Dr. Annette Bieniusa
Mathias Weber, M. Sc.
Peter Zeller, M. Sc.

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Übungsblatt 6: Software-Entwicklung 1 (WS 2017/18)

Ausgabe: 27.11.17
Abgabe: 04.12.17

Aufgabe 1 Strings (9 Punkte)

In Kapitel 3 der Vorlesung haben Sie bereits den Datentyp `String` kennen gelernt. Die Klasse `String` bietet neben den dort vorgestellten Operationen auch noch folgende Konstruktoren und Methoden an:

`String(char[] value)`	Wandelt `char[]` zu String um
`String(char[] value, int offset, int count)`	Wandelt `char[]` zu String um, verwendet aber nur `count` Zeichen startend ab `offset`
`int length()`	Länge des Strings
`char charAt(int i)`	Zeichen an Position `i` (erstes Zeichen bei 0)
`String substring(int i, int j)`	Teilstring von Zeichen `i` bis `j-1`
`boolean contains(String s)`	Prüft, ob der String `s` enthält
`boolean startsWith(String s)`	Prüft, ob String mit `s` anfängt
`boolean endsWith(String s)`	Prüft, ob String mit `s` endet
`int indexOf(String s)`	Erste Position, an der `s` im String vorkommt oder `-1` wenn er nicht gefunden wurde
`int indexOf(String s, int i)`	Wie oben, aber Suche wird erst ab Position `i` gestartet
`int lastIndexOf(String s)`	Wie `indexOf`, aber Suche startet am Ende des Strings
`String replace(String a, String b)`	String, in dem alle Vorkommen von `a` durch `b` ersetzt wurden
`String replace(char a, char b)`	String, in dem alle Vorkommen von `a` durch `b` ersetzt wurden
`char[] toCharArray()`	Wandelt String in `char[]` um

Eine detaillierte Dokumentation aller Methoden von `String` finden Sie unter
https://docs.oracle.com/javase/9/docs/api/java/lang/String.html.

In dieser Aufgabe sollen Sie Prozeduren in einer Klasse `StringProcs` schreiben.

a) Schreiben Sie eine Prozedur `String extension(String f)`, welche einen Dateinamen `f` nimmt und die Endung der Datei zurückgibt. Die Endung einer Datei ist der Teil des Dateinamens, der nach dem letzten Punkt kommt. Wenn kein Punkt in `f` vorkommt, soll der leere String zurückgegeben werden.

b) Im Englischen wird das Komma verwendet, um 1000-er Stellen voneinander zu trennen und als Dezimaltrennzeichen wird ein Punkt verwendet. Im Deutschen ist es anders herum.

   Schreiben Sie eine Prozedur `String switchCommaAndDot(String s)`, welche zwischen den Notationen umwandelt, also im String `s` alle Kommas durch Punkte ersetzt und alle Punkte durch Kommas.

c) Schreiben Sie eine Prozedur `int letterCount(String s)`, welche die Anzahl der Buchstaben im String `s` zählt. Verwenden Sie die Prozedur `Character.isLetter(char c)`, um zu prüfen, ob ein `char` ein Buchstabe ist.

d) Schreiben Sie eine Prozedur `int wordCount(String s)`, welche die Anzahl der Wörter im String `s` zählt. Ein Wort besteht aus Buchstaben (siehe oben) und zwischen Wörtern steht immer mindestens ein Zeichen, das kein Buchstabe ist.

Aufgabe 2 Bildbearbeitung (13 Punkte)

Laden Sie sich für diese Aufgabe die Datei `Picture.java` herunter. Diese enthält eine Klasse `Picture`, mit der Bilder bearbeitet werden können. In der Datei `PicEditTests.java` finden Sie JUnit Tests zu den Prozeduren dieser Aufgabe. Schreiben Sie Ihren eigenen Code in einer Datei namens `PicEdit.java`.

Die Klasse bietet zwei Konstruktoren:

```
Picture(String filename)       // Erstellt ein Bild aus einer Datei (.png oder .jpg)
Picture(int width, int height) // Erstellt ein Bild mit gegebener Höhe und Breite
```

Auf einem `Picture`-Objekt können die folgenden Methoden aufgerufen werden:

```
int width() // Breite des Bilds
int height() // Höhe des Bilds
Color get(int x, int y) // Liefert die Farbe des Pixels an der gegebenen Position
void set(int x, int y, Color c) // Setzt die Farbe eines Pixels
void show() // Zeigt das Bild in einem Fenster an
void save(String filename) // Speichert das Bild in einer Datei
```

Die Klasse `Color` ist bereits in der Java Standardbibliothek enthalten und kann mit **`import`** `java.awt.Color;` am Anfang der Datei importiert werden. Eine Farbe ist durch drei **`int`**-Werte zwischen 0 und 255 beschrieben, welche die Intensität der Farben Rot, Grün und Blau angeben (255 ist am hellsten). Eine Farbe kann über den Konstruktor `Color(`**`int`** `red, `**`int`** `green, `**`int`** `blue)` erstellt werden. Zum Lesen der Farbwerte gibt es Methoden `getRed`, `getGreen` und `getBlue`, welche den jeweiligen Farbwert als **`int`** zurückgeben. Weitere Methoden werden für diese Übung nicht benötigt.

a) Schreiben Sie eine Prozedur `lum`, welche eine Farbe nimmt und den Wert der Schwarz-Weiß-Luminanz für die gegebene Farbe berechnet. Diese ergibt sich aus der Formel: $0,299r + 0,587g + 0,114b$, wobei r, g und b für die Intensität der Farben Rot, Grün und Blau stehen.

b) Schreiben Sie eine Prozedur `averageLum`, welche ein `Picture` nimmt und die durchschnittliche Schwarz-Weiß-Luminanz aller Pixel im Bild berechnet.

c) Schreiben Sie eine Prozedur `changedColorSaturation`, welche eine Farbe nimmt und eine neue Farbe mit geänderter Farb-Sättigung zurück gibt. Die Änderung der Sättigung soll dabei durch einen Parameter *change* vom Typ **`double`** angegeben werden.

   Um die Farbe mit neuer Sättigung zu berechnen, verwenden Sie die folgenden Formeln:
   $r' = p + (r - p) \cdot change$
   $g' = p + (g - p) \cdot change$
   $b' = p + (b - p) \cdot change$

   Dabei stehen r, g und b für die Intensität der Farben Rot, Grün und Blau; r', g' und b' sind die Werte mit geänderter Sättigung und p ist der Wert der Schwarz-Weiß-Luminanz der Farbe. Um das Ergebnis als Farbwert zu verwenden, müssen diese mit `Math.round` auf eine ganze Zahl gerundet werden. Falls durch die Formel Werte außerhalb des gültigen Intervalls $[0, 255]$ entstehen, sollen die Werte entsprechend korrigiert werden.

d) Schreiben Sie eine Prozedur `changeSaturation`, welche ein `Picture` und einen Wert zur Änderung der Sättigung nimmt. Die Prozedur soll dann mit Hilfe der Prozedur `changedColorSaturation` für alle Pixel des Bilds die Sättigung der Farbe ändern.

e) Schreiben Sie eine Prozedur `rotatedLeft`, welche ein `Picture` nimmt und ein neues Bild zurück gibt, das um 90° entgegen dem Uhrzeigersinn gedreht ist.

f) Schreiben Sie eine `main`-Prozedur, so dass Ihr Programm für die folgenden 3 Aufgaben verwendet werden kann. Die

   1. Berechnen der durchschnittlichen Schwarz-Weiß-Luminanz eines Bildes mit Angabe der Eingabe-Datei.
   2. Ändern der Sättigung eines Bildes mit Angabe der Eingabe-Datei, Ausgabe-Datei und Änderung der Sättigung.
   3. Drehen eines Bildes um 90° entgegen dem Uhrzeigersinn mit Angabe der Eingabe- und Ausgabe-Datei.

Aufgabe 3 Aufzugsteuerung (7 Punkte)

In dieser Aufgabe sollen Sie eine Klasse `Aufzugsteuerung` entwickeln, welche einen Aufzug kontrolliert.

a) Die Aufzugsteuerung soll einen Konstruktor `Aufzugsteuerung(int currentFloor, int minFloor, int maxFloor)` bieten, in dem das aktuelle Stockwerk, das minimale Stockwerk und das maximale Stockwerk angegeben sind.

Die Aufzugsteuerung hat keine Sensoren, muss also selbst speichern, in welchem Stockwerk sich der Aufzug befindet und ob die Tür auf oder zu ist.

Zu Beginn ist die Tür des Aufzugs auf und er befindet sich in `currentFloor`.

Implementieren Sie den oben genannten Konstruktor und die entsprechenden Attribute für die Klasse `Aufzugsteuerung`.

b) Implementieren Sie eine Methode `void request(int floor)`. Diese wird vom Aufzug aufgerufen, wenn ein Reisender den Knopf für Stockwerk `floor` drückt.

Die angeforderten Stockwerke sollen von der Aufzugsteuerung in einer Warteschlange gespeichert werden. Dazu können Sie sich die Klasse `IntQueue` herunterladen. Diese bietet die folgenden Konstruktoren/Methoden:

	`IntQueue()`	Konstruktor; liefert neue leere Warteschlange
`void`	`enqueue(int x)`	Fügt die Zahl `x` am Ende der Warteschlange ein
`int`	`peek()`	Liefert die erste Zahl in der Warteschlange ohne sie zu entfernen
`int`	`dequeue()`	Entfernt die erste Zahl aus der Warteschlange und gibt sie zurück
`boolean`	`isEmpty()`	Prüft, ob die Warteschlange leer ist

c) Implementieren Sie eine Methode `int action()`. Diese gibt die nächste Aktion zurück, die der Aufzug ausführen soll. Dabei wird eine Aktion durch einen `int`-Wert ausgedrückt. Dieser ist entweder 0 (keine Aktion ausführen), 1 (ein Stockwerk nach oben fahren), 2 (ein Stockwerk nach unten fahren), 3 (Tür öffnen) oder 4 (Tür schließen).

Gehen Sie wie folgt vor, um die nächste Aktion zu bestimmen:

- Wenn die Warteschlange leer ist, soll keine Aktion ausgeführt werden (Rückgabewert 0).
- Wenn das erste Stockwerk in der Warteschlange das aktuelle Stockwerk ist und die Tür geschlossen ist, soll die Tür geöffnet werden. Das Stockwerk ist damit erreicht und kann aus der Warteschlange entfernt werden.
- Andernfalls, muss die Tür geschlossen werden. Wenn die Tür dann geschlossen ist, wird das erste Stockwerk in der Warteschlange betrachtet und dementsprechend soll sich der Aufzug nach oben oder unten bewegen.

d) Testen Sie Ihre Implementierung mit den JUnit-Tests in `AufzugTests.java`.

Aufgabe 4 Tabellenkalkulation (8 Punkte)

In dieser Aufgabe sollen sie Klassen für eine einfache Tabellenkalkulation (engl. Spreadsheet) schreiben, wie sie zum Beispiel in den Programmen Excel oder LibreOffice Calc bekannt ist. Schreiben Sie dazu eine Klasse `Spreadsheet` mit den folgenden Konstruktoren und Methoden:

```
public class Spreadsheet {
  Spreadsheet(int columns, int rows) { ... }
  void set(int column, int row, int value) { ... }
  int get(int column, int row) { ... }
  void setSumCalculation(int resultColumn, int resultRow,
                         int startColumn, int startRow,
                         int endColumn, int endRow) { ... }
}
```

Im Konstruktor wird die Anzahl der Spalten und Reihen der Tabelle angegeben. In jeder Zelle wird ein `int`-Wert gespeichert, welcher zu Beginn 0 ist und mit der Methode `set` geändert werden kann. Dabei hat die Zelle oben links die Reihe 0 und Spalte 0 und die Zelle unten rechts entsprechend Reihe `rows - 1` und Spalte `columns - 1`. Mit der Methode `get` kann der aktuelle Wert einer Zelle gelesen werden.

Neben dem einfachen Speichern und Lesen von Werten kann das Spreadsheet auch Berechnungen durchführen. Die Methode `setSumCalculation` nimmt eine Ergebnis-Zelle `(resultColumn, resultRow)` und einen Bereich (von Zelle `(startColumn, startRow)` bis `(endColumn, endRow)`). In der Ergebnis-Zelle soll dann immer die Summe aller Zellen-Werte im Bereich stehen. Wenn sich ein Wert im angegebenen Bereich ändert, soll auch in der Ergebnis-Zelle die aktualisierte Summe stehen. Sie können davon ausgehen, dass dabei keine zirkulären Abhängigkeiten entstehen. Es ist zum Beispiel nicht erlaubt (und muss auch nicht überprüft werden), dass die Ergebnis-Zelle im Summen-Bereich vorkommt.

Verwenden Sie die Tests aus `SpreadsheetTests.java` um Ihre Implementierung zu testen.

Beispiel:

0	0	0	0
0	0	0	0
0	0	0	0

```
Spreadsheet s = new Spreadsheet(4, 3);

// Ein Spreadsheet mit 4 Spalten und 3 Zeilen
// enthält initial in jeder Zelle den Wert 0.
```

1	2	3	0
4	5	6	0
0	0	0	0

```
s.set(0, 0, 1);
s.set(1, 0, 2);
s.set(2, 0, 3);
s.set(0, 1, 4);
s.set(1, 1, 5);
s.set(2, 1, 6);
// Nach dem  Setzen der Werte mit set
// enthalten die Zellen die Werte
```

1	2	3	6
4	5	6	0
0	0	0	0

```
// In Spalte 3, Zeile 0 soll die Summe von Zeile 1 stehen:
s.setSumCalculation(3, 0, 0, 0, 2, 0);
```

1	2	3	6
4	5	6	15
5	7	9	0

```
// weitere Summen:
s.setSumCalculation(3, 1, 0, 1, 2, 1);
s.setSumCalculation(0, 2, 0, 0, 0, 1);
s.setSumCalculation(1, 2, 1, 0, 1, 1);
s.setSumCalculation(2, 2, 2, 0, 2, 1);
```

1	2	3	6
4	5	6	15
5	7	9	21

```
// Summen können auch von anderen Summen abhängen:
s.setSumCalculation(3, 2, 3, 0, 3, 1);
```

Freiwillige Zusatzaufgabe (0 Punkte)

Die folgende Aufgabe können Sie selbstständig bearbeiten und abgeben. Dazu finden Sie im Exclaim eine zusätzliche Übung “SE1WS17Z”, wo Sie Ihre Lösung als Programm `Dunglish.java` hochladen können. Die Abgaben werden automatisch getestet, aber nicht bewertet.

Bei der Lösung der Aufgaben kommt es auch auf die Effizienz des Algorithmus an. Mit einer langsamen Lösung werden Sie wahrscheinlich für große Eingaben einen Timeout erhalten.

Dunglish

A group of confused Dutchmen. Public domain.

A confused Dutchman trying to speak English could say “*I am in the war*”, even though there is no hostile activity going on. The confusion[1] here is that the English sentence “*I am confused*” is translated in Dutch as “*Ik ben in de war*”, which is phonetically (“sounding”) quite close to the first sentence. Such confusion leads to much enjoyment, but can complicate matters a bit.

Given a sentence in Dutch and a dictionary containing both correct translations as well as phonetic (incorrect) translations of individual words, find the translation of the sentence and indicate whether it is correct, or in case there is more than one find the total number of correct and incorrect translations. A sentence is correctly translated when each word of the sentence is correctly translated.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 20$), the number of words in the Dutch sentence.
- One line with n words, the Dutch sentence s.
- One line with an integer m ($1 \leq m \leq 10^5$), the number of words in the dictionary.
- m lines, each with three strings d, e and c, a Dutch word, the English translation, and “`correct`” if this is the correct translation or “`incorrect`” otherwise.

A word consists of between 1 and 20 lowercase letters. Each word in s appears at least once as a Dutch word in the dictionary, no word appears more than 8 times as a Dutch word in the dictionary, and each combination of a Dutch and English word appears at most once.

Output

In case there is only a single translation of s, output one line with the translation followed by one line with “`correct`” or “`incorrect`”. In case there is more than one translation, output one line with the number of possible correct translations followed by “`correct`”, and one line with the number of possible incorrect translations followed by “`incorrect`”.

[1] Pun intended.

Sample Input 1	Sample Output 1
7 als mollen mollen mollen mollen mollen mollen 4 als when correct mollen moles correct mollen destroy correct mollen mills incorrect	64 correct 665 incorrect

Sample Input 2	Sample Output 2
5 de zuigers zijn buiten werking 6 zijn are correct banaan banana correct de the correct zuigers suckers incorrect buiten out correct werking working incorrect	the suckers are out working incorrect