Praktikum Grundlagen der Programmierung

Prof. Dr. Harald Räcke, R. Palenta, A. Reuss, S. Schulze Frielinghaus WS 2016/17

Übungsblatt 3 Abgabefrist: 14.11.2016 (vor 05:00 Uhr)

Um Ihre Hausaufgaben auf Moodle abgeben zu können, müssen Sie in einer Gruppe in TUMonline eingetragen sein!

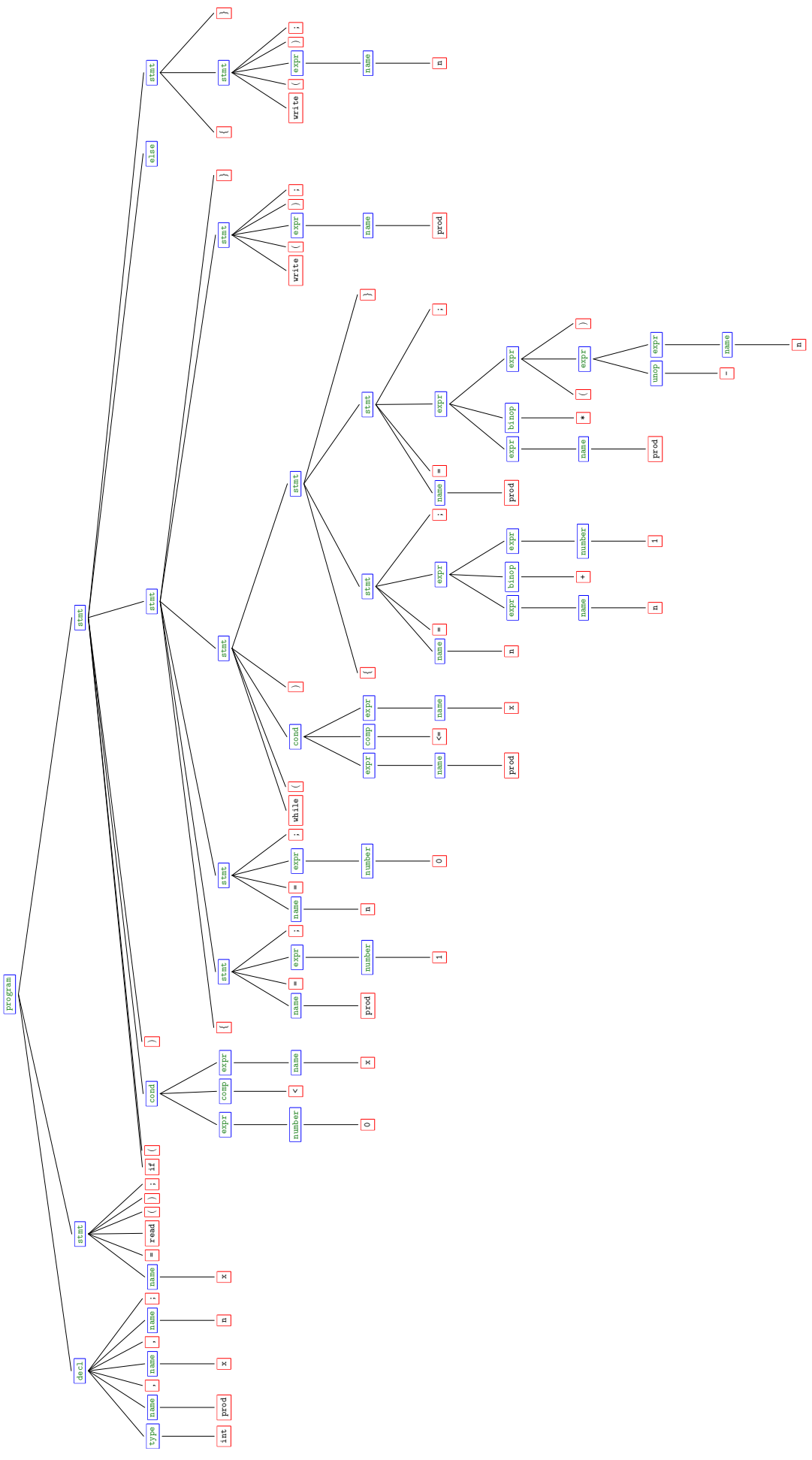
Hausaufgaben, die keine Programieraufgaben sind, müssen als PDF im A4-Format abgegeben werden.

Aufgabe 3.1 (P) Syntaxbaum

Zeichnen Sie für das folgende `MiniJava`-Programm den Syntaxbaum. Dazu steht Ihnen die Grammatik von `MiniJava` aus der Vorlesung zur Verfügung (eine Zusammenfassung finden Sie auch auf Moodle).

```
int prod, x, n;
x = read();
if (0 < x) {
    prod = 1;
    n = 0;
    while (prod <= x) {
        n = n + 1;
        prod = prod * (-n);
    }
    write(prod);
} else {
    write(n);
}
```

Lösungsvorschlag 3.1


program
decl
type
int
name
prod
,
name
x
,
name
n
;
stmt
name
x
=
read
(
)
;
stmt
if
(
cond
expr
name
x
comp
<
expr
number
0
)
{
stmt
while
(
cond
expr
name
x
comp
<
expr
name
prod
)
stmt
{
stmt
name
prod
=
expr
number
1
;
stmt
name
n
=
expr
number
0
;
}
stmt
write
(
expr
name
prod
)
;
else
stmt
{
stmt
write
(
expr
name
n
)
;
stmt
{
stmt
name
n
=
expr
name
n
-
expr
number
1
;
stmt
name
prod
=
expr
name
prod
*
expr
(
expr
name
n
-
number
1
)
;
}
}

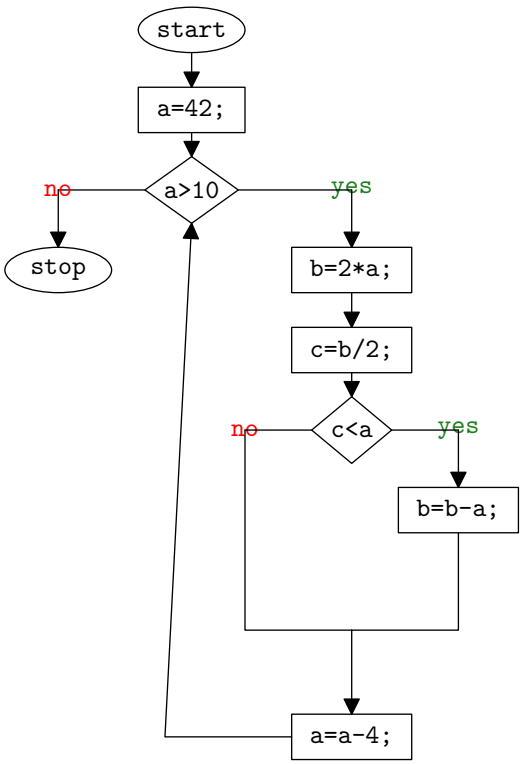
Aufgabe 3.2 (P) Kontrollflussdiagramm

Zeichnen Sie den Kontrollflussgraphen für das folgende `Java`-Fragment:

```
int b, c;
for (int a = 42; a > 10; a = a - 4) {
  b = 2 * a;
  c = b / 2;
  if (c < a) {
    b = b - a;
  }
}
```

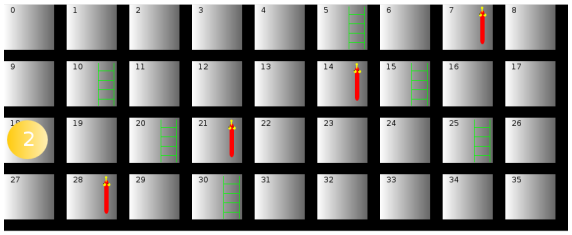
Hinweis: Zum Zeichnen können Sie z.B. das Programm *LibreOffice Draw* verwenden, welches Sie unter `http://www.libreoffice.org/` finden. Speichern Sie Ihren Kontrollflussgraphen im *pdf*-Format ab.

Lösungsvorschlag 3.2



Ein Semikolon nach einer Zuweisung oder nach einem `write`-Aufruf ist optional.

Aufgabe 3.3 (P) Schlangenspiel



In dieser Aufgabe sollen Sie eine Variante des Schlangenspiels für zwei Spieler programmieren. Erstellen Sie dazu ein Programm namens `Schlangenspiel.java`. Die Grundidee des Spiels ist folgende:

- Das Spiel hat die Felder 0 bis 35.
- Jeder Spieler besitzt einen Spielstein.
- Beide Spielsteine starten auf Feld 0.
- Im Wechsel wird gewürfelt. Der Spielstein des entsprechenden Spielers wird um die entsprechende Augenzahl vorgerückt.
- Wer zuerst das Feld 35 erreicht oder überschreitet, gewinnt das Spiel.
- Von allen Feldern, die durch 5 teilbar sind, führt eine Leiter nach oben. Wer ein solches Feld erreicht, kommt sofort 3 Felder weiter.
- Aber Vorsicht vor den Schlangen! Erreicht man ein Schlangenfeld (jedes Feld, das durch 7 teilbar ist), rutscht man automatisch um 4 Felder zurück.
- Die Felder 0 und 35 sind weder Leiter- noch Schlangenfelder.
- Leitern und Schlangen treten in Aktion, wenn ein Spielstein dieses Feld erreicht. Es ist daher möglich, *Ketten* von Schlangen und/oder Leitern zu benutzen.
- Es dürfen zwei Spielsteine gleichzeitig auf einem Feld stehen.
- Am Ende wird der Sieger ausgegeben.

Geben Sie jeweils das Würfelergebnis und das neue Spielfeld aus.

Implementieren Sie ihre Lösung Schritt für Schritt:

1. Beschränken Sie sich zuerst auf einen Spieler und ein leeres Spielfeld. Das Spiel besteht am Anfang nur aus Würfeln bis die 35 überschritten wird.
2. Beachten Sie nun Leitern und Schlangen – nach dem Würfeln prüfen Sie, ob das Zielfeld leer ist.
3. Beachten Sie nun auch Ketten von Leitern und Schlangen – bewegen Sie einen Spielstein nach dem Würfeln so lange, bis er weder auf einer Schlange noch auf einer Leiter landet.
4. Erweitern Sie Ihr Spiel nun um den zweiten Spieler.

Hinweis: Verwenden Sie die Klasse `Spielfeld`, um das Spielfeld zu zeichnen. Ersetzen Sie dazu `extends MiniJava` durch `extends Spielfeld`.

Nun steht Ihnen die Methode `void paintField(int x, int y)` zur Verfügung, mit der Sie sich ein Spielfeld mit den beiden Spielsteinen auf den Feldern mit den Nummern `x` und `y` anzeigen lassen können. Solange Sie nur einen Spielstein verwalten, können Sie für den zweiten die Position 0 verwenden.

Aufgabe 3.4 (H) Syntaxbaum [7 Punkte]

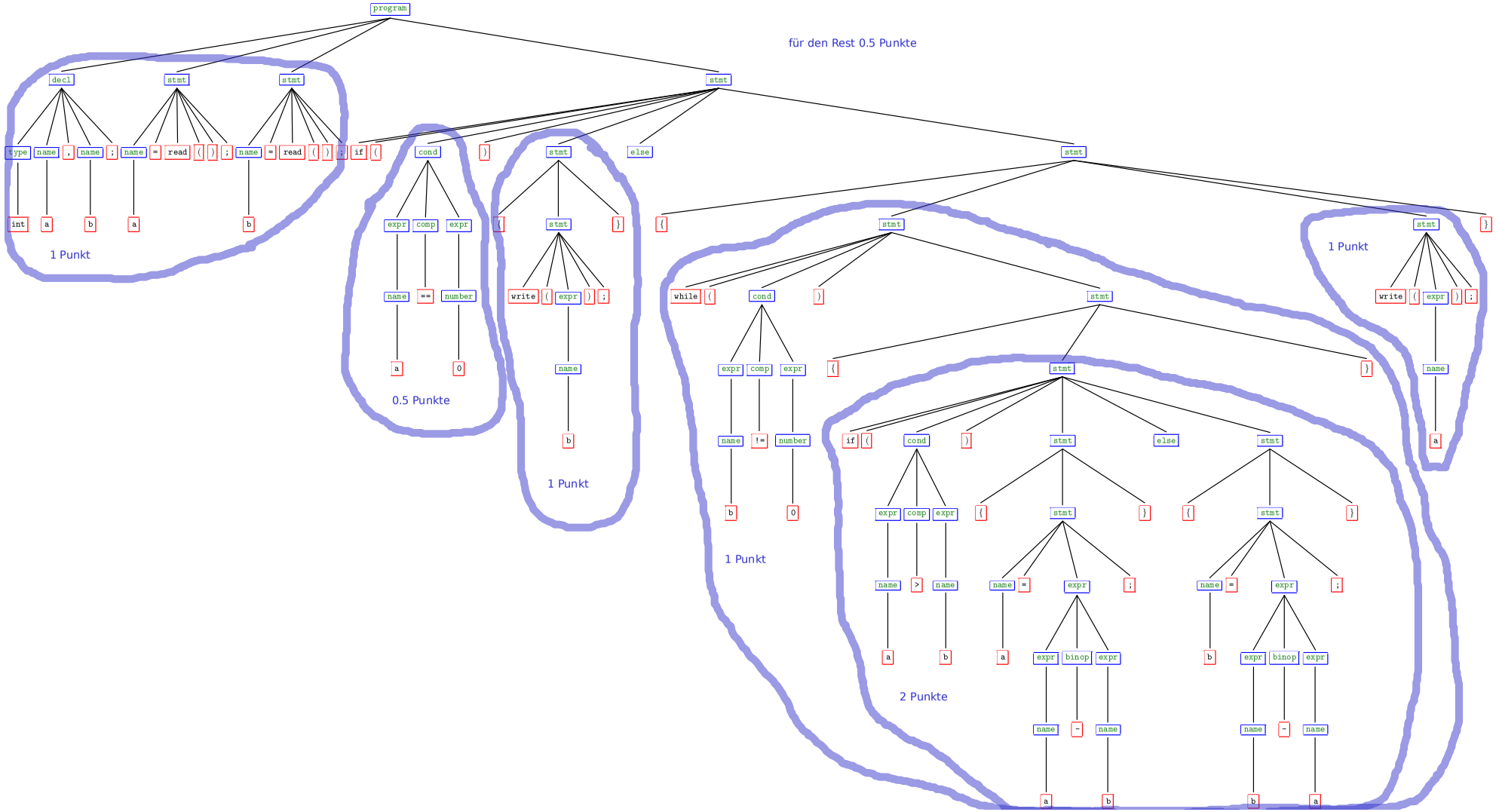
Zeichnen Sie für das folgende `MiniJava`-Programm den Syntaxbaum. Dazu steht Ihnen die Grammatik von `MiniJava` aus der Vorlesung zur Verfügung (eine Zusammenfassung finden Sie auch auf Moodle).

```
int a, b;

a = read();
b = read();

if (a == 0) {
    write(b);
} else {
    while(b != 0) {
        if (a > b) {
            a = a - b;
        } else {
            b = b - a;
        }
    }
    write(a);
}
```

Lösungsvorschlag 3.4


program
decl
stmt
stmt
stmt
type name , name ; name = read ( ) ; name = read ( ) ; if (cond) stmt else stmt
int a b a b
1 Punkt
expr comp expr
name == number
a 0
0.5 Punkte
{ stmt }
write (expr) ;
name
b
1 Punkt
für den Rest 0.5 Punkte
while (cond) stmt
expr comp expr
name != number
b 0
1 Punkt
{ stmt }
if (cond) stmt else stmt
expr comp expr
name > name
a b
{ stmt }
name = expr ;
a
expr binop expr
name - name
a b
{ stmt }
name = expr ;
b
expr binop expr
name - name
b a
2 Punkte
{ stmt }
write (expr) ;
name
a
1 Punkt

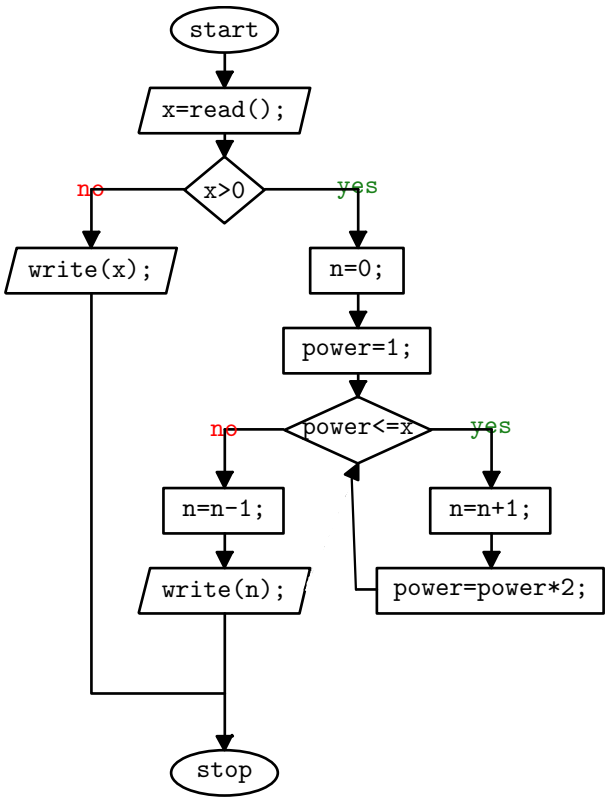
Aufgabe 3.5 (H) Kontrollflussdiagramm [5 Punkte]

Zeichnen Sie den Kontrollflussgraphen für folgendes `MiniJava`-Programm:

```
int x, n, power;
x = read();
if (x > 0) {
    n = 0;
    power = 1;
    while (power <= x) {
        n = n + 1;
        power = power * 2;
    }
    n = n - 1;
    write(n);
} else {
    write(x);
}
```

Hinweis: Zum Zeichnen können Sie z.B. das Programm *LibreOffice Draw* verwenden, welches Sie unter `http://www.libreoffice.org/` finden. Speichern Sie Ihren Kontrollflussgraphen im *pdf*-Format ab.

Lösungsvorschlag 3.5



Ein Semikolon nach einer Zuweisung oder nach einem `write`-Aufruf ist optional.

Korrekturbemerkung: ½ Punkt Abzug für jeden Fehler

Aufgabe 3.6 (H) Kartentricks [7 Punkte]

In dieser Aufgabe wollen wir das Kartenspiel *17 und 4* programmieren, das wie folgt funktioniert:

- Zwei Spieler spielen gegeneinander.
- Jede Spielkarte hat einen Wert zwischen 2 und 11 Punkten.
- Ziel des Spiels ist es, mit den eigenen Karten näher an 21 Punkte heranzukommen als der andere Spieler, ohne dabei den Wert von 21 Punkten zu überschreiten.
- Jeder Spieler hat am Anfang zwei Karten.
- Er kann entscheiden, ob er weitere Karten ziehen möchte oder nicht.
- Glaubt er, nahe genug an 21 Punkte herangekommen zu sein, so lehnt er weitere Karten ab.
- Wenn ein Spieler 22 oder mehr Punkte erreicht, verliert er sofort.
- Der zweite Spieler beginnt erst, wenn der erste Spieler sein Spiel beendet hat. (Es wird *nicht* abwechselnd gezogen!)
- Es gewinnt der Spieler, der am nächsten an 21 Punkte herankommt.
- Haben beide Spieler gleich viele Punkte, so gewinnt der erste Spieler.

Schreiben Sie ein MiniJava-Programm `SuV.java`, mit dem man *17 und 4* zu zweit spielen kann. Jeder Spieler soll über Dialogboxen gefragt werden, ob er weitere Karten ziehen will: 1 *für Ja,* 0 *für Nein.* Verwenden Sie dazu die statische Methode `int drawCard()` der Klasse `MiniJava`, um Karten zu ziehen.

Hinweis: Achten Sie darauf, Eingaben auf ihre Gültigkeit hin zu überprüfen!

Lösungsvorschlag 3.6

Die Lösung befindet sich in der Datei `SuV.java`

Korrekturbemerkung:

- Anfangszustand jeder Spieler hat zwei Karten: 1 Punkt
- Reihenfolge der Spieler: 1 Punkt
- Benutzerentscheidung Karte ziehen oder nicht: 1 Punkt
- Überprüfung der Benutzereingaben: 1 Punkt
- Abbruchbedingung, wenn ein Spieler 22 Punkte oder mehr hat: 1 Punkt
- Gewinnbedingung, falls beide Spieler gleich viele Punkte haben: 1 Punkt
- Gewinnbedingung, welcher Spieler näher an 21 Punkten ist, falls beide Spieler weniger als 22 Punkte haben: 1 Punkt