Vorgehensweise und Entscheidungsfindung

Nach der Mail von Herrn Kulas haben wir zunächst überlegt, wie wir nun Vokale und Konsonanten erneut festlegen. Wir haben uns nach einiger Zeit und vielen verschiedenen Meinungen im Internet und in Büchern dazu entschieden, Vokale und Konsonanten wie folgt zu deklarieren:
Vokale: a,e,i,o,u,ä,ö,ü
Konsonanten: b,c,d,f,g,h,j,k,l,m,n,p,q,r,s,ß,t,v,w,x,y,z
Weiterhin betrachten wir weitere etwaige Sonderzeichen, wie z.B. das é oder á, als Zeichen und weder als Vokal noch als Konsonant. Weitere Sonderfälle, wie z.B. auch dass ein y manchmal ein Vokal ist, haben wir nicht weiter verfolgt.
Für die Wort- und Zeichenlisten haben wir die LinkedList gewählt, da sich schnell neue Elemente einfügen lassen. Wenn wir durch eine List iterieren, entstehen die gleichen Kosten wie bei einer ArrayList, da wir nicht nur ein einzelnes Element brauchen, sondern alle durchlaufen.
Zu Beginn des Programmablaufs muss zunächst die Methode “processTxtFile” ausgeführt werden, damit die Listen “words” und “chars” gefüllt werden.
Um die Wörter zählen zu können, haben wir uns zwei Pattern angelegt. Das Erste enthält alle im Text gefundenen Trennzeichen. Anschließend hatten wir aber das Problem, dass manche Leerzeilen noch als Wort erkannt wurden und manche Bindestriche. Also haben wir ein weiteres Pattern erstellt, welches besagt, dass ein Wort mindestens ein Zeichen zwischen a-z und A-Z enthalten muss. Zusammen mit der Abfrage, dass die Stringlänge ungleich null sein muss, konnten wir so die übrigen Konstrukte herausfiltern, die keine Wörter darstellen.
Bei der Anagrammerkennung werden auch Anagramme mit unterschiedlicher Groß- und Kleinschreibung berücksichtigt, dadurch gibt es zwar auch “unechte” Anagramme, wie z.B. “wie” und “Wie”, dafür gehen aber Anagramme mit unterschiedlicher Bedeutung bei unterschiedlicher Groß- und Kleinschreibung nicht verloren, z.B. “Fest” und “fest”.
Gespeichert werden die Anagramme in einer Map, als Key dient der sortierte String und alle gefundenen Anagramme werden zum jeweiligen Key in einem Set gespeichert.
Eigene Testfälle für die Aufgabenteile 4 und 5 haben wir nicht geschrieben, wir haben statt dessen einfach eine Testdatei mit Wörtern benutzt ☺

class InputOutput

Diese Klasse regelt alle Ausgaben für die Konsole. Sie ist recht einfach gehalten, die Methoden erhalten die auszugebenden Werte als Übergabeparameter und geben sie anschließend in geeigneter Form wieder aus.

class FileIO

Diese Klasse ist für die Dateibearbeitung. Die Textdatei wird eingelesen und verarbeitet. Im Anschluss daran kann man die weiteren Methoden nutzen, um die entsprechenden Werte zu erhalten.

class VowalsConsonantsHelper

Diese Klasse ist recht trivial, sie hilft bei der Erkennung, ob ein Zeichen ein Vokal oder Konsonant ist und gibt entsprechend true oder false zurück. Die beiden Methoden sind sehr einfach gestrickt und könnten sicher eleganter gelöst werden.

class MainApp (main-Methode)
Diese Klasse enthält die main-Methode und steuert den Programmablauf. Zunächst wird versucht die Datei einzulesen. Mögliche Exceptions hierbei werden abgefangen. Anschließend werden Vokale, Konsonanten, Wörter und Zeichen gezählt und ausgegeben. Danach werden Anagramme und Palindrome gesucht und ausgegeben.

Martin Slowikowski
Matrikelnummer: 199 91 66

Jan-Tristan Rudat
Matrikelnummer: 200 78 52

Teamname: Bernie und Ert

Quellenangaben

Class Pattern
Sun Microsystems, Inc.; Class Pattern
URL: http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html
(abgerufen am 13.06.2010)