17. Schrittweise Verfeinerung

Entwurfsmethode für Algorithmen

Wie kommt man von der Aufgabenstellung zum Programm?

Beispiel

geg.: *Text aus Wörtern*
ges.: *Wie oft kommt jedes Wort im Text vor?*

Welche Befehle würde man sich wünschen, um diese Aufgabe zu lösen?

- Lies Wort
- Zähle Wort (mittels Tabelle)
- Drucke Zähler

Leider gibt es keine Sprache mit diesen Befehlen
Þ Befehle als Methoden implementieren (d.h. gewünschte "Sprache" selbst bauen)

Vorgehensweise

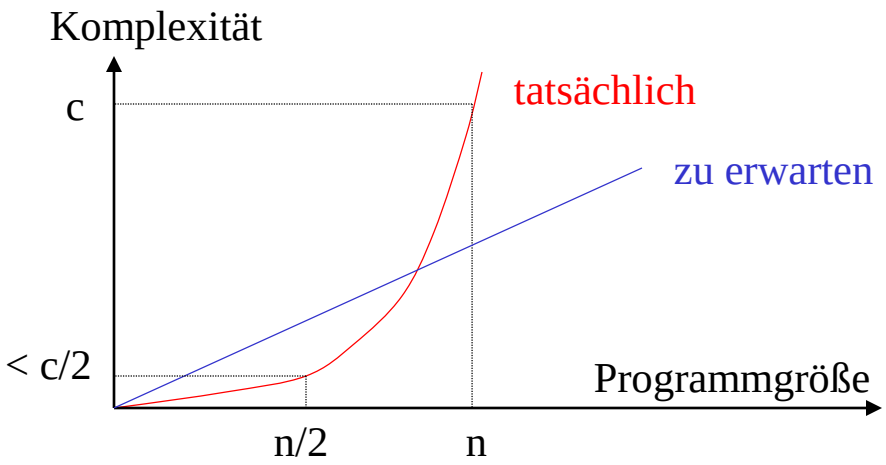
Schrittweise Verfeinerung

1. Zerlege Aufgabe in Teilaufgaben und spezifiziere deren Schnittstelle
2. Nimm an, dass die Teilaufgaben schon gelöst sind
   Implementiere Gesamtaufgabe mit Hilfe der Teillösungen
3. Sind die Teilaufgaben einfach genug?

   ja => implementiere sie direkt in einer Programmiersprache

   nein => Zerlege sie weiter (Schritt 1)

Was bringt das?



Komplexität steigt überproportional mit der Programmgröße

Halbierung der Programmgröße
reduziert die Komplexität
um mehr als die Hälfte!

17. Schrittweise Verfeinerung

Beispiel: Wortzählung

Aufgabe: "*Zähle die Häufigkeit von Wörtern in einem Text*"

1. **Zerlege Aufgabe in Teilaufgaben und spezifiziere deren Schnittstelle**

- Lies Wort	word = readWord();	liefert nächstes Wort oder *null*, wenn kein Wort mehr gelesen werden kann

Klasse *WordTable* mit folgenden zwei Methoden:

- Zähle Wort	tab.count(word);	- *word* Ï *tab* : *word* mit Häufigkeit 1 eintragen - *word* Î *tab* : Häufigkeit um 1 erhöhen
- Drucke Zähler	tab.print();	gibt Wörter und ihre Häufigkeiten aus

2. **Nimm an, dass die Teilaufgaben schon gelöst sind**

Beispiel (Forts.)

Implementiere Gesamtaufgabe mit Hilfe der Teillösungen

```
class WordCount {
    public static void main (String[] arg) {
        WordTable tab = new WordTable();
        In.open("input.txt");
        String word = readWord();
        while (word != null) {
            tab.count(word);
            word = readWord();
        }
        In.close();
        tab.print();
    }
}
```

Was haben wir bisher geleistet?

Sehr viel! Wir haben die komplexe Aufgabe *WordCount* auf relativ einfache Teilaufgaben wie *readWord()* oder *tab.count(word)* reduziert.

Beispiel (Forts.)

3. Zerlege Teilaufgaben weiter (*readWord*)

- Lies Zeichen — `ch = In.read();` — *In.done* == false, wenn Dateiende
- Prüfe, ob Buchstabe — `Character.isLetter(ch)`
- Füge Zeichen an Wort an — `word.append(ch);`

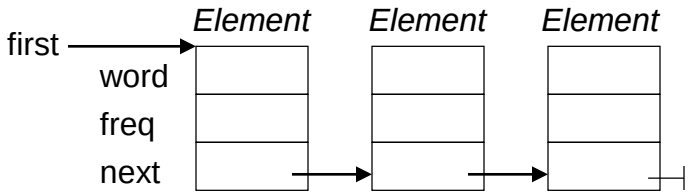
Alle Teilaufgaben sind bereits in der Java-Bibliothek implementiert.
Implementiere *readWord*() mit ihnen

```
static String readWord () {
   char ch;
   //----- skip nonletters
   do ch = In.read(); while (In.done() && ! Character.isLetter(ch));
   // !In.done() || ch is a letter
   //----- build the word
   StringBuilder word = new StringBuilder();
   while (In.done() && Character.isLetter(ch)) {
      word.append(ch);
      ch = In.read();
   }
   // ! In.done() || ch is not a letter
   if (word.length > 0) return word.toString(); else return null;
}
```

Beispiel (Forts.)

Grobstruktur der Worttabelle

Datenstruktur:
verkettete Liste von Wörtern
und Häufigkeiten




```
class Element {
    String word;
    int freq;
    Element next;

    Element (String w) {
        word = w; freq = 1;
    }
}
```

```
class WordTable {
    Element first = null;

    void count (String word) {...}
    void print () {...}
}
```

Beispiel (Forts.)

4. Zerlege Teilaufgaben weiter (*tab.count(word)*)

- Suche *word* in *tab* — `elem = tab.find(word);` — liefere *null*, wenn nicht gefunden
- Trage *word* in *tab* ein — `tab.enter(word);`
- Erhöhe Worthäufigkeit — `elem.freq++;`

```
void count (String word) {
   Element e = tab.find(word);
   if (e == null) tab.enter(word); else e.freq++;
}
```

find und *enter* sind so einfach, dass man sie sofort implementieren kann

```
Element find (String word) {
   Element e = first;
   while (e != null && !word.equals(e.word)) {
      e = e.next;
   }
   // e == null || word.equals(e.word)
   return e;
}
```

```
void enter (String word) {
   Element e = new Element(word);
   e.next = first;
   first = e;
}
```

Beispiel (Forts.)

4. **Zerlege Teilaufgaben weiter** (*tab.print()*)

   Ist so einfach, dass man es sofort implementieren kann

```
void print () {
    for (Element e = first; e != null; e = e.next)
        Out.println(e.word + ": " + e.freq);
}
```

Zusammensetzen der einzelnen Teile

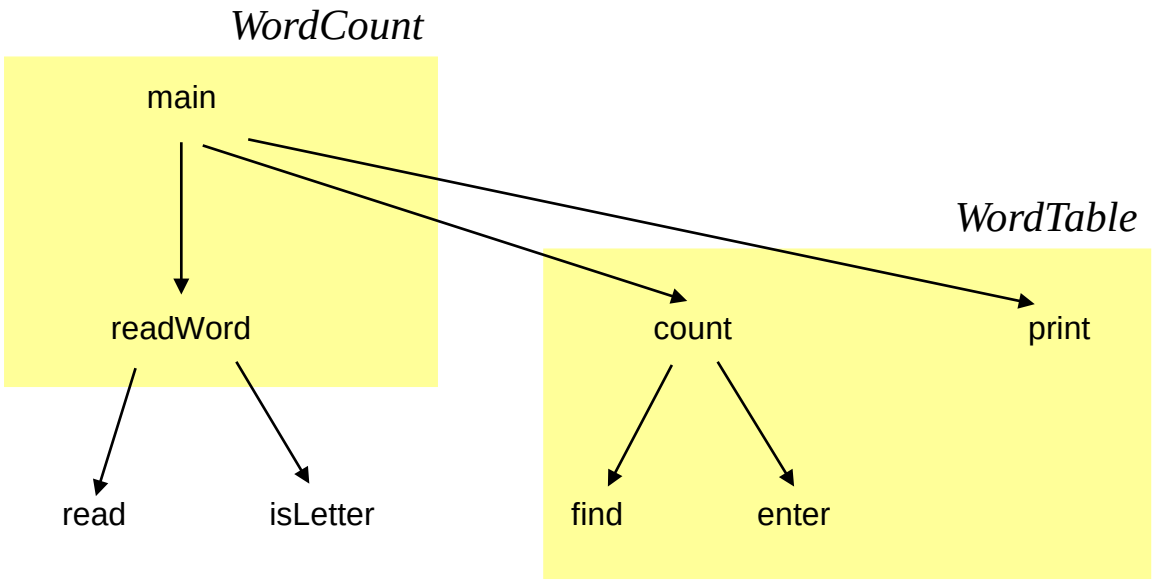

```
class Element {
    String word;
    int freq;
    Element next;

    Element (String w) {
        word = w; freq = 1;
    }
}
```

```
public class WordTable {
    private Element first = null;

    private Element find (String word) {
        Element e = first;
        while (e != null && !word.equals(e.word)) e = e.next;
        return e;
    }

    private void enter (String word) {
        Element e = new Element(word);
        e.next = first; first = e;
    }

    public void count (String word) {
        Element e = tab.find(word);
        if (e == null) tab.enter(word); else e.freq++;
    }

    public void print () {
        for (Element e = first; e != null; e = e.next)
            Out.println(e.word + ": " + e.freq);
    }
}
```

Zusammensetzen der einzelnen Teile


```
class WordCount {
    public static void main (String[] arg) {
        WordTable tab = new WordTable();
        In.open("input.txt");
        String w = readWord();
        while (w != null) {
            tab.count(w);
            w = readWord();
        }
        In.close();
        tab.print();
    }
    private static String readWord () {
        StringBuilder word = new StringBuilder();
        char ch;
        do ch = In.read(); while (In.done() && ! Character.isLetter(ch));
        while (In.done() && Character.isLetter(ch)) {
            word.append(ch);
            ch = In.read();
        }
        if (word.length > 0) return word.toString(); else return null;
    }
}
```

Aufrufhierarchie



17. Schrittweise Verfeinerung

Beispiel: Stichwortverzeichnis

Eingabe

Seitennummer *Stichwort* *Ende einer Seitenangabe*

```
1 = if-Anweisung; Verzweigung; Abfrage; #
2 = while-Anweisung; Schleife;
    Abfrage; #
3 = ...
```

Zwischendatei

```
if-Anweisung     1
Verzweigung      1
Abfrage          1
while-Anweisung  2
Schleife         2
Abfrage          2
...
```

Sortierung →

```
Abfrage          1
Abfrage          2
if-Anweisung     1
Schleife         2
Verzweigung      1
while-Anweisung  2
...
```

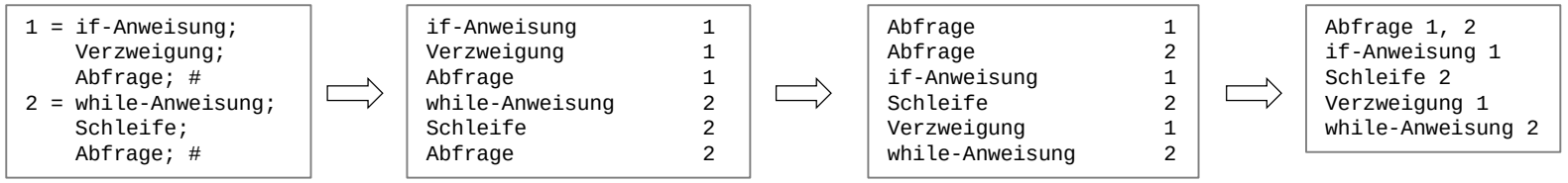
Ausgabe

```
Abfrage  1, 2
if-Anweisung  1
Schleife  2
...
```

Oberste Ebene

Welche Befehle würden wir uns wünschen?

- Erzeuge Wortliste buildWordList(String inFileName, String outFileName);
- Sortiere Wortliste sortWordList(String fileName);
- Erzeuge Index buildIndex(String inFileName, String outFileName);



Damit kann man die oberste Ebene bereits implementieren

```
public static void main(String[] arg) {
   if (arg.length < 2) {
      Out.println("input and output file names expected");
      return;
   }
   buildWordList(arg[0], "temp.txt");
   sortWordList("temp.txt");
   buildIndex("temp.txt", arg[1]);
}
```

Teiloperationen sind bereits wesentlich einfacher

buildWordList

Welche Befehle würden wir uns wünschen?

- Verarbeite Seite `boolean ok = processPage();`

  verarbeitet nächste Seitenangabe (Seitennummer, Stichwörter);
  liefert *false*, wenn keine Seite mehr verarbeitet werden kann

Damit kann man diese Ebene bereits implementieren

```
static void buildWordList(String inFileName, String outFileName) {
   In.open(inFileName);
   Out.open(outFileName);
   boolean ok;  // page correctly processed?
   do {
      ok = processPage();
   } while (ok);
   In.close();
   Out.close();
}
```

Teiloperation *processPage()* ist bereits wesentlich einfacher

ok = processPage()

Verarbeitet Seitenangabe der Art: pageNr = word; word; word; #
liefert *false*, wenn keine Seite mehr verarbeitet werden kann

Welche Befehle würden wir uns wünschen?

- Lies Seitennummer — String pageNr = readNumber();
  überliest führende Blanks; liefert Ziffernstring; überliest am Ende '=';
  gibt *null* zurück, wenn keine Seitennummer kommt
- Lies Stichwort — String word = readWord();
  überliest führende Blanks; liefert Text bis zu ';'
  gibt *null* zurück, wenn kein Stichwort kommt
- Gib Wortpaar aus — printPair(word, pageNr);
  gibt *word* mit 20 Zeichen aus, dahinter *pageNr* rechtsbündig mit 3 Stellen

```
static boolean processPage() {
   String pageNr = readNumber();
   if (pageNr == null) return false;
   String word = readWord();
   while (word != null) {
      printPair(word, pageNr);
      word = readWord();
   }
   return true;
}
```

readNumber() und readWord()

Sind einfach genug, um direkt zu implementiert zu werden

```
static String readNumber() {
   //--- skip leading blanks
   char ch;
   do {
      ch = In.read();
   } while (In.done() && !Character.isDigit(ch));
   // !In.done || ch is digit
   if (!In.done()) return null;
   // ch is digit

   //--- read number
   StringBuilder b = new StringBuilder();
   while (In.done() && Character.isDigit(ch)) {
      b.append(ch);
      ch = In.read();
   }
   // !In.done() || ch != digit

   //--- skip rest up to '='
   while (In.done() && ch != '=') ch = In.read();
   // !In.done() || ch == '='
   if (!In.done()) return null;
   return b.toString();
}
```

```
static String readWord() {
   //--- skip leading blanks
   char ch;
   do {
      ch = In.read();
   } while(In.done() && ch != '#'
      && !Character.isLetter(ch));
   // !In.done || ch == '#' || ch is letter
   if (!In.done() || ch == '#') return null;
   // ch is letter

   //--- read word
   StringBuilder b = new StringBuilder();
   while (In.done() && ch != ';') {
      b.append(ch);
      ch = In.read();
   }
   // !In.done() || ch == ';'
   return b.toString();
}
```

printPair

ist einfach genug, um direkt zu implementiert zu werden

```
static void printPair(String word, String pageNr) {
   int len = word.length();
   if (len >= MAXLEN) {
      word = word.substring(0, MAXLEN);
   } else { // len < MAXLEN
      word = word + BLANKS.substring(0, MAXLEN-len);
   }
   Out.print(word);
   Out.print(adjust(pageNr, 3));
   Out.println();
}
```

```
static final int MAXLEN = 20;
static final String BLANKS
   = "                    ";
```

```
static void adjust(String s, int n) {
   for (int i = s.length(); i < n; i++) s = " " + s;
   return s;
}
```