Javakurs
SS 2006, Freitagsrunde und Tutoren

Aufgabenblatt 2

letzte Aktualisierung: 26. März, 12:39

Ausgabe: 03.04.2006

Thema: *Methoden und Schleifen*

Lernziele:

- *Methoden und Schleifen verstehen*
- *Arrays benutzen*

Hinweis: Die Vorgaben zu den einzelnen Aufgaben findet ihr unter:

`~seclab/javakurs/le2`

1. Aufgabe: Sprachliche Konstrukte (arrays, while)

Analysiere die gegebenen Methoden einzeln und finde für jede heraus, wo das Problem liegt. Dafür kannst Du eigene Ausgaben einbauen. Um die Methoden einzeln testen zu können, kommentiere jeweils die Methodenaufrufe aus der `main`-Methode aus, die du nicht testen willst.

Beispiellösung:

```
1  static void test00() {
2          // Problem: inta ist null,
3          // daher ist jeglicher Zugriff toedlich
4          int[] inta = null;
5          System.out.println(inta.length);
6          System.out.println(inta[0]);
7          System.out.println(inta[-1]);
8  }
9
10 static void test01() {
11         // Problem: der Index -1 ist nicht zulaessig
12         int[] inta = new int[10];
13         System.out.println(inta.length);
14         System.out.println(inta[0]);
15         System.out.println(inta[-1]);
16 }
17
18 static void test02() {
19         // Problem: Schleifenzaehlvariable wird
20         // nicht hochgezaehlt => Endlosschleife
21         int[] inta = new int[10];
22         int i = 0;
23         while (i < 10)
24                 System.out.println(inta[i]);
25 }
26
```



```
27 static void test03() {
28         // Problem: Schleifenzaehlvariable ist groesser
29         // als Vergleichsindex => Endlosschleife?
30         int[] inta = new int[10];
31         int i = 11;
32         while (i != 10) {
33                 i++;
34                 System.out.println(inta[i]);
35         }
36 }
37
38 static void test04() {
39         // Problem: inta[10] ist nicht definiert!
40         // Letzter Index ist 9.
41         int[] inta = new int[10];
42         int i = 0;
43         while (i <= 10) {
44                 System.out.println(inta[i]);
45                 i++;
46         }
47 }
48
49 static void test05() {
50         // Problem: einige wurden vergessen (indizes 10..19)
51         int[] inta = new int[20];
52         int i = 0;
53         while (i < 10) {
54                 System.out.println(inta[i]);
55                 i++;
56         }
57 }
58
59 static void test06() {
60         // Problem: inta ist nicht initialisiert,
61         // normalerweise heisst das 0,
62         // eigene initialisierung ist dennoch angeraten!
63         int[] inta = new int[20];
64         int i = 0;
65         while (i < inta.length) {
66                 System.out.println(inta[i]);
67                 i++;
68         }
69 }
70
71 static void test07() {
72         // Problem: Schleifenvariable wurde nicht
73         // zurueck auf 0 gesetzt!
74         int[] inta = new int[20];
75         int i = 0;
76         while (i < inta.length) {
77                 inta[i] = i * i;
78                 i++;
79         }
80         while (i < inta.length) {
```

```
                inta[i] = i * i;
                System.out.println(inta[i]);
                i++;
        }
}

static void test08() {
        // Problem: ein Array kann man so leider nicht
        // ausgeben: Ausgabe ist "nur" die Referenz
        int[] inta = new int[20];
        int i = 0;
        while (i < inta.length) {
                inta[i] = i * i;
                i++;
        }
        System.out.println(inta);
}

static void test09() {
        // Problem: i wird nicht erhoeht
        int[] inta = new int[20];
        for (int i = 0; i < 10;)
                inta[i] = i * i;
}

static void test10() {
        // Problem: nicht alle Werte werden initialisiert
        int[] inta = new int[20];
        for (int v = 0; v < 10; v++)
                inta[v] = v - v * v;
}
```


2. Aufgabe: Die Fibonacci-Funktion

$$fib(x) := \begin{cases} x & \text{falls } x < 2 \\ fib(x-1) + fib(x-2) & \text{sonst} \end{cases}$$

$$fib := 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ..$$

Opal-Code:

```
FUN fib : nat -> nat
DEF fib(n) ==
   IF n<2
   THEN n
   ELSE fib(n-1)+fib(n-2)
   FI
```

2.1. Simple Implementierung in Java Implementiere die Fibonacci-Funktion in JAVA. Orientiere Dich dabei möglichst stark am Opal–Code bzw. der mathematischen Definition.

Beispiellösung:


```
/** Returns the <code>x</code>'s fibonacci number. */
static int fib(int x) {
        if (x < 2)
                return x;
        return fib(x - 1) + fib(x - 2);
}
```


2.2. Effizientere Implementierung in Java Ändere Deine Implementierung und passe diese an die folgende (effizientere) Opal–Implementierung an.

```
FUN fib: nat -> nat
DEF fib(x) ==
   IF x<2
   THEN x
   ELSE fibAcc(2, 1, 1, x)
   FI

FUN fibAcc: nat ** nat ** nat ** nat -> nat
-- idx : momentaner Index
-- p1  : fib(idx)
-- p2  : fib(idx-1)
-- max : der Index des zu berechnenden Folgengliedes
DEF fibAcc(idx, p1, p2, max) ==
   IF idx<max THEN fibAcc(idx+1, p1+p2, p1, max)
   IF idx=max THEN p1
   FI
```

Beispiellösung:

```
/** Returns the <code>x</code>'s fibonacci number. */
static int fib(int x) {
        if (x < 2)
                return x;
        return fibAcc(2, 1, 1, x);
}

/**
        * This method supports {@link #fib}.
        *
        * @param idx
        *           the current index.
        * @param p1
        *           fib(idx)
        * @param p2
        *           fib(idx-1)
        * @param max
        *           index of the fibonacci number to be calculated.
        * @return the <code>max</code>'s fibonacci number.
        */
static int fibAcc(int idx, int p1, int p2, int max) {
        if (idx < max)
                return fibAcc(idx + 1, p1 + p2, p1, max);
```

```
        return p1;
}
```

2.3. Implementierung in Java Verwende anstatt einer Rekursion eine Iteration.

Beispiellösung:

```
/** Returns the <code>x</code>'s fibonacci number. */
static int fib(int x) {
        if (x < 2)
                return x;
        int p1 = 1, p2 = 1, idx = 2;
        int tmp;
        while (idx < x) {
                idx++;
                tmp = p2;
                p2 = p1;
                p1 += tmp;
        }
        return p1;
}
```

2.4. Mehrere Fibonacci-Zahlen in einem Array Der Anwender möchte nun die ersten n Fibonacci-Zahlen berechnen. Finde eine effiziente Implementierung mit folgender Signatur: `static int[] fib(int max)`

Beispiellösung:

```
/**
        * This is the calculating method.
        *
        * @param max
        *           the index of the last element of the
        *           fibonacci numbers to be calculated.
        * @return an array containing the elements of the
        *         fibonacci numbers from index 0 to max.
        */
static int[] fib(int max) {
        int[] result = new int[max];
        if (result.length > 0)
                result[0] = 0;
        if (result.length > 1)
                result[1] = 1;
        for (int cur = 2; cur < result.length; cur++)
                result[cur] = result[cur - 1] + result[cur - 2];
        return result;
}
```

3. Aufgabe: Primzahlen

3.1. Primzahlprüfung Die Methode `isPrime` soll überprüfen ob die übergebene Zahl i eine Primzahl ist.

$$i\ Prim \ \Leftrightarrow\ i \in \mathbb{N} \wedge i > 1 \wedge \forall j \in \{2, .., i-1\} : i \not\equiv 0 \ mod\ j$$

Primzahlen sind also: $2, 3, 5, 7, 11, 13, 17, 19, 23, ..$

Beispiellösung:

```
/**
* This method checks wether a number is prime or not.
*
* @param i
*          the number to check.
* @return <code>true</code> exactly if the number <emph>IS </emph> prime.
*/
static boolean isPrime(int i) {
    //********************** LOESUNG ************************
    if (i < 2) {
        return false;
    }
    // it is possible to make this funktion faster, since we just need to
    // check {2, .., floor(sqrt(i))}.
    for (int j = 2; j < i; j++) {
        if (i % j == 0) {
            return false;
        }
    }
    return true;
    //********************** LOESUNG(ENDE) ******************
}
```

3.2. Methodenuntersuchung Untersuche die Methode `findPrimes`.

- Was tut sie (genau)?
- Bei welchen Eingaben tauchen Fehler auf?
- Wieso ist der Code nicht schön?
- Wo verliert die Funktion Zeit?
- Was könnte man verbessern?

Baue für Deine Untersuchung zusätzliche Ausgaben in der Funktion `findPrimes` ein und analysiere die Ausgaben der Funktion.

```
static int[] findPrimes(int n) {
   int[] res = new int[(n + 1) / 2];
   int lastIdx = 0;
   for (int i = 1; i <= n; i++)
      if (isPrime(i)) {
         res[lastIdx] = i;
         lastIdx++;
      }
   int[] realRes = new int[lastIdx];
   for (int i = 0; i < lastIdx; i++)
      realRes[i] = res[i];
   return realRes;
}
```

Beispiellösung:

```
tobo added
```

4. Aufgabe: Glücksspiel (Zusatzaufgabe)

Bei diesem Spiel gibt es *37* Felder, auf die der Spieler *200*€ (in Einheiten von 1€) verteilt. Danach wird eine Zufallszahl n ermittelt und an den Spieler wird der Gewinn ausgezahlt. Die Gewinnsumme errechnet sich wie folgt:

$$\sum_{i=0}^{36} (\texttt{board}[i] \cdot (6 - \mid i - n \mid))$$

wobei `board`$[i]$ den Wert der Belegung des Feldes i darstellt.
Allerdings sollen nur positive Summanden in die Summe eingehen.

4.1. Berechnung der Gewinnsumme Vervollständige die `play`-Funktion, welche bei der Eingabe eines *boards* die Gewinnsumme errechnet. Die Funktion soll also die Aufgabe eines Spielleiters bei einem Roulettespiel einnehmen.

Hinweis: Es wird sinnvoll sein, eine Methoden `static int abs(int x)` und eine Methode `static int max(int x, int y)` zu implementieren. `abs()` liefert dabei den Betrag der übergebenen Zahl zurück, während `max()` den größeren der beiden übergebenen Werte zurückliefert.

Beispiellösung:

```
static int play(int[] boardSet) {
        numb = getRand(37);
        //**************** LOESUNG *******************
        // res stores the brutto prize
        int res = 0, weight;
        // it is only necessary to sum over:
        // i in {min{numb-5, 0}, max{numb+5, boardSet.length}}
        // this would be a possible optimization
        for (int i = 0; i < boardSet.length; i++) {
                // weight is somewhat needless but this way
                // its hopefully easier to read
                weight = Math.max(-Math.abs(i - numb) + 6, 0);
                res += boardSet[i] * weight;
        }
        return res;
        //**************** LOESUNG(ENDE) *************
}

static int abs(int x) {
        if (x < 0) {
                return -x;
        } else {
                return x;
        }
}

static int max(int x, int y) {
        if (x > y) {
                return x;
        } else {
                return y;
        }
}
```


4.2. Ideale Boardbelegungen Erstelle ein paar Board-Belegungen und versuche Deinen Gewinn zu maximieren. Hierbei ist es nützlich, die Gewinnfunktion zu verstehen. Mache Dir also klar unter welchen Umständen man wieviel gewinnt.

- Gibt es Zahlen die günstiger sind als andere?
- Lohnt es sich auf einzelne Felder besonders viel zu setzen?
- Ist es besser den Einsatz gleichmäßig zu verteilen?

Versuche eine möglichst ideale Board-Belegung zu finden, indem du viele (z.B. 100 000) Spiele mit Deiner festen Belegung simulierst. Falls Du magst, kannst Du auch andere Gewinnfunktionen ausprobieren. z.B. $*36$ bei einem Volltreffer und $*6$ falls eines der acht benachbarten Felder getroffen wurde. Die Praktikabilität einer neuen Gewinnfunktion kannst Du mit der Lösung aus Aufgabe 3.1 überprüfen. Der durchschnittliche Gewinn sollte den Einsatz nicht überschreiten!

Beispiellösung:

```
public static void main(String[] args) {
        // place the bet
        int[] boardSet = placeBet(3);
        System.out.println("You␣won:␣" + play(boardSet));

        System.out.println("\nSimulating␣100000␣games:");
        //**************** LOESUNG ******************
        // the average brutto prize
        // (the avg is built at the end!)
        int mean = 0;
        // the maximal brutto prize
        int max = Integer.MIN_VALUE;
        // the minimal brutto prize
        int min = Integer.MAX_VALUE;
        // the current brutto prize (always >=0)
        int curBrutto;
        for (int i = 0; i < 100000; i++) {
                curBrutto = play(boardSet);
                if (curBrutto < min)
                        min = curBrutto;
                if (curBrutto > max)
                        max = curBrutto;
                mean += curBrutto;
        }
        System.out.println("mean" + mean / 100000);
        System.out.println("min" + min);
        System.out.println("max" + max);
        //**************** LOESUNG(ENDE) *************
}

/**
        * This method creates some board-sets.
        *
        * @param mode
        *            the method creates different boards for 0 to 3.
        * @return the board-set related to <code>mode</code>.
        */
static int[] placeBet(int mode) {
```

```
        int[] boardSet = new int[37];
        if (mode == 0) {
                // distribute the money over all slots equally
                int rem = 200;
                for (int i = 0; rem > 0; i++) {
                        boardSet[i % 37]++;
                        rem--;
                }
        } else if (mode == 1) {
                boardSet[0] = 20;
                boardSet[5] = 30;
                boardSet[10] = 20;
                boardSet[15] = 50;
                boardSet[20] = 15;
                boardSet[25] = 20;
                boardSet[30] = 22;
                boardSet[35] = 20;
        } else if (mode == 2) {
                //**************** LOESUNG *******************
                // place bets on some fewer slots
                for (int rem = 200; rem > 0; rem -= 10)
                        boardSet[getRand(37)] += 10;
                //**************** LOESUNG(ENDE) *************
        } else if (mode == 3) {
                //**************** LOESUNG *******************
                // place all the money on a single slot
                // not at the border of the interval
                boardSet[20] = 200;
                //**************** LOESUNG(ENDE) *************
        } else
                System.out.println("invalid␣mode");
        return boardSet;
}
```

5. Aufgabe: Endlicher Automat (Zusatzaufgabe)

Definition Wort: Eine geordnete, nichtnegative Anzahl von Terminalsymbolen (auch Symbol oder Zeichen genannt). Definition Sprache: Menge von Worten (auch String oder Zeichenkette genannt). Eine Sprache kann unendlich groß sein. Trotzdem lassen sich einige Sprachen schematisch darstellen. $\underline{a}^\star$ bedeutet, dass das Terminalsymbol beliebig häufig (0,1,2. . . mal) vorkommt. Die Sprache dieser Aufgabe enthält also z.B. die Worte: (), (a), (bc), (d), (aabcd), (bcbcbcd), (aaddd). Nicht enthalten sind z.B. die Worte: λ (das leere Wort), (ab, (ab), (ada), (abd). Die Aufgabe ist nun, zu entscheiden, ob ein eingegebenes Wort (String) in dieser Sprache enthalten ist. Das Ergebnis ist also entweder: Ja, das eingebene Wort ist in der Sprache, oder: Nein, das eingegebene Wort ist nicht in der Sprache.

Eine Grammatik ist eine Vorschrift, wie man Worte einer Sprache erzeugen kann. Außerdem sind alle Worte, die sich nicht durch die Grammatik ableiten lassen, nicht in der Sprache enthalten. Mit der gegebenen Grammatik lässt sich folgendermaßen ein Wort ableiten:

$$L = \left\{ \underline{(}\ \underline{a}^\star\ (\underline{b}\ \underline{c})^\star\ \underline{d}^\star\ \underline{)} \right\}$$

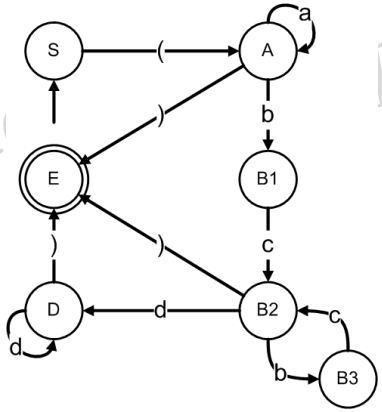

Abbildung 1: Ein Automat zu der Grammatik aus Aufgabe 4

Ableitung			Grammatik		
S	$\rightarrow$	$(A$	S	$\rightarrow$	$\underline{(}\ A$
	$\rightarrow$	$(aA$	A	$\rightarrow$	$\underline{a}\ A$
	$\rightarrow$	$(aB$		$\mid$	B
	$\rightarrow$	$(abcB$	B	$\rightarrow$	$\underline{b}\ \underline{c}\ B$
	$\rightarrow$	$(abcbcB$		$\mid$	D
	$\rightarrow$	$(abcbcD$	D	$\rightarrow$	$\underline{d}\ D$
	$\rightarrow$	$(abcbc)$		$\mid$	$\underline{)}$

5.1. Entscheider Gegegeben ist folgende Sprache L mit der o.g. Grammatik. Jedes Terminalsymbol ist ein `char`. Terminale sind unterstrichen, Nonterminale sind nicht unterstrichen.

Schreibe ein Programm, das für einen gegebenen String entscheidet, ob dieser String in der Sprache enthalten ist.

Beispiellösung:

```
public class Aufgabe5 {

/**
* Main-Method.
*
* @param args
*            not used.
*/
public static void main(String[] args) {
        System.out.println(parseBegin(new StringBuffer("(abcbcd)")));
        //********************** LOESUNG *************************
        System.out.println(parseBegin(
                new StringBuffer("(bcbcd)")));
        System.out.println(parseBegin(
                new StringBuffer("(ad)")));
        System.out.println(parseBegin(
                new StringBuffer("(abcbc)")));
        System.out.println(parseBegin(
```

```
                new StringBuffer("a)")) == false);
        System.out.println(parseBegin(
                new StringBuffer("(a")) == false);
        System.out.println(parseBegin(
                new StringBuffer("(")) == false);
        System.out.println(parseBegin(
                new StringBuffer("(abcbd)")) == false);
        System.out.println(parseBegin(
                new StringBuffer("(bcbcda)")) == false);
        //********************** LOESUNG(ENDE) *******************
}

static boolean parseAs(StringBuffer str) {
        if (str.length() > 0 && str.charAt(0) == 'a')
                return parseAs(str.deleteCharAt(0));
        return parseB(str);
}

static boolean parseB(StringBuffer str) {
        if (str.length() > 0 && str.charAt(0) == 'b')
                return parseC(str.deleteCharAt(0));
        return parseD(str);
}

static boolean parseBegin(StringBuffer str) {
        if (str.length() > 0 && str.charAt(0) == '(')
                return parseAs(str.deleteCharAt(0));
        return false;
}

static boolean parseC(StringBuffer str) {
        //********************** LOESUNG *************************
        if (str.length() > 0 && str.charAt(0) == 'c')
                if (str.substring(1).charAt(0) == 'b')
                        return parseB(str.deleteCharAt(0));
                else
                        return parseD(str.deleteCharAt(0));
        //********************** LOESUNG(ENDE) *******************
        return false;
}

static boolean parseD(StringBuffer str) {
        //********************** LOESUNG *************************
        if (str.length() > 0 && str.charAt(0) == 'd')
                return parseD(str.deleteCharAt(0));
        if (str.length() > 0 && str.charAt(0) == ')')
                return parseEnd(str);
        //********************** LOESUNG(ENDE) *******************
        return false;
}

static boolean parseEnd(StringBuffer str) {
        return str.length() == 1;
}
```



```

}
```

6. Aufgabe: Approximationen (Zusatzaufgabe)

6.1. Approximation von π Approximiere π mit der Formel von Ramanujan. Zerlege hierfür die gegebene Formel in die markierten Teilausdrücke und speichere die Zwischenergebnisse in temporären Variablen.

$$\pi \approx \underbrace{\frac{4}{\sqrt{522}}}\ln\underbrace{\left[\underbrace{\left(\frac{5+\sqrt{29}}{\sqrt{2}}\right)}^{3}\underbrace{\left(5\sqrt{29}+11\sqrt{6}\right)}\underbrace{\left(\underbrace{\sqrt{\frac{9+3\sqrt{6}}{4}}}+\underbrace{\sqrt{\frac{5+3\sqrt{6}}{4}}}\right)}^{6}\right]}$$

Beispiellösung:

```
/**
* This method approximates pi using the methos pow and sqrt from this
* class.
*
* @return an approximated value for pi.
*/
static double pi() {
        //********************** LOESUNG ************************
        double fac1 = 4 / sqrt(522);
        double fac2 = pow((5 + sqrt(29)) / sqrt(2), 3);
        double fac3 = 5 * sqrt(29) + 11 * sqrt(6);
        double sum1 = sqrt((9 + 3 * sqrt(6)) / 4);
        double sum2 = sqrt((5 + 3 * sqrt(6)) / 4);
        double fac4 = pow((sum1 + sum2), 6);
        double lnval = Math.log(fac2 * fac3 * fac4);
        return lnval * fac1;
        //********************** LOESUNG(ENDE) *******************
}
```

6.2. Approximiere e Entwickle eine iterative und eine rekusive Implementierung und vergleiche die Ergebnisse. Überlege Dir gegen welchen Wert .. konvergiert und setze diesen ein.

$$e-2 \approx \frac{1+\frac{1+\frac{1+\frac{\ddot{}}{6}}{5}}{4}}{3}{}^{2} = \frac{1}{2}(1+\frac{1}{3}(1+\frac{1}{4}(1+\frac{1}{5}(1+\frac{1}{6}(1+..)))))$$

Beispiellösung:

```
/**
* This method approximates the euler-number recursively.
*
* @param a
*           start-value, should be 2 initially
* @param b
*           start-value, the higher b the better the approximiation
* @return an approximation for e
*/
```

```
static double euler(int a, int b) {
        //********************** LOESUNG *************************
        if (a > b)
                return 0;
        else
                return (1 + euler(a + 1, b)) / a;
        //********************** LOESUNG(ENDE) *******************
}

/**
* This method approximates the euler-number iteratively.
*
* @param a
*           start-value, should be 2 initially
* @param b
*           start-value, the higher b the better the approximiation
* @return an approximation for e
*/
static double euler_iterativ(int a, int b) {
        //********************** LOESUNG *************************
        double res = 0;
        while (b >= a) {
                res = (1 + res) / b;
                b--;
        }
        return res;
        //********************** LOESUNG(ENDE) *******************
}
```