Medieninformatik(B): Funktionale und Reaktive Programmierung, Prof. Knabe, 2014-12-01

Aufgabe FileServ: Lazy Evaluation und Streams

Gegeben sind der Scala-Trait `FileService` und der Testtreiber `FileServiceTest`.

```
/**Trait for lazy File operations.*/
trait FileService {

  /**Opens a file for reading.*/
  def open(filename: File)

  /**Gets a Stream for lazily reading the File managed by this service. Each line will become an element of the Stream.*/
  def getStream(): Stream[String]

  /**Gets a List for immediately reading the File managed by this service. Each line will become an element of the List.*/
  def getList(): List[String]

  /**Closes this file service. Should be invoked after usage in order to free resources.*/
  def close()

}
```

Implementieren Sie diesen Trait in einer Klasse `FileServiceImpl`.

Testen Sie ihn mittels des mitgelieferten Testtreibers.

Welcher Vorteil ergibt sich hier aus der Verwendung eines `Streams` statt einer `List`?

Hinweise

- Zeilenweise lesen können Sie mit einem `BufferedReader`, den Sie auf einen `FileReader` ansetzen.
- Speichern Sie den `BufferedReader` als Attribut Ihrer `FileServiceImpl`.
- In `getStream` bauen Sie den `Stream` rekursiv mittels `Stream.cons` auf.
- Beim Rekursionsabbruch liefern Sie `Stream.empty`.
- In `getList` lesen Sie in einer Schleife alle Zeilen der Eingabedatei ein. Sie können dabei schrittweise eine `List` aufbauen, die jedoch dadurch inverse Reihenfolge hat. Diese können Sie einfach mittels Methode `reverse` in die richtige Reihenfolge transformieren.
- Die **`while`**-Schleife von Scala hat exakt die gleiche Syntax und Verhalten wie in Java. Es gibt jedoch keine **`break`**-Anweisung. Wenn die Schleife aber Hauptbestandteil einer Methode ist, kann man sie stattdessen durch **`return`** abbrechen.