technische universität
dortmund

Bearbeitet von:
Sebastian Hauer
Sergej Karsten

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm,
Alexander Lochmann, Bartho Rudak, Thomas Schmitz

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2014/15

Übungsblatt 4

Bearbeitungszeit:
17.11.14 -
21.11.14

Aufgabe 4.1 – Anweisungen und Blöcke

In Java, wie auch in anderen Programmiersprachen, ist es möglich, Anweisungen in sogenannten Blöcken zusammenzufassen. Ein Block wird durch eine öffnende und schließende geschweifte Klammer deklariert. Variablen, welche innerhalb eines Blocks deklariert werden, sind nur innerhalb dieses Blocks sichtbar, Variablen von außerhalb des Blocks aber immernoch innerhalb des Blocks. Um das entsprechende Verhalten zu verdeutlichen, erstellen Sie eine neue Klasse namens „BlockTest“ und übertragen Sie folgendes Programm in ihren Quellcode:

```
1  public static void main(String[] args){
2      int outer = 1;
3      int outerSum = 0;
4      //Begin Block
5      {
6          int inner = 2;
7          int innerSum = 0;
8          System.out.println( "Outer Integer: " + outer
9                              + "Inner Integer: " + inner);
10         outer++;
11         System.out.println( "I can manipulate the outer variable: "
12                             + outer);
13         innerSum = outer + inner;
14         System.out.println( "I can use both variables: "
15                             + innerSum);
16     }
17     //End Block
18     int inner = 5;
19     System.out.println( "I need to redeclare the integer \"inner\": "
20                         + inner);
21     outerSum = outer + inner;
22     System.out.println( "Now I can use the variable as a new one: "
23                         + outerSum);
24 }
```

Fügen Sie anschließend nach Zeile 17 eine Zeile mit der Anweisung `outer += inner;` ein. Welche Fehlermeldung wird Ihnen angezeigt?

Besprechen Sie den Grund für diese Fehlermeldung mit Ihrer Praktikumsgruppe. Löschen Sie die Zeile im Anschluss wieder.

Debuggen Sie anschließend das Programm mit einem Breakpoint in Zeile 2 und gehen Sie schrittweise durch das Programm. Markieren Sie auf dem Übungsblatt den Bereich, in welchem die Variable `innerSum` existiert. Der Codebereich, in welchem eine Variable existiert, nennt man auch „scope“.

Aufgabe 4.2 – Logische Operatoren

In der Vorlesung haben Sie die logischen Operatoren `&&`, `||` und `!` kennengelernt. Tragen Sie in die untenstehenden Tabellen die Ergebnisse der angegebenen Ausdrücke ein:

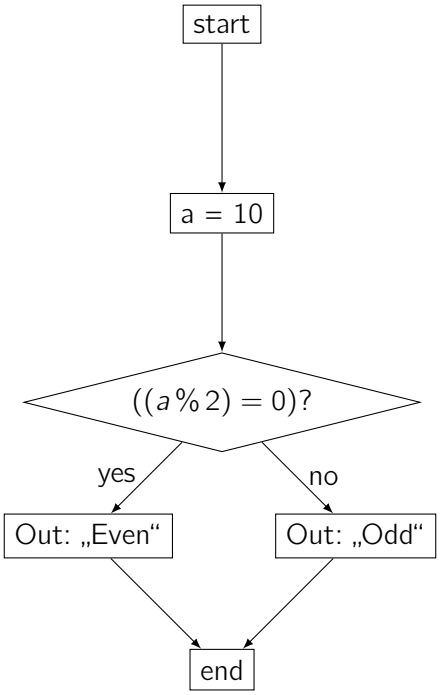
Ausdruck	**Ergebnis**
`true && true`	
`false && true`	
`true && false`	
`false && false`	
`!false`	
`!true`	

Ausdruck	**Ergebnis**
`true || true`	
`false || true`	
`true || false`	
`false || false`	
`!(false || true) && true`	
`true || (!false && false)`	

Aufgabe 4.3 – Programmflusskontrolle mit `if`

Wir werden Ihnen in den folgenden Aufgaben Diagramme präsentieren, die den Verlauf eines Programms repräsentieren sollen. Diese sollen Ihnen helfen, ein besseres Verständnis für den Programmfluss zu erhalten. Das repräsentierte Programm beginnt beim `start`-Knoten und endet beim `end`-Knoten. Die restlichen Knoten repräsentieren Anweisungen eines Programms, jedoch sind diese bewusst keine syntaktisch korrekten Anweisungen eines Java-Programms, sondern müssen von Ihnen in ein semantisch äquivalentes Programm übersetzt werden.
Knoten der Form `x = y` repräsentieren eine Zuweisungen des Wertes `y` an eine Variable mit dem Namen `x`.
Knoten der Form `(A)?` repräsentieren eine Fallunterscheidung A, die mit `yes` oder `no` beantwortet werden kann und entsprechend der Antwort den Programmfluss entlang einer mit dem entsprechendem Label beschrifteten Kante leitet.
Knoten der Form `Out: Ausdruck` geben den Ausdruck auf dem Bildschirm aus.

Sehen Sie sich folgendes Programmflussdiagram an. Was wird ausgegeben?



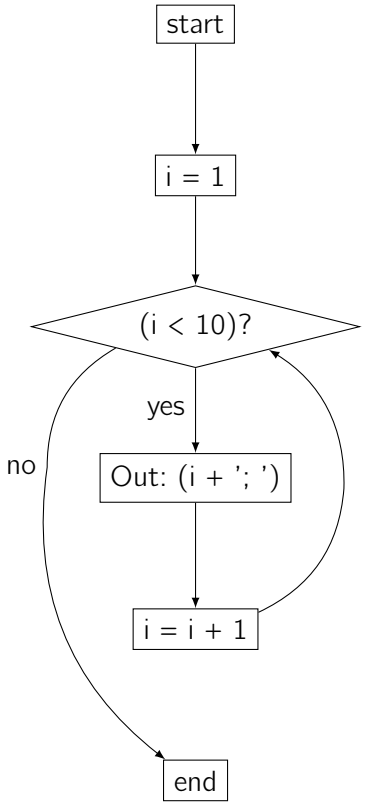
Erstellen Sie anschließend eine neue Klasse mit dem Namen „ParityTest“ und schreiben Sie in die main-Methode ein Programm, welches dem Verhalten des obrigen Diagramms entspricht.

Verwenden Sie dazu den Kontrollfluss kontrollierende `if`-Anweisung:

```
if(Bedingung){
    ...
} else {
    ...
}
```

Aufgabe 4.4 – Programmflusskontrolle mit `while`

Betrachten Sie folgendes Programmflussdiagramm:


start
i = 1
(i < 10)?
yes
no
Out: (i + '; ')
i = i + 1
end

Was wird ein durch dieses Diagramm repräsentiertes Programm ausgeben?

Legen Sie anschließend eine neue Klasse mit dem Namen „WhileTest1“ an und schreiben Sie eine main-Methode, die dem Verhalten des Diagrammes entspricht.

Verwenden Sie dabei das Konstrukt der `while`-Schleifen:

```
while(Bedingung){
    ...
}
```

Legen Sie weitere Klassen mit den Namen „WhileTestX“ an (ersetzen Sie X durch eine entsprechende Nummer) und schreiben Sie anschließend Programme mit `while`-Schleifen, welche folgende Ausgaben haben:

- 2;4;6;...;16;18
- 1;2;4;8;16;32;64;128;
- 50;45;40;...;-5;-10;
- **Zusatz:** 12;6;3;10;5;16;8;4;2;1;

Aufgabe 4.5 – `do ... while` **- Schleifen**

Der Inhalt einer `do ... while` - Schleife wird immer mindestens einmal durchgeführt und die Abbruchbedingung am Schluss überprüft. Überlegen Sie sich, wie ein entsprechender Abschnitt in einem Programmflussdiagramm aussehen müsste und zeichnen Sie einen solchen auf.

Aufgabe 4.6 – Inkrement, Dekrement und alternative Zuweisung

Mit den Operatoren `+=` , `-=`, `*=`, `/=` können Sie den Wert einer Variablen modifizieren, indem der aktuelle Wert der Variablen unter dem angegebenen Operator `+,-,*,/` mit dem Wert rechts vom = kombiniert wird und der Variablen als neuer Wert zugewiesen wird. So wird z.B mit `x *= 2;` der Wert der Variablen `x` verdoppelt.

Mit den Operatoren `++,−−` kann eine Variable vor oder nach ihrer Auswertung um 1 erhöht (inkrementiert) oder um 1 verringert (dekrementiert) werden. So wird z.B. mit der Anweisung `System.out.println(++i);` der Wert der Variablen i zuerst um 1 erhöht und dann ausgegeben, während ein `i++` zuerst den aktuellen Wert ausgeben und dann um 1 erhöhen würde.

Formen Sie die in den vorherigen Aufgaben geschriebenen Programme so um, dass Sie an passenden Stellen die entsprechenden Operatoren verwenden, falls Sie diese nicht bereits verwendet haben.