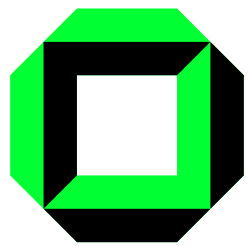
Info I – Übungsblatt 2

Joachim Breitner

mit einer Java-Aufgabe von Martin Kiefel

14. November 2005



Übungsblatt-Rückblick

- Erfreuliches Ergebnis:
  - Fast alle mehr als 50%
  - Schnitt 44 Punkte (81%)
- Häufigste Fehler
  - Simulation (laut Abeck) kann nicht Korrektheit überprüfen
  - Programmieren im Großen vs. im Kleinen

Tutoriums-Homepage

`http://www.joachim-breitner.de/wiki/Infotut`

- Folien der Tutorien
- Themen-Wunschliste
- Links
- Eventuell Rechnerübungs-Planung
- Es ist ein Wiki: Sei mutig!©

Regeln für Java-Programme

- Programmcode auf Englisch
- Kommentare auf Englisch
- Ein-/Ausgabe auf Deutsch
- Ausführlich kommentieren
- Keine Java-Klassen wie `Math` verwenden

Erinnerung: Praxisaufgaben-Abgabe

1. Programme schreiben und testen
2. Quelldateien (`*.java`) in einen Ordner „Nachname – Blatt 5“
3. Diesen Ordner in eine `.zip`- oder `.tar.gz`-Datei
4. Diese Datei an eine e-Mail an mich anhängen
   Adresse ist `mail@joachim-breitner.de`
5. Betreffzeile: „Tutorium 6 - Aufgabe 4.2“
6. Vor Freitag 13 Uhr losschicken

Auch andere e-Mails bitte mit Betreff á la
„Tutorium 6 - Mein großer Zeh juckt“

Rechnerübungs-Ablauf

1. Programme schreiben
2. Eventuell per e-Mail schicken
3. In die Rechnerübung kommen
4. Rechnerplatz bekommen (Ihr habt Vorrang!)
5. Mich auf euch aufmerksam machen
6. Programme vorführen
7. Fertig

Variablentypen und implizites Casting

Ganzzahlige Variablentypen

- `long`
- `int`
- `short`
- `byte`

Bei arithmetischer Auswertung wird grundsätzlich in `int` gecastet. Außer...

- es ist ein `long` in der Auswertung enthalten.
- man castet explizit in den gleichen oder „kleineren“ Typ.

Beispiele

Welche Typen erhalten wir nach der Auswertung? Welchen Wert haben sie? Könnte die Zuweisung schief gehen?

Vorgegeben sind

```
long l = 2;
int i = 5;
byte c = 3;
```

Zuweisungen (unabhängig)

```
i = l/c;
i = i*c;
c = c+1;
c = c++;
```

Beispiele

Welche Typen erhalten wir nach der Auswertung? Welchen Wert haben sie? Könnte die Zuweisung schief gehen?

Vorgegeben sind

```
long l = 2;
int i = 5;
byte c = 3;
```

Zuweisungen (unabhängig)

`i = l/c;` (0; Typ `long`, Casting nötig: `i = (int)(l/c);`!)

```
i = i*c;
c = c+1;
c = c++;
```

Beispiele

Welche Typen erhalten wir nach der Auswertung? Welchen Wert haben sie? Könnte die Zuweisung schief gehen?

Vorgegeben sind

```
long l = 2;
int i = 5;
byte c = 3;
```

Zuweisungen (unabhängig)

`i = l/c;` (0; Typ `long`, Casting nötig: `i = (int)(l/c);`!)

`i = i*c;` (15; Typ `int`)

```
c = c+1;
c = c++;
```

Beispiele

Welche Typen erhalten wir nach der Auswertung? Welchen Wert haben sie? Könnte die Zuweisung schief gehen?

Vorgegeben sind

```
long l = 2;
int i = 5;
byte c = 3;
```

Zuweisungen (unabhängig)

`i = l/c;` (0; Typ `long`, Casting nötig: `i = (int)(l/c);`!)

`i = i*c;` (15; Typ `int`)

`c = c+1;` (4; Typ `int`, Casting nötig: `c = (byte)(c+1);`!)

`c = c++;`

Beispiele

Welche Typen erhalten wir nach der Auswertung? Welchen Wert haben sie? Könnte die Zuweisung schief gehen?

Vorgegeben sind

```
long l = 2;
int i = 5;
byte c = 3;
```

Zuweisungen (unabhängig)

`i = l/c;` (0; Typ `long`, Casting nötig: `i = (int)(l/c);`!)
`i = i*c;` (15; Typ `int`)
`c = c+1;` (4; Typ `int`, Casting nötig: `c = (byte)(c+1);`!)
`c = c++;` (3; Typ `byte`)

weitere Beispiele

Vorgegeben sind

```
long a = 3;
int b = 4;
short c = 5;
byte d = 6;
```

Ausdrücke

```
d / b * a
c + b * (d+1)
d/(c-1) * b/2
d % b
-d % b
-d / b
c++ % d
```

weitere Beispiele

Vorgegeben sind

```
long a = 3;
int b = 4;
short c = 5;
byte d = 6;
```

Ausdrücke

```
d / b * a
```
(3; Typ `long`)

```
c + b * (d+1)
d/(c-1) * b/2
d % b
-d % b
-d / b
c++ % d
```

weitere Beispiele

Vorgegeben sind

```
long a = 3;
int b = 4;
short c = 5;
byte d = 6;
```

Ausdrücke

`d / b * a`	(3; Typ `long`)
`c + b * (d+1)`	(33; Typ `int`)
`d/(c-1) * b/2`	
`d % b`	
`-d % b`	
`-d / b`	
`c++ % d`	

weitere Beispiele

Vorgegeben sind

```
long a = 3;
int b = 4;
short c = 5;
byte d = 6;
```

Ausdrücke

`d / b * a`	(3; Typ `long`)
`c + b * (d+1)`	(33; Typ `int`)
`d/(c-1) * b/2`	(2; Typ `int`)
`d % b`	
`-d % b`	
`-d / b`	
`c++ % d`	

weitere Beispiele

Vorgegeben sind

```
long a = 3;
int b = 4;
short c = 5;
byte d = 6;
```

Ausdrücke

`d / b * a`	(3; Typ `long`)
`c + b * (d+1)`	(33; Typ `int`)
`d/(c-1) * b/2`	(2; Typ `int`)
`d % b`	(2; Typ `int`)
`-d % b`	
`-d / b`	
`c++ % d`	

weitere Beispiele

Vorgegeben sind

```
long a = 3;
int b = 4;
short c = 5;
byte d = 6;
```

Ausdrücke

`d / b * a`	(3; Typ `long`)
`c + b * (d+1)`	(33; Typ `int`)
`d/(c-1) * b/2`	(2; Typ `int`)
`d % b`	(2; Typ `int`)
`-d % b`	(-2; Typ `int`)
`-d / b`	(-1; Typ `int`)
`c++ % d`	

weitere Beispiele

Vorgegeben sind

```
long a = 3;
int b = 4;
short c = 5;
byte d = 6;
```

Ausdrücke

`d / b * a`	(3; Typ `long`)
`c + b * (d+1)`	(33; Typ `int`)
`d/(c-1) * b/2`	(2; Typ `int`)
`d % b`	(2; Typ `int`)
`-d % b`	(-2; Typ `int`)
`-d / b`	(-1; Typ `int`)
`c++ % d`	(5; Typ `int`)

Überläufe

Wenn man explizit castet, kann es zu unerwarteten Auswirkungen kommen: Wenn die Zahl nicht in den Typ passt, so wird „modulo“ weitergerechnet.

Beispiele

```
(byte)128          // -128
(byte)-129         // 127
(1000000000*10)    // 1410065408
```

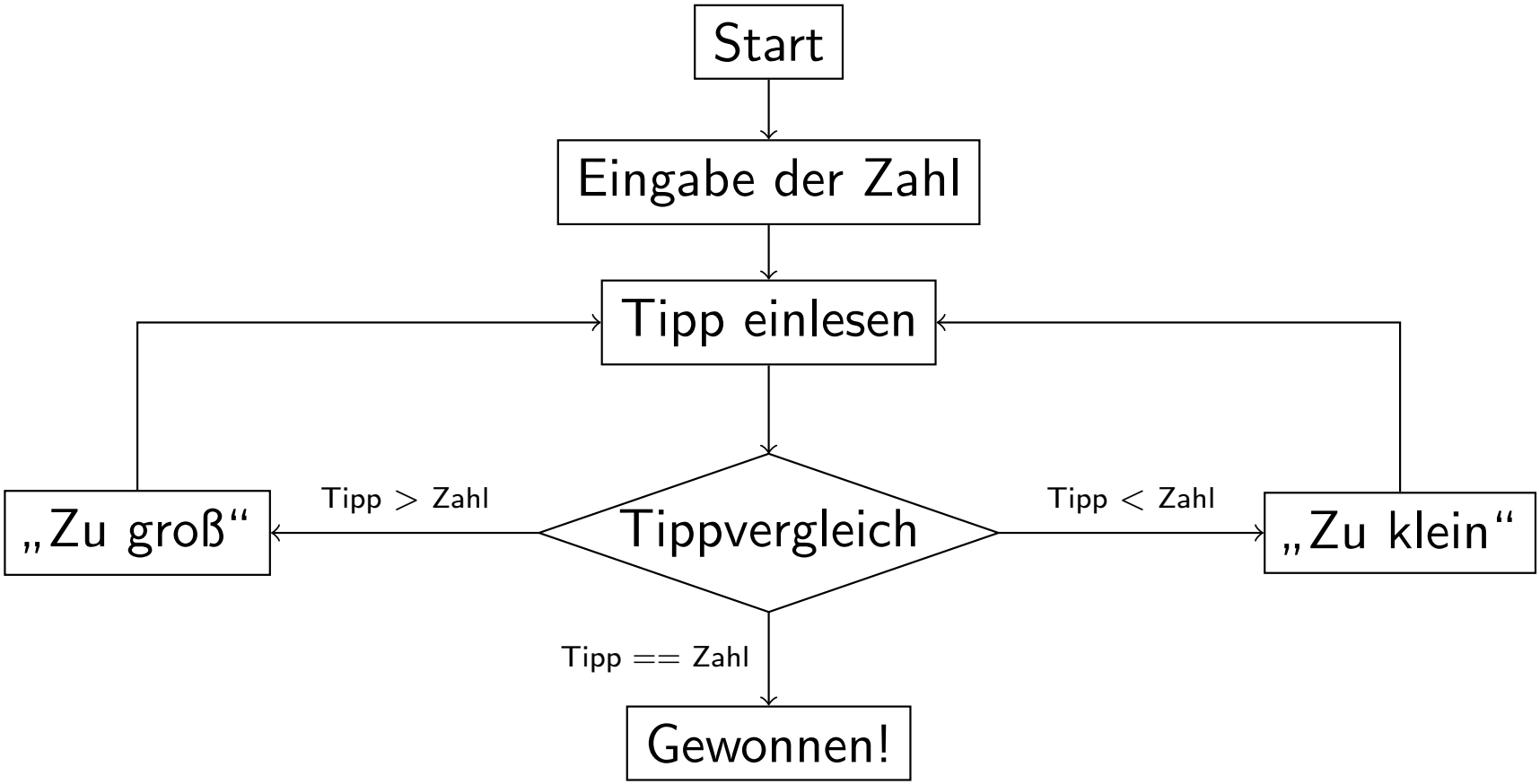
Kleines Javaprogramm „Zahlenraten“

Wir wollen gemeinsam ein kleines Spiel für Zwei in Java programmieren. Die Spielregeln sind wie folgt:

- Ein Spieler gibt verdeckt eine (natürliche) Zahl ein.
- Anschließend muss der andere Spieler solange die Zahl raten, bis er sie gefunden hat.
- Der Computer verrät ihm nach jedem Raten, ob die gesuchte Zahl „kleiner“ oder „größer“ ist.
- Das Spiel ist zu Ende, sobald die Zahl gefunden wurde.

Aufgabe: Erstellt ein Ablaufdiagramm! (jeder für sich)

Ablaufdiagramm



Das „Drum-Herum“ eines Javaprogramms

Der Rahmen eines Javaprogramm sieht *fast* immer so aus:

```
// Program Classname by Alan Turing. Does nothing.
public class Classname {
  public static void main(String[] args) {
    // Here, the program does all it does:

    // That was: ......Nothing
  }
}
```

Achtung, nicht vergessen: Der Name der Datei muss dem Namen der Klasse entsprechen! (Hier: `Classname.java`)

NumGuesser.java (1)

```
// NumGuesser by Tutorium 6.
// Plays with two humans or similar lifeforms
public class NumGuesser {
  // main function
  public static void main(String[] args) {

    // declare variables for target number and guess
    int number, guess;
```

NumGuesser.java (1)

```
// NumGuesser by Tutorium 6.
// Plays with two humans or similar lifeforms
public class NumGuesser {
  // main function
  public static void main(String[] args) {

    // declare variables for target number and guess
    int number, guess;
```

NumGuesser.java (2)

```
// read the secret number
Out.print("Bitte geheime Zahl eingeben: ");
number = In.readInt();

// make sure we really start the game
guess = number+1;

// keep playing while the number was not found
while (guess != number) {

  // ask player for his guess
  Out.print("Dein Tipp? ");
  guess = In.readInt();
```

NumGuesser.java (2)

```
// read the secret number
Out.print("Bitte geheime Zahl eingeben: ");
number = In.readInt();

// make sure we really start the game
guess = number+1;

// keep playing while the number was not found
while (guess != number) {

  // ask player for his guess
  Out.print("Dein Tipp? ");
  guess = In.readInt();
```

NumGuesser.java (2)

```
// read the secret number
Out.print("Bitte geheime Zahl eingeben: ");
number = In.readInt();

// make sure we really start the game
guess = number+1;

// keep playing while the number was not found
while (guess != number) {

  // ask player for his guess
  Out.print("Dein Tipp? ");
  guess = In.readInt();
```

NumGuesser.java (2)

```
// read the secret number
Out.print("Bitte geheime Zahl eingeben: ");
number = In.readInt();

// make sure we really start the game
guess = number+1;

// keep playing while the number was not found
while (guess != number) {

  // ask player for his guess
  Out.print("Dein Tipp? ");
  guess = In.readInt();
```

NumGuesser.java (3)

```
      // check if guess is too small or too big
      if (guess < number) {
        Out.println("groesser");
      } else if (guess > number) {
        Out.println("kleiner");
      }
    }

    // finished
    Out.println("Gewonnen");
  }
}
```

NumGuesser.java (3)

```
        // check if guess is too small or too big
        if (guess < number) {
          Out.println("groesser");
        } else if (guess > number) {
          Out.println("kleiner");
        }
      }

      // finished
      Out.println("Gewonnen");
    }
  }
```

Spielablauf als EBNF?

Aufgabe:

Kann man einen gültigen Spielablauf von „NumGuesser“ mit EBNF darstellen?

Probleme

- Unendliche Zahl der Zahlen $\Rightarrow$ Beschränkung auf $\{1, \ldots, 9\}$
- Immernoch große Zahl der Kombinationen $\Rightarrow$ Indizierung
- Achtung: Nicht mehr wirklich EBNF (Aber bei endlichen Indizes eindeutg übertragbar.)

Spielablauf als EBNF?

Aufgabe:

Kann man einen gültigen Spielablauf von „NumGuesser“ mit EBNF darstellen?

Probleme

- Unendliche Zahl der Zahlen $\Rightarrow$ Beschränkung auf $\{1, \ldots, 9\}$
- Immernoch große Zahl der Kombinationen $\Rightarrow$ Indizierung
- Achtung: Nicht mehr wirklich EBNF (Aber bei endlichen Indizes eindeutg übertragbar.)

Spielablauf als EBNF?

Aufgabe:

Kann man einen gültigen Spielablauf von „NumGuesser“ mit EBNF darstellen?

Probleme

- Unendliche Zahl der Zahlen $\Rightarrow$ Beschränkung auf $\{1, \ldots, 9\}$
- Immernoch große Zahl der Kombinationen $\Rightarrow$ Indizierung
- Achtung: Nicht mehr wirklich EBNF (Aber bei endlichen Indizes eindeutg übertragbar.)

Spielablauf als EBNF?

Aufgabe:

Kann man einen gültigen Spielablauf von „NumGuesser“ mit EBNF darstellen?

Probleme

- Unendliche Zahl der Zahlen $\Rightarrow$ Beschränkung auf $\{1, \ldots, 9\}$
- Immernoch große Zahl der Kombinationen $\Rightarrow$ Indizierung
- Achtung: Nicht mehr wirklich EBNF (Aber bei endlichen Indizes eindeutg übertragbar.)

Spielablauf als EBNF?

Aufgabe:

Kann man einen gültigen Spielablauf von „NumGuesser“ mit EBNF darstellen?

Probleme

- Unendliche Zahl der Zahlen $\Rightarrow$ Beschränkung auf $\{1, \ldots, 9\}$
- Immernoch große Zahl der Kombinationen $\Rightarrow$ Indizierung
- Achtung: Nicht mehr wirklich EBNF (Aber bei endlichen Indizes eindeutg übertragbar.)

Spielablauf als EBNF?

Aufgabe:

Kann man einen gültigen Spielablauf von „NumGuesser“ mit EBNF darstellen?

Probleme

- Unendliche Zahl der Zahlen $\Rightarrow$ Beschränkung auf $\{1, \ldots, 9\}$
- Immernoch große Zahl der Kombinationen $\Rightarrow$ Indizierung
- Achtung: Nicht mehr wirklich EBNF (Aber bei endlichen Indizes eindeutg übertragbar.)

Pseudo-EBNF

Syntax auf den Blättern und in der Klausur nicht gültig!

NumGuesser = SpielMit$_1$ | ... | SpielMit$_9$.

Daneben$_n$ = Kleiner$_n$ "Größer" | Groesser$_n$ "Kleiner".

SpielMit$_n$ = "Zahl?" n { "Tipp?" Daneben$_n$ } Ende$_n$.

Kleiner$_n$ = "1" | ... | $(n-1)$.

Größer$_n$ = $(n+1)$ | ... | "9".

Ende$_n$ = "Tipp?" n "Gewonnen!".

Beim Ausschreiben als EBNF beachten:

Kleiner$_1$ und Größer$_9$ dürften nicht hingeschrieben werden.
(Leere Regeln sind in der EBNF nicht erlaubt.)

Pseudo-EBNF

Syntax auf den Blättern und in der Klausur nicht gültig!

NumGuesser = SpielMit$_1$ | ... | SpielMit$_9$.

Daneben$_n$ = Kleiner$_n$ "Größer" | Groesser$_n$ "Kleiner".

SpielMit$_n$ = "Zahl?" n { "Tipp?" Daneben$_n$ } Ende$_n$.

Kleiner$_n$ = "1" | ... | $(n-1)$.

Größer$_n$ = $(n+1)$ | ... | "9".

Ende$_n$ = "Tipp?" n "Gewonnen!".

Beim Ausschreiben als EBNF beachten:

Kleiner$_1$ und Größer$_9$ dürften nicht hingeschrieben werden.
(Leere Regeln sind in der EBNF nicht erlaubt.)

Pseudo-EBNF

Syntax auf den Blättern und in der Klausur nicht gültig!

$$\texttt{NumGuesser} = \texttt{SpielMit}_1 \mid \ldots \mid \texttt{SpielMit}_9.$$

$$\texttt{Daneben}_n = \texttt{Kleiner}_n\ \texttt{"Größer"} \mid \texttt{Groesser}_n\ \texttt{"Kleiner"}.$$

$$\texttt{SpielMit}_n = \texttt{"Zahl?"}\ n\ \{\ \texttt{"Tipp?"}\ \texttt{Daneben}_n\ \}\ \texttt{Ende}_n.$$

$$\texttt{Kleiner}_n = \texttt{"1"} \mid \ldots \mid (n-1).$$

$$\texttt{Größer}_n = (n+1) \mid \ldots \mid \texttt{"9"}.$$

$$\texttt{Ende}_n = \texttt{"Tipp?"}\ n\ \texttt{"Gewonnen!"}.$$

Beim Ausschreiben als EBNF beachten:

$\texttt{Kleiner}_1$ und $\texttt{Größer}_9$ dürften nicht hingeschrieben werden.
(Leere Regeln sind in der EBNF nicht erlaubt.)

Pseudo-EBNF

Syntax auf den Blättern und in der Klausur nicht gültig!

$$\begin{aligned}
\text{NumGuesser} &= \texttt{SpielMit}_1 \mid \ldots \mid \texttt{SpielMit}_9.\\
\text{Daneben}_n &= \texttt{Kleiner}_n\ \texttt{"Größer"} \mid \texttt{Groesser}_n\ \texttt{"Kleiner"}.\\
\text{SpielMit}_n &= \texttt{"Zahl?"}\ n\ \{\ \texttt{"Tipp?"}\ \text{Daneben}_n\ \}\ \texttt{Ende}_n.\\
\text{Kleiner}_n &= \texttt{"1"} \mid \ldots \mid (n-1).\\
\text{Größer}_n &= (n+1) \mid \ldots \mid \texttt{"9"}.\\
\text{Ende}_n &= \texttt{"Tipp?"}\ n\ \texttt{"Gewonnen!"}.
\end{aligned}$$

Beim Ausschreiben als EBNF beachten:

$\texttt{Kleiner}_1$ und $\texttt{Größer}_9$ dürften nicht hingeschrieben werden.
(Leere Regeln sind in der EBNF nicht erlaubt.)

Pseudo-EBNF

Syntax auf den Blättern und in der Klausur nicht gültig!

$$\begin{aligned}
\text{NumGuesser} &= \texttt{SpielMit}_1 \mid \ldots \mid \texttt{SpielMit}_9.\\
\text{Daneben}_n &= \texttt{Kleiner}_n\ \texttt{"Größer"} \mid \texttt{Groesser}_n\ \texttt{"Kleiner"}.\\
\text{SpielMit}_n &= \texttt{"Zahl?"}\ n\ \{\ \texttt{"Tipp?"}\ \text{Daneben}_n\ \}\ \texttt{Ende}_n.\\
\text{Kleiner}_n &= \texttt{"1"} \mid \ldots \mid (n-1).\\
\text{Größer}_n &= (n+1) \mid \ldots \mid \texttt{"9"}.\\
\text{Ende}_n &= \texttt{"Tipp?"}\ n\ \texttt{"Gewonnen!"}.
\end{aligned}$$

Beim Ausschreiben als EBNF beachten:

$\texttt{Kleiner}_1$ und $\texttt{Größer}_9$ dürften nicht hingeschrieben werden.
(Leere Regeln sind in der EBNF nicht erlaubt.)

Pseudo-EBNF

Syntax auf den Blättern und in der Klausur nicht gültig!

$$\begin{aligned}
\text{NumGuesser} &= \texttt{SpielMit}_1 \mid \ldots \mid \texttt{SpielMit}_9.\\
\text{Daneben}_n &= \texttt{Kleiner}_n\ \texttt{"Größer"} \mid \texttt{Groesser}_n\ \texttt{"Kleiner"}.\\
\text{SpielMit}_n &= \texttt{"Zahl?"}\ n\ \{\ \texttt{"Tipp?"}\ \text{Daneben}_n\ \}\ \text{Ende}_n.\\
\text{Kleiner}_n &= \texttt{"1"} \mid \ldots \mid (n-1).\\
\text{Größer}_n &= (n+1) \mid \ldots \mid \texttt{"9"}.\\
\text{Ende}_n &= \texttt{"Tipp?"}\ n\ \texttt{"Gewonnen!"}.
\end{aligned}$$

Beim Ausschreiben als EBNF beachten:

$\texttt{Kleiner}_1$ und $\texttt{Größer}_9$ dürften nicht hingeschrieben werden.
(Leere Regeln sind in der EBNF nicht erlaubt.)

Pseudo-EBNF

Syntax auf den Blättern und in der Klausur nicht gültig!

$$\begin{aligned}
\text{NumGuesser} &= \texttt{SpielMit}_1 \mid \ldots \mid \texttt{SpielMit}_9.\\
\text{Daneben}_n &= \texttt{Kleiner}_n\ \texttt{"Größer"} \mid \texttt{Groesser}_n\ \texttt{"Kleiner"}.\\
\text{SpielMit}_n &= \texttt{"Zahl?"}\ n\ \{\ \texttt{"Tipp?"}\ \text{Daneben}_n\ \}\ \text{Ende}_n.\\
\text{Kleiner}_n &= \texttt{"1"} \mid \ldots \mid (n-1).\\
\text{Größer}_n &= (n+1) \mid \ldots \mid \texttt{"9"}.\\
\text{Ende}_n &= \texttt{"Tipp?"}\ n\ \texttt{"Gewonnen!"}.
\end{aligned}$$

Beim Ausschreiben als EBNF beachten:

$\texttt{Kleiner}_1$ und $\texttt{Größer}_9$ dürften nicht hingeschrieben werden.
(Leere Regeln sind in der EBNF nicht erlaubt.)

Pseudo-EBNF

Syntax auf den Blättern und in der Klausur nicht gültig!

$$\begin{aligned}
\text{NumGuesser} &= \texttt{SpielMit}_1 \mid \ldots \mid \texttt{SpielMit}_9.\\
\text{Daneben}_n &= \texttt{Kleiner}_n\ \texttt{"Größer"} \mid \texttt{Groesser}_n\ \texttt{"Kleiner"}.\\
\text{SpielMit}_n &= \texttt{"Zahl?"}\ n\ \{\ \texttt{"Tipp?"}\ \text{Daneben}_n\ \}\ \text{Ende}_n.\\
\text{Kleiner}_n &= \texttt{"1"} \mid \ldots \mid (n-1).\\
\text{Größer}_n &= (n+1) \mid \ldots \mid \texttt{"9"}.\\
\text{Ende}_n &= \texttt{"Tipp?"}\ n\ \texttt{"Gewonnen!"}.
\end{aligned}$$

Beim Ausschreiben als EBNF beachten:

$\texttt{Kleiner}_1$ und $\texttt{Größer}_9$ dürften nicht hingeschrieben werden.
(Leere Regeln sind in der EBNF nicht erlaubt.)

Hinweise zum 2. Übungsblatt

- Korrekte EBNF (z.B. der Punkt am Ende)
- Zuweisungen sind unabhängig
  - Im IPO neue Version des Übungsblattes
- Programme ausführlich kommentieren
- Konventionen beachten (Sprache, Variablenbenennung)
- Gefundene Fehler begründen

Erklärung Brainfuck-HelloWorld

Brainfuck-Kommandos und ASCII-Codes

<>	benachbarte Speicherzelle wählen	¶	10	e	101
.,	Speicherzelle ausgeben/einlesen	„ “	32	l	108
[]	Schleifen, bis Speicherzelle 0	,	44	o	111
+-	Speicherwert um 1 erhöhen/verringern	H	72	r	114
		d	100	w	119

HelloWorld.b

```
++++++++++[>++++>++++++++++>+++++++<<<-]
>>>++.<+.+++++++..+++.<++++.<+++[>----<-]>.
>++++++++.--------.+++.------.--------.
<<+++[>++++<-]>++.<++++++++++.
```

Fragen

Fragen?