1 Wörter zählen (3+1 Punkte)

Schreibe ein Programm, das zählt, wie oft die einzelnen Wörter in einem Korpus auftreten. Du kannst annehmen, dass zwischen je zwei Wörtern mindestens ein Leerzeichen steht. Interpunktion (z.B. Kommata) zählen als eigene Wörter. Groß- und Kleinschreibung werden unterschieden.

Da wir noch nicht wissen, wie man Dateien einliest, kannst Du die abstrakte Klasse `LineByLine` von der Webseite herunterladen und Deine Klasse davon ableiten. `LineByLine` erwartet, dass die abstrakte Methode `processLine` überschrieben wird; diese Methode bekommt sukzessive die einzelnen Zeilen der Datei als Argument übergeben. Um den Durchlauf durch eine Datei mit Namen `filename` anzustoßen, rufst Du `lbl.processFile(filename)` auf einem Objekt `lbl` auf.

Teste Dein Programm auf dem Korpus `A00-no-markup`, das Du von der Webseite herunterladen kannst. (Es handelt sich um die erste Datei des British National Corpus.) Die Ausgabe sollte ungefähr so aussehen:

```
Despite  ->  2
possible  ->  3
time  ->  14
...
```

Für den Bonuspunkt lade Dir das Korpus `A00-pos` herunter. In diesem Korpus stehen nicht einzelne Wörter, sondern Paare `wort$pos` aus Wort und Part-of-Speech-Tag. Wort und Tag sind durch ein Dollar-Zeichen getrennt. Schreibe ein Programm, das für jedes Wort ausgibt, welches POS-Tag wie oft mit diesem Wort aufgetreten ist.

2 Primzahlen (3 Punkte)

Schreibe ein Programm, das alle Primzahlen unterhalb einer Zahl n berechnet. Die Zahl n soll auf der Kommandozeile angegeben werden. Implementiere dazu den Algorithmus “Sieb des Eratosthenes”, der zunächst alle Zahlen zwischen 2 und n als Primzahlen markiert. Dann durchläuft er in einer Schleife alle Zahlen zwischen 2 und $n/2$ und markiert ihre Vielfachen als Nicht-Primzahlen.

(a) Implementiere den Algorithmus unter Verwendung eines `Set<Integer>`-Objekts, das alle Primzahlen enthält.

(b) Implementiere den Algorithmus unter Verwendung eines `boolean`-Arrays, das angibt, ob der Index eine Primzahl ist oder nicht.

(c) Vergleiche die Laufzeiten der beiden Implementierungen für $n = 10^5$, $n = 10^6$ und $n = 10^7$, und interpretiere die Ergebnisse. Die Primzahlen müssen hier nur berechnet, nicht ausgegeben werden. Verwende zur Zeitmessung die Methode `currentTimeMillis()` aus der Klasse `System`.

3 Tic-Tac-Toe, Teil 1 (5 Punkte)

Im Laufe des Semesters werden wir ein System bauen, bei dem Menschen gegeneinander und evtl. gegen den Computer Tic-Tac-Toe spielen können. Im ersten Schritt geht es diese Woche darum, das Spielbrett und die Spielregeln zu implementieren. Ladet Euch dazu von der Webseite folgende Interfaces und Klassen herunter:

- Interface `games.IBoard`: Repräsentiert ein rechteckiges Spielbrett; auf jedem Feld kann ein Spielstein einer Farbe liegen.
- Interface `games.ILogic`: Repräsentiert die Spielregeln, indem auf einem gegebenen `IBoard`-Objekt Züge ausgeführt werden können. Objekte dieses Interfaces prüfen auch, ob das Spiel vorbei ist und welcher Spieler gewonnen hat.
- Interface `games.IConverter`: Repräsentiert eine Methode, ein `IBoard`-Objekt in einen String (zur Ausgabe) zu konvertieren.
- Klasse `games.Color`: Definiert verschiedene Farben.
- Klasse `games.TicTacToe`: Enthält ein Hauptprogramm, das ein Spiel durchspielt; die Züge werden auf der Kommandozeile angegeben.

Schreibe nun Klassen, die die drei Interfaces für Tic-Tac-Toe implementieren. Du sollst nichts an den vorgegebenen Dateien ändern: Stell Dir vor, wir hätten sie Dir nur kompiliert gegeben. Das bedeutet, Deine eigenen Klassen müssen in der Package `games.tictactoe` liegen.

Ein Beispielablauf des Programms soll ungefähr folgendermaßen aussehen:

```
koller@cicero:~$ java TicTacToe 1 1 0 0 2 2 1 2 2 1 0 1 2 0
... (einige Ausgaben des Programms für die ersten paar Züge) ...
Game over!
And the winner is white
+--+--+--+
|><|  |()|
+--+--+--+
|><|()|()|
+--+--+--+
|  |><|()|
+--+--+--+
```

Dabei ist `()` die Markierung für Steine von Weiß und `><` die für Steine von Schwarz.

4 Sprachdesign (1 Punkt)

Was haben sich die Designer von Java dabei gedacht, als sie verboten haben, eine Methode gleichzeitig `abstract` und `static` zu deklarieren?

Abgabe bis 31. 5. 2004, 9 Uhr

(java-uebungen@coli.uni-sb.de)