TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Aufgaben zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Aaron Tacke (tacke@in.tum.de)
http://db.in.tum.de/teaching/ss22/ei2/

Weiterentwicklung des Dateisystems

Aufgabe 1: Die Methode *ui()*

Laden Sie das Java-Projekt *SimpleFileSystem* herunter: `https://github.com/AaronTacke/SimpleFileSystem`

Dieses sollte Ihnen aus der Fragestunde bekannt vorkommen. Neben neuen ausgiebigen Kommentaren hat sich auch die Methode *ui()* in *Folder.java* geändert. Machen Sie sich mit der neuen Funktionsweise vertraut, und geben Sie an, wann und weshalb das Aufteilen einer großen Methode sinnvoll ist.

Lösung:

Während *ui()* in der Fragestunde noch eine einzige große Methode war, welche sowohl die while-Schleife als auch die Logik der einzelnen Befehle beinhaltete, wurde sie nun in verschiedene Methoden aufgeteilt:
ui() selber übernimmt nur noch das wiederholte auslesen der Konsole, bis `cd ..` eingegeben wurde.
Alle anderen Eingaben werden durch *handleInput(String[] input)* je nach Befehl an dedizierte Methoden weitergegeben.

Die Vorteile sind einleuchtend: Einerseits ist der bestehende Code nun leichter verständlich. Dies erleichtert das Suchen und Beheben von möglichen Fehlern. Andererseits ist das Hinzufügen von neuen Bestandteilen deutlich einfacher geworden, da man ohne großen Eingriff in die existierenden Methoden vorgehen kann.

Grundsätzlich ist es immer sinnvoll, zu große Methoden aufzuteilen, da Leserlichkeit, Verständlichkeit, Wartbarkeit, Flexibilität, und Testbarkeit nur so gewährleistet werden können.

Aufgabe 2: Befehl `help` hinzufügen

Ermöglichen Sie die Ausgabe der Benutzungs-Hilfe, welche bisher beim Start des Programms ausgegeben wird, mittels des Befehls `help`.

Achten Sie darauf, Code-Duplikate zu vermeiden! Geben Sie ein konkretes Beispiel an, weshalb das Vermeiden von Code-Duplikaten hier wichtig ist.

Lösung

`https://github.com/AaronTacke/SimpleFileSystem/commit/bac263ff8658d8bee161a9097737f11fa66efa77`

Falls man die Benutzungshilfe an mehreren Stellen im Code stehen hat (Code-Duplikat), muss man bei sämtlichen Änderungen stets darauf achten, die Benutzungshilfe an verschiedenen

Orten entsprechend abzuändern. Dies führt schnell zu Inkonsistenzen und ist grundsätzlich für den Entwicklungsprozess hinderlich (hinsichtlich Effizienz und Verlässlichkeit).

Wenn Sie einen Fehler in einem anderen Projekt beheben sollen, gehen Sie auch davon aus, dass Sie nach dem Umsetzen der Lösung nicht noch die gesamte Code-Basis durchsuchen müssen, um potenzielle Kopien des Fehlers erneut zu beheben.

Aufgabe 3: Befehl `clear DATEI` hinzufügen

Ermöglichen Sie das Entfernen des Inhalts einer Datei mittels des Befehls `clear`.

Achten Sie auf die richtige Anzahl von Argumenten, und darauf, dass clear nur für Dateien funktioniert.

Lösung

`https://github.com/AaronTacke/SimpleFileSystem/commit/776fc39db258f81b37592a5be59e72810d3bdc8a`

Aufgabe 4: Befehl `rm ELEMENT` hinzufügen

Ermöglichen Sie das Entfernen eines Elements mittels des Befehls `rm`.

Achten Sie darauf, dass Ihr Dateisystem stets konsistent bleibt. Eine Hilfsmethode kann der Übersichtlichkeit hier zuträglich sein.

Lösung

`https://github.com/AaronTacke/SimpleFileSystem/commit/0e5c0042dd97c284da6bab94a867791def72c191`

Aufgabe 5: Befehl `tree` hinzufügen

Ermöglichen Sie die rekursive Ausgabe aller Elemente im aktuellen Ordner in Baumstruktur auf der Konsole mittels des Befehls `tree`.

Die Ausgabe kann beispielsweise folgendes Format haben:

```
User
 |-Uni
 |   |-Semester1
 |   |   |-Paedagogik
 |   |   |   |-folien
 |   |   |   |-skript
 |   |   |   |-buch
 |   |   |-Sport
 |   |   |-Mathematik
 |   |   |   |-skript
 |   |   |-Vorkurs
 |   |-Semester2
 |-Privat
 |   |-liebesbrief
 |   |-Fotos
```

```
16 |   |   |-foto1
17 |   |   |-foto2
18 |-Arbeit
19 |   |-Bewerbung
20 |   |   |-motivationsschreiben
21 |   |   |-lebenslauf
22 |   |-Steuern
23 |   |   |-januar
24 |   |   |-maerz
25 |   |   |-februar
```

Tipp: Nutzen Sie eine rekursive Hilfsmethode zum Generieren der Ausgabe.

Bonus: Schauen Sie sich die *StringBuilder* Klasse an, um effizienter mit Strings arbeiten zu können

Lösung

`https://github.com/AaronTacke/SimpleFileSystem/commit/2c64be34ea677927735c01fbf6a8036e42f101dd`

Aufgabe 6: Befehl `mv ELEMENT PATH` hinzufügen

Ermöglichen Sie das Verschieben eines Elements mittels des Befehls `mv`.

Testen Sie Ihre Java-Fähigkeiten und Kreativität mittels dieser Aufgabe, denken Sie jedoch stets daran, dass Ihr Code möglichst übersichtlich, verständlich und korrekt bleiben soll.

Nutzen Sie folgende Anregungen um Ihren `mv`-Befehl zu gestalten:

- Erlauben Sie das Verschieben in einen Unterorder (z.B. `subfolder/subsubfolder/` als PATH), sowie in einen Parent-Ordner (z.B. `../` als PATH).
- Erlauben Sie das Verschieben in arbiträre Ordner, solange diese existieren (z.B. `../../test/subtest/` als PATH).
- Gehen Sie sicher, dass Objekte nach dem Verschieben stets nur einmal vorhanden sind.
- Gehen Sie sicher, dass kein Ordner mehrere gleichnamige Objekte enthält.
- Prüfen Sie, dass Ordner nicht Teil von sich selbst werden können (dies würde die Baumstruktur des Dateisystems zerstören).
- Erlauben Sie absolute Pfade (beginnend mit `/`), welche einen Ordner beginnend bei der Wurzel des Dateisystems beschreiben.
- Erlauben Sie das Umbenennen von Dateien, falls am Ende des Pfades ein anderer Name angegeben ist.
- Fügen Sie Regeln hinzu, um die Benennung von Dateien sinnvoll einzuschränken, und setzen Sie diese durch.

Lösung

`https://github.com/AaronTacke/SimpleFileSystem/commit/4d5b8147fa2aa00c18fcbce657346446f11caa68`

Aufgabe 7: Befehl `cp ELEMENT PATH` hinzufügen

Ermöglichen Sie das Kopieren eines Elements mittels des Befehls `cp`. Nutzen Sie dafür die Implementierung von `mv` (entweder die gegebene Lösung oder Ihre eigene).

Achten Sie darauf, dass kopierte Elemente tatsächlich vollständig kopiert sind, also beispielsweise die Dateien aus kopierten Ordnern verschiedene Inhalte haben können.

Lösung

`https://github.com/AaronTacke/SimpleFileSystem/commit/68c4f444fc8537223926ca94f3f097dee089595f`

Aufgabe 8: Testen und Spaß haben

Testen Sie Ihre Implementierung ausführlich. Seien Sie kreativ, um mögliche Randfälle abzufangen.

Herzlichen Glückwunsch, Sie haben nun ein funktionierendes Dateisystem entwickelt!