Übung 3: Multithreading und NIO (40 Punkte)

Aufgabenstellung

In dieser Übung sollen Sie ein Programm schreiben, das ein Verzeichnis mit Textdateien (Endung `.txt`) beobachtet, die Textdateien im Verzeichnis nach mit Hashtags gekennzeichneten Wörtern durchsucht und einen Index aufbaut. Im Index sollen für jedes Wort die Dateipfade, der Dateien die dieses Wort enthalten, eingetragen werden.

Der Index ist in einer Swing-Anwendung wie in Abbildung 1 zu sehen anzuzeigen. Links ist eine Liste (`JList`) mit den gefundenen Wörtern zu sehen. Im mittleren Teil sind für das selektierte Wort in einer Liste (`JList`) die Dateipfade angezeigt, die das selektierte Wort enthalten. Im rechten Teil ist der Inhalt der selektierten Datei in einer `JTextArea` ausgegeben. Bei jeder Änderung soll die Anzeige der Wörter und der angezeigten Dateipfade aktualisiert werden (der Text muss bei Änderungen der Datei nicht aktualisiert werden).

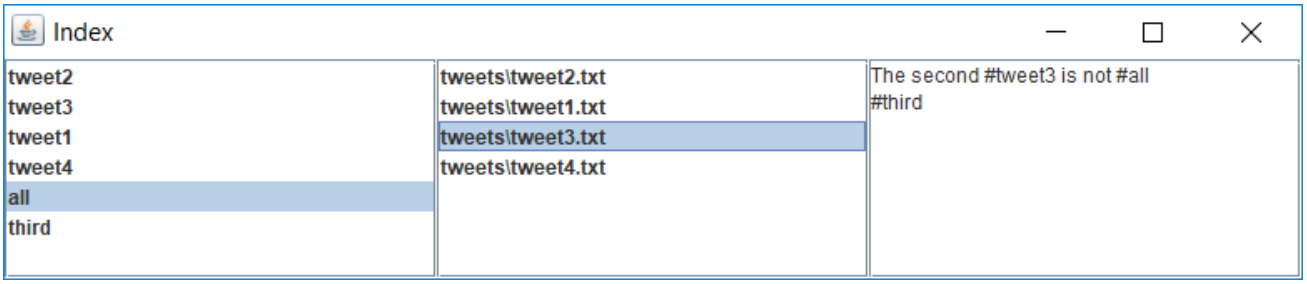

Abbildung 1: Beispielausgabe

Technische Anforderungen

Das Programm ist mit Java New IO und als Multi-Threading-Anwendung unter Verwendung eines ThreadPools zu lösen. Folgende Anforderungen sind bei der Realisierung des Programms zu berücksichtigen:

- Beim Start des Programms soll das Verzeichnis durchgegangen (FileWalk) und alle Textdateien analysiert werden. Dies baut den initialen Index auf.
- Setzen Sie dann zur Beobachtung des Verzeichnisses einen FileWatcher ein.
- Das Analysieren der Textdateien soll jeweils in einem eigenen Task erfolgen, der in einem ThreadPool ausgeführt wird.
- Das heißt auch, dass das Analysieren der Textdateien nebenläufig passiert und damit die Indexstruktur *threadsafe* ausgeführt werden muss. Entwerfen Sie daher eine Indexstruktur, die einen Thread-sicheren Zugriff erlaubt.
- Die Swing-Applikation soll jede Änderung im Index anzeigen. Beachten Sie, dass ein Update der UI immer im AWT-Thread erfolgen muss.

Architektur

Abbildung 2 zeigt die Multithreading-Architektur des Systems. Die einzelnen Komponenten sollen folgend arbeiten:

FileWalk: In einem ersten Thread werden die Dateien des zu durchsuchenden Verzeichnisses durchgegangen und für jede Textdatei ein Task zum Lesen und Analysieren abgesetzt. Nach dem Durchgehen des Verzeichnisses wird der Thread beendet.

FileWatcher: Wiederrum in einem eigenen Thread werden die Ereignisse, die über den WatchService gemeldet werden, behandelt. Es ist Folgendes zu tun:

- Neue Dateien müssen neu gelesen und analysiert werden.
- Für modifizierte Dateien müssen zuerst die entsprechenden Einträge im Index gelöscht und dann die Datei neu gelesen und analysiert werden.
- Für gelöschte Dateien müssen die entsprechenden Einträge im Index gelöscht werden.

Dieser Thread soll in einer Schleife bis zur Beendigung des Programm laufen.

AnalysisTask, RedoAnalysisTask, RemovePathEntriesTask: Die drei Tasks sollen die eigentliche Analyse und den Aufbau des Index durchführen. Sie werden auf Basis der beobachteten Änderungen im beobachteten Verzeichnisses generiert und einem ThreadPool zur Ausführung übergeben.

Index: Dieses Objekt speichert den Index mit den gefundenen Wörtern und den Dateipfaden zu den Wörtern.

Swing-Applikation: Die Liste der Wörter, die Dateipfade des selektierten Wortes und der Text der selektierten Datei werden angezeigt.

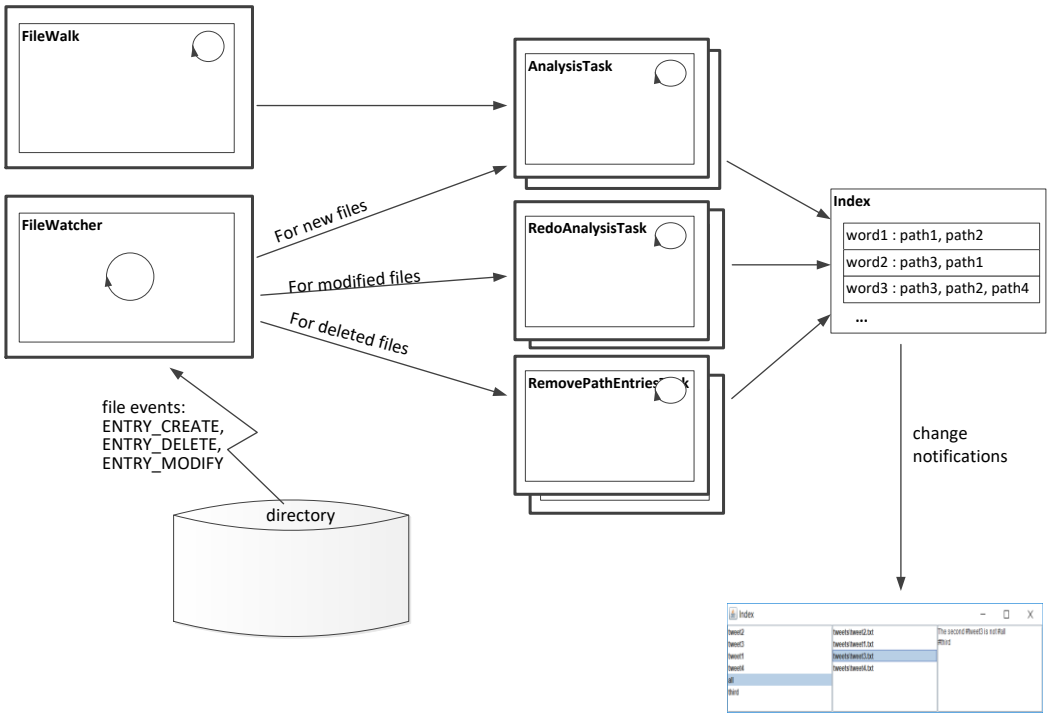

Abbildung 2: Architektur

Hinweise:

Achten Sie auf die korrekte und effiziente Synchronisation der Indexstruktur.

Achten Sie darauf, dass die Applikation richtig beendet wird, d.h., alle Threads terminieren und der ThreadPool niedergefahren wird.

Achten Sie darauf, dass Sie die vielen Ausnahmen, die passieren können, sinnvoll behandeln.

Verwenden Sie Logging, um das Verhalten der Applikation zu beobachten.

Beachten Sie, dass der FileWatcher einen Lock auf registrierte Verzeichnisse hält und damit die Datei-Manipulationen eingeschränkt sein können.