Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Angewandte und Numerische Mathematik

Programmieren: Einstieg in die Informatik mit Java

WS 2006/2007

Dr. G. Bohlender
Dipl.–Math. techn. M. Richter

04.12.2006

Aufgabenblatt 6

Bearbeitungszeitraum: 07.12.2006 – 20.12.2006

Aufgabe 15 (Pflichtaufgabe): *Phonetische Suche I*

Unter einem *phonetischen Algorithmus* versteht man ein Verfahren, das Wörtern anhand ihres Klanges eine Zeichenfolge zuordnet, die man als *phonetischen Code* bezeichnet. Ziel eines phonetischen Algorithmus ist es, ähnlichklingenden Wörtern denselben phonetischen Code zuzuordnen. Ein solcher Algorithmus wird bespielsweise benötigt, wenn man in einer Namensliste alle Personen finden möchte, deren Nachname wie „Meier“ klingt. Bekannterweise gibt es für diesen Nachnamen mehrere Schreibweisen, z.B. „Meyer“, „Maier“ oder „Mayer“.
Der bekannteste phonetische Algorithmus ist der SOUNDEX-Algorithmus, der jedoch nur für die englische Sprache gut funktioniert. Für die deutsche Sprache wurde daher ein Algorithmus entwickelt, der unter dem Namen „Kölner Phonetik“ bekannt ist. Dieser Algorithmus ordnet jedem Wort einen phonetischen Code bestehend aus den Zahlen 0 bis 8 zu. Dabei repräsentiert die Zahl 4 beispielsweise einen K-Laut.
Schreiben Sie ein Java-Programm mit dem Namen `PhonetischeSuche`, das ein Wort in einen phonetischen Code nach den Regeln der „Kölner Phonetik“ umwandelt. Gehen Sie dabei im einzelnen wie folgt vor:

(a) Erstellen Sie eine statische Methode `ersetzeGleichklingend` mit einen Argument `s` vom Typ `String`, die einen `String`-Wert zurück gibt. In dieser Methode sollen ähnlich- oder gleichklingende Konsonanten (z.B. G, K, Q) in der Zeichenkette `s` durch ein einziges Zeichen (z.B. K) ersetzt werden.

Wandeln Sie die Zeichenkette `s` zunächst in Großbuchstaben um, und fügen Sie das Dollarzeichen `$` an den Anfang von s an. Führen Sie anschließend folgende Substitutionen in der angegebenen Reihenfolge durch:

1. `PH` → `F`	8. `CA` → `KA`	15. `$CL` → `$KL`
2. `V` → `F`	9. `CO` → `KO`	16. `$CR` → `$KR`
3. `W` → `F`	10. `CU` → `KU`	17. `Z` → `S`
4. `B` → `P`	11. `SC` → `S`	18. `C` → `S`
5. `D` → `T`	12. `CH` → `K`	19. `TS` → `S`
6. `G` → `K`	13. `CK` → `K`	
7. `M` → `N`	14. `CX` → `KX`	

Geben Sie die veränderte Zeichenkette **ohne Dollarzeichen** zurück. Verwenden Sie zum Lösen dieses Aufgabenteils die Methoden `toUpperCase()`, `replace()`, `substring()` und `length()` der `String`-Klasse.

(b) Erstellen Sie eine statische Methode `entferneWiederholungen` mit einem Argument `s` vom Typ `String`. Die Methode soll in der übergebenen Zeichenkette `s` alle Zeichenwiederholungen (z.B. `LL`) durch ein einzelnes Zeichen (z.B. `L`) ersetzen. Definieren Sie dazu zunächst eine leere Zeichenkette `res`. Durchlaufen Sie anschließend `s` zeichenweise und stellen Sie jeweils fest, ob das Zeichen `z` an einer Position i mit seinem Nachfolger an der Position $i+1$ identisch ist. Falls nicht, so fügen Sie `z` an das Ende von `res` an. Geben Sie zum Schluss die Zeichenkette `res` zurück. Verwenden Sie die Methode `charAt()` der `String`-Klasse. Beachten Sie, dass diese Methode einen `char`-Wert zurück gibt.

(c) Erstellen Sie eine statische Methode mit dem Namen `erzeugeCode` und einem Argument `s` vom Typ `String`. Die Methode soll eine Zeichenkette, die zuvor durch die Methoden `ersetzeGleichklingend()` und `entferneWiederholungen()` modifiziert wurde, in phonetischen Code umwandeln. Gehen Sie dabei wie folgt vor: Definieren und initialisieren Sie zwei `String`-Variablen `zeichen` und `phcode` wie folgt

```
String zeichen = "AEIOUYJÄÖÜHPTFKLNRS";
String phcode  = "0000000000012345678";
```

Definieren Sie eine leere Zeichenkette `res` und durchlaufen Sie `s` zeichenweise. Bestimmen Sie für jedes Zeichen `z` von `s` die entsprechende Position `pos` innerhalb der Zeichenkette `zeichen`. Falls `z` in `zeichen` vorkommt, fügen Sie das Zeichen an der Position `pos` der Zeichenkette `phcode` an das Ende von `res` an. Das Zeichen `'0'` darf dabei nur am Anfang von `res` stehen. Verwenden Sie die Methoden `charAt()` und `indexOf()` der `String`-Methode. Geben Sie am Schluss die Zeichenkette `res` zurück.

(d) Erstellen Sie eine statische Methode mit dem Namen `phonetisch` und einem Argument `s` vom Typ `String`. Modifizieren sie in der Methode die Zeichenkette `s` zunächst mit `ersetzeGleichklingend()` und danach mit `entferneWiederholungen()`. Ersetzen Sie anschließend in der modifizierten Zeichenkette das Zeichen `X` durch `KS` und wenden Sie erneut `entferneWiederholungen()` auf die Zeichenkette an. Wandeln Sie zum Schluss die Zeichenkette mittels `erzeugeCode()` in phonetischen Code um und geben Sie diesen Code zurück.

Lesen Sie in der `main`-Methode Zeichenketten von der Konsole ein und wandeln Sie diese mit der Methode `phonetisch()` in phonetischen Code um. Geben Sie den phonetischen Code auf der Konsole aus. Überprüfen Sie Ihr Programm mit folgenden Eingaben

`Christoph, Christof, Kristov`	(Code: `47823`)
`Anna, Anne, Emma`	(Code: `06`)
`superkagiphagilistisch`	(Code: `81744345828`)

Aufgabe 16: *Phonetische Suche II*

Auf der Homepage der Vorlesung finden Sie unter

`www.mathematik.uni-karlsruhe.de/ianm2/lehre/java2006w`

die Textdatei `vornamen.txt`. Erweitern Sie das Programm `PhonetischeSuche` wie folgt: Lesen Sie in der `main`-Methode einen Vornamen ein und suchen Sie anschließend in der Datei **phonetisch** nach diesem Vornamen, indem Sie die von der Methode `phonetisch()` erzeugten Codes vergleichen. Verwenden Sie dazu die Methode `compareTo()` der `String`-Klasse. Geben Sie alle „Treffer“ auf der Konsole aus.

Aufgabe 17: *Fibonacci–Zahlen*

Die Fibonacci–Zahlen $a_1, a_2, a_3, \ldots (n \in \mathbb{N})$ werden durch die folgende Vorschrift definiert:

$$a_1 := 1, \quad a_2 := 1, \quad a_n := a_{n-1} + a_{n-2}$$

(a) Schreiben Sie eine statische Methode `iterativFibonacci` mit einem Argument `n` vom Typ `int`, welche die n-te Fibonacci–Zahl a_n iterativ berechnet. Dies soll mit Hilfe einer `for`-Schleife geschehen.

(b) Schreiben Sie eine statische Methode `rekursivFibonacci` mit zwei einem Argument `n` vom Typ `int`, welche die n-te Fibonacci–Zahl a_n **durch Rekursion** berechnet. Es soll keine Schleife verwendet werden.

(c) Erstellen Sie eine `main`-Methode, in der der Index n der gewünschten Fibonaccizahl a_n von der Konsole eingelesen wird. Rufen Sie anschließend nacheinander die beiden Methoden `iterativFibonacci()` und `rekursivFibonacci()` auf und geben Sie das jeweilige Ergebnis zurück.

Vergleichen Sie die beiden Funktionen, indem Sie die neunte, zehnte und die fünfundzwanzigste Fibonacci–Zahl berechnen.