KV Betriebssysteme	Übung #6	SS 2005
Name:	Matr-Nr:	Gruppe:
Abgabe: 21.6.2005 (Alle Gruppen!)		Tutor:

Beispiel 8: Dateisystem-Synchronisation

Schreiben Sie ein Programm zur Simulation von Zugriffen auf ein Dateisystem. Ihr Programm soll den Betriebssystem-Teil simulieren, welcher den gemeinsamen Zugriff auf Dateien ermöglicht (Siehe Interfaces auf der LVA-Webseite). Es können beliebig viele Threads eine Datei gleichzeitig lesen, aber immer nur ein einziger Thread darauf schreiben (gleichzeitig kein Read-access erlaubt!). Die eigentliche Speicherung erfolgt über die Klasse RandomAccessFile; die Implementierung ist nur für Synchronisation und Verwaltung zuständig. Stellen Sie in ihrem Programm sicher, dass der gegenseitige Ausschluss (Leser vs. Schreiber) sowie beim Zugriff auf dieselbe Datei korrekt funktioniert.

Zum Testen schreiben Sie zwei Klassen: Einen Single-Thread und einen Multi-Thread Test.

Der Single-Thread Tester soll die verschiedenen Möglichkeiten (sofern ohne Deadlock möglich; z.B. eine Datei gleichzeitig zwei mal zum Lesen öffnen) testen. Dokumentieren Sie genau, welche Varianten geprüft werden!

Für den Multi-Thread Test erzeugen Sie 10 identische Threads, welche aus einer Menge von drei Dateien zufällig eine auswählen, um darauf entweder zu schreiben oder diese nur zu lesen (zufällig). Dies benötigt einige Zeit (Zufall aus Bereich 0,5-2 s). Anschließend wartet der Thread für eine Sekunde, bevor er wieder zu arbeiten beginnt. Um keinen globalen Gleichtakt zu erreichen, beginnen die 5 Thread jeweils mit 150 ms Verzögerung zueinander (d.h. Thread 1 beginnt sofort, Thread 2 nach 150 ms, Thread 3 nach 300 ms, etc.). Zur Visualisierung existiert noch ein weiterer Thread, welcher jede Sekunde den derzeitigen Zustand aller geöffneten Dateien (und nicht der Threads!) ausgibt. Dieser Thread darf zusätzliche Methoden des Betriebssystem-Interfaces (package-privat) verwenden, muss jedoch auch darauf achten, dass der Zustand exakt einem bestimmten einzigen Zeitpunkt entspricht und dadurch keine Fehler auftreten.

Bitte beachten Sie, dass die Beschreibung des Interfaces nicht alle Sonderfälle beinhaltet: Insbesondere bei "open" sind besondere Umstände zu beachten, die von Ihnen **explizit dokumentiert** werden müssen: Je nachdem, für welche der Alternativen Sie sich entschieden haben! Weiters sind im Interface Filesystem_API ev. zusätzliche Methoden (package-private oder öffentlich) notwendig, um ein (tatsächlich immer korrektes) Auslesen der aktuellen Daten zu erreichen (="Einfrieren" des Filesystem-APIs).

Abgabe elektronisch und auf Papier!