Aufgaben „Reguläre Ausdrücke“

Zielsetzung der Aufgaben

Diese Aufgaben schaffen eine Übungsgrundlage für den Umgang mit regulären Ausdrücken. Nach einer grundlegenden Einführung in die Thematik wird die Konstruktion von regulären Ausdrücken für verschiedene Zwecke sowie die Anwendung auf Zeichenketten aus unterschiedlichen Quellen geübt.

In dieser Aufgabe werden besonders das Verketten unterschiedlicher Streams und das Einlesen von Textdateien beübt.

Zielsetzung dieses Dokuments

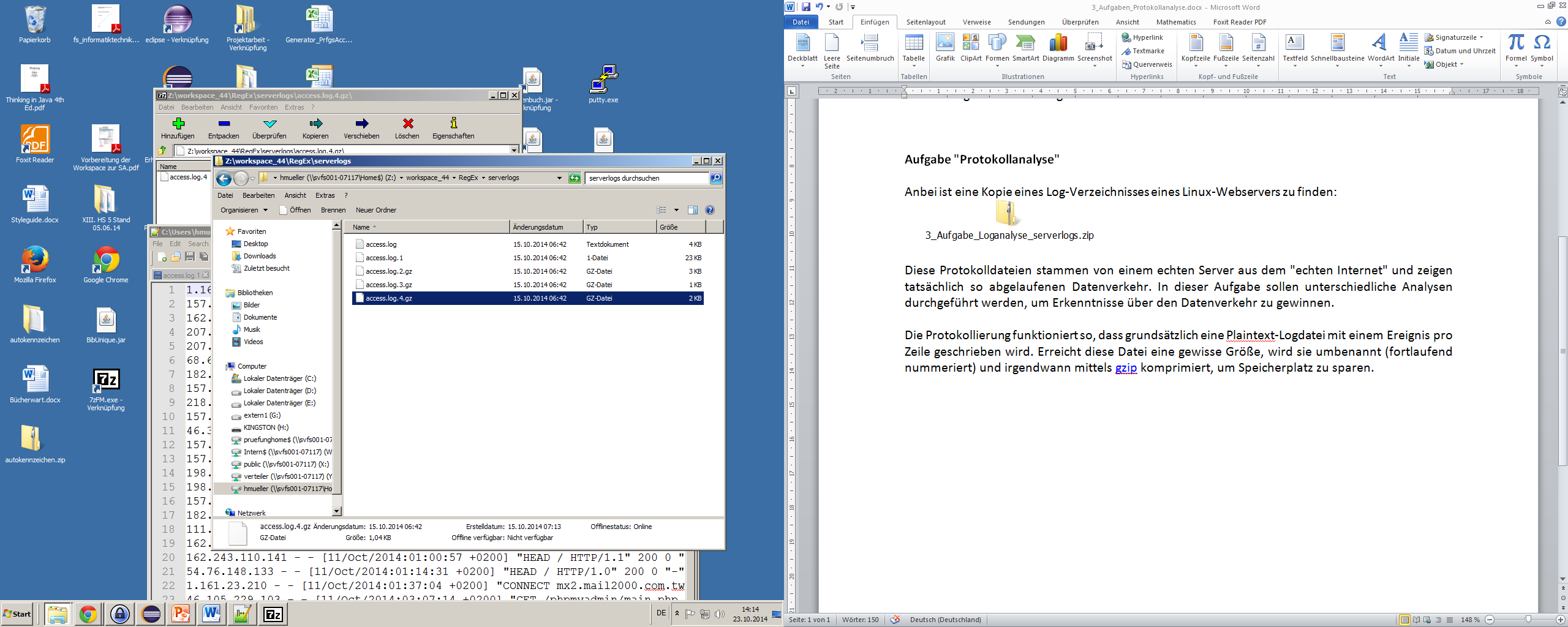
Dieses Dokument beschreibt die Programmieraufgaben vollständig. Eventuell erforderliche Klassendiagramme sind eingebettet.

**Aufgabe "Protokollanalyse eines Webservers"**

Vorbemerkung

Anbei ist eine Kopie eines Log-Verzeichnisses eines Linux-Webservers zu finden:

Diese Protokolldateien stammen von einem echten Server aus dem "echten Internet" und zeigen tatsächlich so abgelaufenen Datenverkehr. In dieser Aufgabe sollen unterschiedliche Analysen durchgeführt werden, um Erkenntnisse über den Datenverkehr zu gewinnen.

Die Protokollierung funktioniert so, dass grundsätzlich eine [Plaintext](http://static.squarespace.com/static/518f5d62e4b075248d6a3f90/t/542c74bae4b07ed32b846dc1/1412199617004/?format=750w)-Logdatei mit einem Ereignis pro Zeile geschrieben wird. Erreicht diese Datei eine gewisse Größe, wird sie umbenannt (fortlaufend nummeriert) und irgendwann mittels [gzip](http://de.wikipedia.org/wiki/Gzip) komprimiert, um Speicherplatz zu sparen.

Nebenstehend ist der Inhalt des serverlogs-Verzeichnisses zu sehen. Die access.log-Datei ist die aktuelleste, in zur Zeit geschrieben wird.

Die access.log.1-Datei ist die zweitjüngste, umbenannt, aber noch nicht komprimiert.

Die übrigen access.[234].gz-Dateien sind die alten Logs – je größer die Zahl, deso älter die Einträge.

Programmierung eines LogFileReaders

Die Klasse LogFileReader enthält nur eine öffentliche Methode, die einen Pfad-String übernimmt und einen String mit den Dateiinhalten zurückgibt. Handelt es sich bei dem übernommenen Pfad um eine Datei, soll diese eingelesen werden.

Handelt es sich um ein Verzeichnis, sollen alle Dateien in diesem Verzeichnis eingelesen werden – und zwar unabhängig davon, ob es sich um Textdateien oder um bereits komprimierte Dateien handelt.

Die korrekte Reihenfolge der Dateien ist nicht wichtig – jede Ereignis-Zeile enthält einen Zeitstempel, der eine zeitliche Einordnung ermöglicht.

GZip-komprimierte Dateien können mit einem [GZIPInputStream](http://docs.oracle.com/javase/7/docs/api/java/util/zip/GZIPInputStream.html) gelesen werden, allerdings muss zusätzlich noch eine "Übersetzung" des resultierenden (entpackten) InputStreams in einen zeichenorientierten Stream erfolgen.

Überlegen Sie sich eine kluge Art, mit den unterschiedlichen Dateiformaten umzugehen!

InputStream in Reader überführen, nachlesen!

Analyzer-Klassen

Alle Analyzer implementieren das Analyzer-Interface. Dieses schreibt eine öffentliche Methode

String analyze( String logContent)

vor. Die Rückgabe enthält das Analyseergebnis in "menschenlesbarer" Form. Unterschiedliche Analyzer untersuchen unterschiedliche Dinge.

Achten Sie darauf, die verwendeten Patterns mit dem Flag [Pattern.MULTILINE](http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#MULTILINE) (= im Modus für Strings mit Zeilenumbrüchen) zu erzeugen!

HTTP-Protokollversion-Analyzer

Dieser Analyzer findet die unterschiedlichen verwendeten Versionen des HTTP-Protokolls, und gibt einen String zurück, in dem die vorkommenden Versionen aufgeführt sind.

HTTP-Command-Analyzer

Dieser Analyzer findet und zählt die unterschiedlichen protokollierten HTTP-Befehle (mindestens GET, POST, HEAD und CONNECT).

UserAgent-String-Analyzer

Dieser Analyzer findet die einzigartigen UserAgent-Strings. Die Rückgabe besteht aus der Anzahl der gefundenen unterschiedlichen UserAgent-Strings, der Anzahl der UserAgent-Strings, die behaupten, von einem Bot (z.B. Googlebot, bingbot)zu stammen, und aus allen einzigartigen UserAgent-Strings.

Einzigartig in Collection nachlesen

Welche UserAgent-Strings wirken dabei "etwas merkwürdig"? Auf was kann man daraus schließen?

IP-Analyzer

Dieser Analyzer extrahiert die IPv4-Adressen am Zeilenbeginn (d.h. die Adressen der Clients). Er findet die einzigartigen Adressen, und gibt deren Anzahl sowie sämtliche einzigartigen Adressen in einer Zeile zurück.

Voraussetzung für nächste Aufgabe

In der nächsten Aufgabe wird dieser analyzer um eine Geolokation über einen Webdienst erweitert.

Starter-Klasse

Die Starter-Klasse enthält die main()-Methode. Hier werden die Logs einmal eingelesen und dann durch die verschiedenen Analyzer analysiert. Die Ausgabe könnte so aussehen:

Loaded 53983 bytes in log files.

Command analysis:

Command POST found 8 times.

Command GET found 286 times.

Command CONNECT found 17 times.

Command HEAD found 9 times.

HTTP-Version analysis :

Found HTTP-Versions: [1.0, 1.1]

User Agent analysis:

Found 24 different user agent Strings.

5 are identifying as Bots

Mozilla/5.0 (Windows NT 6.1; WOW64; rv:29.0) Gecko/20100101 /\*gekürzt\*/

IP Address Analysis:

Extracted 89 unique adresses.

[1.169.94.122, /\*gekürzt\*/ 157.55.39.185]