teil von der Konvention abgewichen wird.

Korrigieren Sie nun in folgendem Quelltext die Fehler, die beim Einrücken gemacht wurden.

```
1  class KaputteEinrueckung {
2    public static void main(String[] args) {
3        System.out.println("Zahlen, bitte!");
4      for (int i = 0; i < 20; ++i)
5      {
6        if (i == 0) { System.out.print("Die erste Zahl: ");
7        } else {
8        System.out.print("Noch eine Zahl: ");
9        }
10     System.out.println(i);
11     }
12 System.out.println("Das wars schon."); } }
```

Hinweis: Dr. Java kann Sie bei der Einhaltung der Konventionen unterstützen und die Einrückung (in gewissen Grenzen) anpassen. Dazu markieren Sie den gesamten Quelltext (z.B. durch Drücken von Strg+A) und drücken die TAB-Taste.

Aufgabe 5.5 – break und continue

Betrachten Sie das folgende Quelltext-Fragment:

```
1 for (int i = 0; i < 10; i++) {
2   System.out.println("i=" + i);
3 }
```

Welche Ausgabe erwarten Sie?

Führen Sie nun die folgende Variante in Java aus:

```
1 for (int i = 0; i < 10; i++) {
2   if (i == 3) {
3     continue;
4   }
5   if (i == 7) {
6     break;
7   }
8   System.out.println("i=" + i);
9 }
```

Welche Wirkung haben die Anweisungen `break` und `continue`?

Aufgabe 5.6 – `for`-Schleifen

Betrachten Sie das folgende Quelltext-Fragment:

```
1  Scanner scanner = new Scanner(System.in);
2  int n = scanner.nextInt();
3  int s = 0;
4
5  while (true) {
6    if (2 > n) {
7      break;
8    }
9    if (n % 2 == 0) {
10     s += n;
11   }
12   n--;
13 }
```

Programmieren Sie eine alternative Version, welche eine `for`-Schleife verwendet und dasselbe Berechnungsergebnis für s liefert.

Aufgabe 5.7 – Fallunterscheidung mit `switch–case–default`

Die switch-Anweisung wird in Kapitel 3.2 auf Folie 16 erklärt.

Schreiben Sie ein Programm, das den Benutzer um die Eingabe einer Zahl von 1 bis 12 bittet und dann mit Hilfe einer `switch–case–default`-Anweisung den dazu passenden Monatsnamen ausgibt (z. B. „März“ bei der Eingabe von 3). Nutzen Sie den `default`-Teil des `switch` in geeigneter Weise, um Fehleingaben abzufangen.

Aufgabe 5.8 – Spielergebnis

In dieser Aufgabe sollen Sie für das Ergebnis eines Würfelspiels die Punktzahl berechnen. Ihr Programm soll es dem Benutzer erlauben, die Augenzahlen eines Wurfes mit zwei Würfeln einzugeben, und aus diesen Werten dann gemäß folgender Regeln das Spielergebnis berechnen:

- Besteht der Wurf aus einer 1 und einer 2, so ist das Spielergebnis 1000.
- Für ein „Pasch“, das heißt einen Wurf mit zwei gleichen Augenzahlen, erhält der Spieler 100· *Augenzahl* als Spielergebnis. Der Wurf 3, 3 hat zum Beispiel den Wert 300.
- In allen anderen Fällen ist das Ergebnis 10· *(höhere Augenzahl)+(niedrigere Augenzahl)*. Der Wurf 2, 4 hat zum Beispiel den Wert 42.

Ihr Programm soll nach Eingabe der beiden Augenzahlen des Wurfes den berechneten Wert als Spielergebnis ausgeben.

Aufgabe 5.9 – Syntaxdiagramme

In der Vorlesung haben Sie Syntaxdiagramme kennen gelernt, um den Aufbau einer Anweisung darzustellen; sie wurden insbesondere auf den Folien 27–29 in Kapitel 2 eingeführt.
In Kapitel 3.1 auf Folie 26 finden sie ein Syntaxdiagramm für boolesche Ausdrücke. Nehmen Sie sich einen Moment Zeit, um den Aufbau und Bedeutung der einzelnen Bestandteile dieses Syntaxdiagramms zu verinnerlichen. Entscheiden sie dann mit diesem Syntaxdiagramm welche der folgenden Ausdrücke boolesche sind: (In Ausdruck i handelt es sich bei `d` um eine deklarierte und initialisierte Variable vom Typ `boolean`.)

a) `false` _____

b) `4 && 9` _____

c) `!(true && ( false || false)` _____

d) `true && 4 || 6` _____

e) `(false && true > false))` _____

f) `boolean c = false` _____

g) `99 < 0 && true || false` _____

h) `true && false || false && true && false` _____

i) `!d && d || (4 >= 8)` _____

j) `!99 < 0 && true || false` _____