Image Analysis Lab
Jun.-Prof. Dr.
Olaf Ronneberger
Claudius Korzen

Algorithmen und Datenstrukturen
(ESE + IEMS) WS 2014 / 2015

http://lmb.informatik.uni-freiburg.de/lectures/

UNI FREIBURG

Übungsblatt 4

Abgabe für ESE: bis Donnerstag, den 20. November um 10:00 Uhr

Abgabe für IEMS: bis Donnerstag, den 4. Dezember um 10:00 Uhr

Aufgabe 1 (15 Punkte)

Schreiben Sie eine Klasse *GeoNamesAnalyzer* mit folgenden Methoden:

1. Eine Methode *readInfoFromFile*, die alle für das Folgende relevanten Informationen aus der auf der Vorlesungs-Homepage abgelegten GeoNames-Datei *allCountries.txt* einliest. Als Orte gelten alle Zeilen in *allCountries.txt* mit einem *P* in Spalte 7. Entnehmen Sie den Ortsnamen der Spalte 2. Betrachten Sie nur Orte mit > 0 Einwohnern (Spalte 15). Der Ländercode steht in Spalte 9.

2. Eine Methode *computeMostFrequentCityNamesUsingSorting*, die die weltweit am häufigsten vorkommenden Ortsnamen ermittelt, mittels Sortieren.

3. Eine Methode *computeMostFrequentCityNamesUsingMap*, die genau das selbe berechnet, aber mittels eines assoziatives Arrays.

Schreiben Sie für die beiden *compute...* Methoden jeweils einen Unit Test. Nutzen sie für die Tests nicht die *allCountries.txt*-Datei, sondern lassen sie ihr Programm eine minimale Test-Datei selbst erzeugen. ACHTUNG: Die *allCountries.txt*-Datei dürfen Sie in keinem Fall ins SVN einchecken!

Tipps:

- Lesen Sie die Datei zeilenweise (getline() in C++, BufferedReader.readLine() in Java)
- Nutzen Sie String.split() in Java um die einzelnen Felder zu erhalten.
- In C++ gibt es leider kein split(). Schreiben sie eine Schleife, die alle Tab-Positionen (line[i] == "\t") findet und extrahieren sie die Felder, die sie brauchen, anschließend mit substr(), also z.B.

  ```
  std::string features = line.substr( tabpos[5]+1, tabpos[6]-tabpos[5]-1);
  ```

  etc.
- Sie dürfen beim Einlesen und Parsen der Datei auf jegliche Fehlerbehandlung verzichten.
- Erstellen sie zwei Hilfsklassen: *CityInfo* mit *name* und *country_code* um die eingelesenen Städte zu speichern und *CityCount* mit *name* und *count* um die Häufigkeit eines Städtenamens zu speichern. In C++ können Sie in beiden Klassen den Vergleichsoperator *operator<()* überladen. In Java müssen Sie jeweils eine Comparator-Klasse dazu implementieren.

- Benutzen Sie die Algorihtmen aus der jeweiligen Standard-Bibliothek. Also std::sort und std::unordered_map in C++, sowie Collections.sort und HashMap in Java.
- Nachdem Sie die Häufigkeit für jede Stadt ermittelt haben, müssen Sie die Ergebnisse in ein neues Array (vom Typ CityCount) umkopieren, um sie dann zu sortieren.
- Zum Entwickeln ihres Programmes können Sie z.B. die Daten aus der Schweiz (CH.zip auf geonames.org) nutzen. Die Datei hat nur ca. 23000 Zeilen.
- Fragen Sie im Forum, wenn Sie nicht weiterkommen.

Aufgabe 2 (5 Punkte)

Schreiben Sie ein Programm *GeoNamesAnalyzerMain*, welches die drei weltweit am häufigsten vorkommenden Ortsnamen berechnet. Vergleiche Sie die Laufzeiten Ihrer beiden *compute...* Methoden für diese Aufgabe. Notieren Sie Ihre Ergebnisse, sowie eine kurze(!) Diskussion dazu in der Datei *erfahrungen.txt* (zusammen mit dem üblichen Feedback, siehe unten).

Zusatzaufgabe 3 (5 Punkte)

Ändern Sie dann eine der beiden Methoden (oder beide) so ab, dass Sie die drei weltweit am häufigsten Ortsnamen berechnen, die auch *mindestens einmal* in Deutschland (Ländercode DE) vorkommen.

Committen Sie, wie gehabt, Ihren Code in das SVN, in einen neuen Unterordner *uebungsblatt_04*, sowie, ebendort, Ihr Feedback in einer Textdatei *erfahrungen.txt*. Insbesondere: Wie lange haben Sie ungefähr gebraucht? An welchen Stellen gab es Probleme und wieviel Zeit hat Sie das gekostet?