Wir tanzen einen Datenstrom – Aktive Visualisierung einer Filterkette

Markus Dahm

Fachbereich Medien, FH Düsseldorf

Zusammenfassung

Eigentlich sind Ein- und Ausgabestreams (z.B. in Java) ganz einfach – wenn man denn endlich verstanden hat, dass die Streams nach dem Lego- oder Pipeline-Prinzip aus mehreren spezialisierten Filter-Objekten zusammengebaut werden. Trotzdem haben in jedem Semester viele Studierende die gleichen Schwierigkeiten, das zu verstehen; vor allem das Lesen von Daten über viele Filter hinweg.

Daher wird versucht, das Prinzip des Datenstroms zusätzlich zu seiner symbolischen Darstellung (z.B. in UML) durch „antropomorphe Modellierung“ zu veranschaulichen: Jedes Filter-Objekt wird von einem oder einer Studierenden dargestellt. Verkettungen werden durch Handauflegen verdeutlicht, Methodenaufrufe werden durch Übergeben von Papierzetteln verdeutlicht. Das sieht hinreichend lustig aus, dient aber nach vielen Rückmeldungen nicht nur der Unterhaltung, sondern führt auch zu einem richtigen (nicht nur besseren) Verständnis des Aufbaus und Ablaufs von Datenströmen.

1 Aufforderung zum Tanz

Datenströme in Java und anderen Umgebungen können durch Filter modifiziert werden. Für jede Funktionalität wird ein spezieller Filter definiert. Um einen Datenstrom auf mehrfache Art zu modifizieren, werden passenden Filter hintereinander geschaltet. Immer wieder haben viele Studierende Probleme bei der Durchdringung dieses Konzepts. Vor allem das Lesen durch mehrere Filter hindurch erscheint verwirrend, da die Richtungen von Datenfluss, Methodenaufruf und Assoziation von Objekten nicht gleich sind (siehe Bild 1).

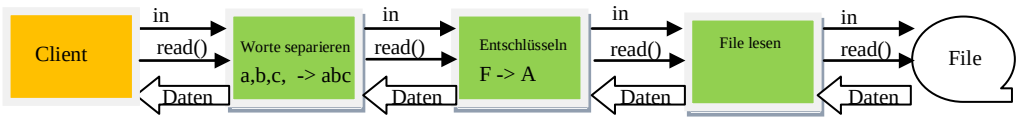

Bild 1 – Lesen durch eine Filter-Pipeline (von oben: Assoziation, Methodenaufruf, Datenfluss)

2 Tanz der Stromdaten

Um ein besseres „Gefühl“ und ein vollständigeres Verständnis für das Filterkonzept, die Verkettung der Objekte, für die Methodenaufrufe, für die Verwendung der Parameter und schließlich für den Fluss der Daten zu vermitteln, wird der Aufbau der Filterkette und der Ablauf des Schreibens und Lesens sowie der Datenfluss durch Studierende selber dargestellt.

2.1 Ausstatten

Die Übertragung der abstrakten Konzepte „Objekt“, „Methode“, „Parameter“, etc. in die anfassbare reale Welt soll so einen zusätzlichen Weg zum Verständnis eröffnen. Die Mittel dafür sind überall verfügbar, preiswert und sofort verständlich:

- Jedes Filter-**Objekt** sowie das Client-Objekt wird durch eine/n **Studierenden** repräsentiert. Die Klasse jedes Objekts steht auf einem eigenen **Namensschild** (s. Bild 2)**.**
- Jede **Verkettung** (`in` oder `out`) wird durch einen **Arm** dargestellt. Die Finger zeigen die Navigationsrichtung an (s. Bild 2).
- Jeder **Methodenaufruf** wird nachgestellt durch das Überreichen eines Blattes **Papier** mit dem Methodennamen (s. Bild 3).
- **Parameter** und Rückgabewerte werden live auf **post-it**s geschrieben und an die passenden Stellen auf dem Methodenaufruf, in den Puffer oder das File geklebt.
- **Puffer-Arrays** und Files werden durch entsprechend beschriftete **Blätter** dargestellt.

2.2 Aufstellen

Die Objekte werden in der richtigen Reihenfolge aufgestellt. Mit den Armen werden die `in`-Verkettungen (`in` ist der Verweis auf den Filter, von dem gelesen werden soll) dargestellt. Nur entlang einer dieser Verkettungen kann eine Methode aufgerufen werden (siehe Bild 2).

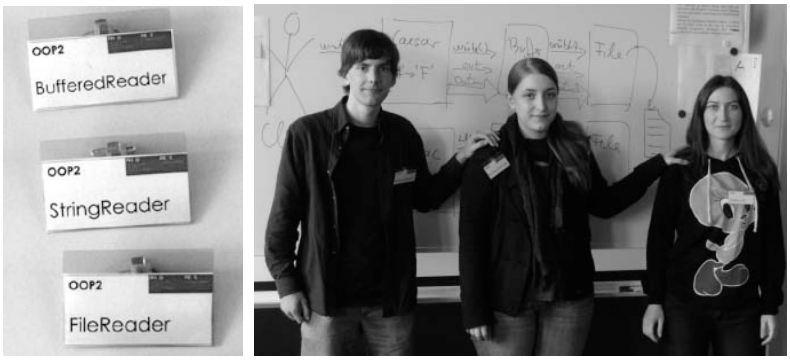

Bild 2 – Namensschilder für Klassen, Aufstellung der Filter-„Objekte“ mit Darstellung der Verkettung

2.3 Abtanzen

Der Client (das „Objekt“ ganz links) möchte mit Hilfe der Filterkette aus dem File (ganz rechts) lesen. Dazu ruft er eine Methode `read`() des Filters auf, der ihm am nächsten steht. Das tut er, indem er ihm das Blatt mit der Beschriftung `read`() gibt. Der behält (!) dieses Blatt und gibt seinerseits ein weiteres Blatt mit `read`() an den nächsten Filter. So ruft jeder Filter die Methode `read`() des Objekts auf, auf den seine `in`-Variable zeigt – so lange, bis der FileReader ein oder mehrere Zeichen aus dem File (Blatt Papier) entnehmen kann.

- Es ist wichtig, dass nicht das gleiche Blatt mit `read`() weitergegeben wird, sondern jeweils ein neues Blatt (siehe Bild 3). So wird der falsche Eindruck vermieden, es würde eine Methode weitergegeben, statt immer eine neue Methode aufzurufen.

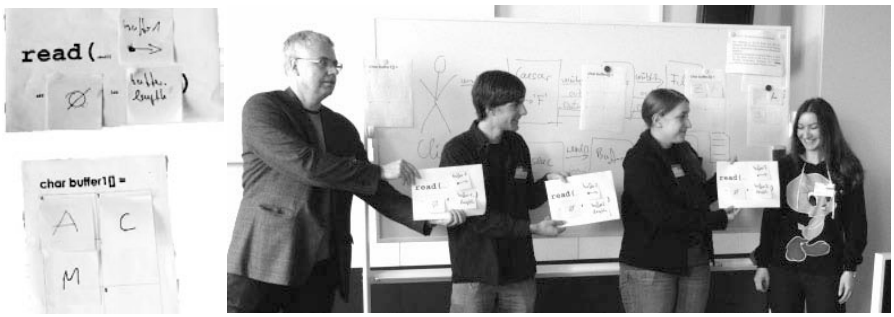

Bild 3 – Aufruf von Methoden mit Parametern – `in.read(buff,off,len)`

Ist die `read`()-Stafette beim FileReader angekommen, liest dieses Objekt einen Wert aus dem File, schreibt den auf ein neues post-it und gibt ihn als Rückgabewert zusammen mit dem Blatt des Methoden-Aufrufs zurück an den Aufrufer. Der verarbeitet diesen gelesenen Wert entsprechend seiner spezifischen Aufgabe (z.B. Ver/Entschlüsselung, Groß/klein-Wandlung, Pufferung, …) und gibt den verarbeiteten Wert seinerseits auf einem neuen post-it zurück. Diese Verarbeitung und Rückgabe geschieht solange bis endlich der Client eine Rückgabe bekommt.

- Es sollten immer neue post-its mit neuen Parametern geschrieben und auf die Methoden-Blätter geklebt werden. Das live-Schreiben verdeutlicht, dass diese Daten und Werte zur Laufzeit ermittelt werden. Außerdem muss für jedes post-it erarbeitet werden, welcher Wert übergeben soll. So kann auch das Auditorium immer wieder einbezogen werden.

2.4 Standardtänze

Auf die Art und Weise wie oben für das Lesen beschrieben wird zuerst das Schreiben durch eine Filterkette dargestellt, danach das Lesen. Später noch das Leeren der Filterkette mit `flush`() und das Schließen der Filterkette mit `close`(). Außerdem kann so auch das Erzeugen der einzelnen Filterobjekte und das schrittweise Verketten visualisiert werden.

3 Erfahrungen vom Parkett

3.1 Mit langsamem Takt beginnen

Natürlich sollte nicht mit den komplexesten Filteralgorithmen und schwierigsten Methoden (siehe Abschnitt 3.2) angefangen werden, sondern zuerst mit einfachen Aufgaben die generellen Konzepte vermittelt werden. Der Zweck der Filterkette liegt in der Funktion und nicht in ihrem Aufbau – der Ablauf sollte also vor dem Aufbau behandelt werden.

Das Schreiben ist einfacher zu verstehen als das Lesen, es geschieht logisch zuerst, damit sollte also begonnen werden. Beim Lesen sollte zunächst die einfache `read`()-Methode verwendet werden, die ein gelesenes Zeichen zurückgibt, um den Ablauf zu visualisieren. Wenn dieser verstanden wurde, kann zur `read(buff,off,len)`-Methode übergegangen werden, bei der ein Puffer übergeben wird, der vom Filter beschrieben werden soll.

Es ist nach meiner Erfahrung außerdem eine gute Idee, beim ersten Durchlauf nicht alle Feinheiten bis ins Letzte korrekt zu behandeln. Es ist z.B. nicht nötig, einen Zeiger in das File vollständig zu verwalten. Diese Einzelheiten würden vom konkreten Teilziel (Verständnis des Aufbaus oder des Ablaufs der Methodenaufrufe) ablenken und sollten daher zunächst unbeachtet bleiben.

Auch, welche Datentypen genau verwendet werden, wie das File geöffnet und geschlossen wird, was im Fehlerfall passiert und andere Details sollten zunächst gar nicht behandelt werden. Diese, natürlich wichtigen, Details sollten im Nachgang besprochen werden – was den Studierenden natürlich auch so klar gemacht werden muss.

3.2 Fußfehler

Eines der größten Verständnisprobleme liegt darin, dass beim *Lesen* die Zeichen auch *geschrieben* werden sollen – nämlich dann, wenn `read()` ein Puffer übergeben wird. Damit tut sich meiner Erfahrung nach mindestens ein Drittel der Studierenden sehr schwer.

- In diesem Zusammenhang kann und soll auch ein Prinzip der **Verantwortung** für Objekte, die auf dem Heap erzeugt werden verdeutlicht werden: Wer ein Objekt erzeugt muss auch dafür sorgen, dass es wieder gelöscht, bzw. dem garbage collector vorgeworfen werden, werden kann. Das ist notwendig, da nur im erzeugenden Kontext entschieden werden kann, wann das erzeugte Objekt nicht mehr gebraucht wird.

Der Client erzeugt einen Puffer, ein char[], repräsentiert durch ein vorbereitetes Blatt Papier. Die Referenz auf dieses Array wird als Pfeil auf ein post-it geschrieben und als Parameter auf ein weiteres Blatt Papier mit einem `read(buf,off,len)` Aufruf geklebt (s. Bild 3).

- Auch hier können grundlegende Konzept nochmals wiederholt werden: Es wird nicht (!) das Blatt mit dem Array übergeben sondern nur eine **Referenz** darauf. Das Blatt bleibt beim Client-Objekt. Die anderen Objekte können bei der Ausführung ihrer `read`-Methoden über die Referenz darauf zugreifen.

3.3 Bein stellen - Fehler provozieren

Da man vor allem aus Fehlern lernt, auch aus denen anderer, versuche ich auch immer wieder Fehler zu provozieren. Die falsch gegebenen Antworten sind immer wieder sehr nützlich und treiben den Lernprozess weiter: Sie weisen direkt auf noch nicht Verstandenes hin, auf Missverständnisse, auf untereinander verbreitete oder aus „dem Internet“ übernommene Irrtümer. Wenn diese erkannt werden, können sie sofort korrigiert werden.

Eine gute Gelegenheit, Fehler aufzudecken ist z.B. die Diskussion der Richtung der Verkettung zwischen den Filter-Objekten. Stellt man die suggestive Frage „Die Daten sollen vom File zum Client transportiert werden. In welche Richtung zeigen also die Assoziationen?“, so wird sehr wahrscheinlich die falsche Richtung vom File zum Client geantwortet ☺.

3.4 Abschlussschritt

Die aktive Visualisierung der Datenströme ist eine Abwechslung im sonst eher konventionellen Ablauf der Vorlesung OOP. Das Feedback am Ende des Semesters zeigt, dass die meisten Studierenden diese Abwechslung begrüßen. Darüber hinaus zeigen die Klausurergebnisse, dass mehr Studierende als vorher diese Aufgabe korrekt bearbeiten können.

Mein Eindruck ist, dass die Beteiligung und Aufmerksamkeit sowohl der „Objekte“ als auch des Auditoriums gegenüber der sonst üblichen Vorlesung erhöht ist. Allerdings muss hier darauf geachtet werden, dass die Aktion nicht zu lange dauert. Wenn ein kompletter Datenfluss zwei- oder dreimal durchlaufen wurde, sollte die Aktion langsam beendet werden. Andernfalls lässt das Interesse aller Beteiligten sehr rasch nach.

Zwischen einem „Tanz“ zum Schreiben und einem zum Lesen von Daten sollte entweder eine Pause, ein Theorieteil oder eine ganze Vorlesung liegen, um durch den Wechsel der Lehrform die Aufmerksamkeit und Lernfähigkeit der Studierenden hoch zu halten.

Entwickelt hat sich diese Lehrform aus der aktiven Vermittlung von Algorithmen über zusammengesetzte Datenstrukturen, z.B. Einfügen, Suchen, Löschen in Bäumen und Listen, auch mit rekursiven Methoden. Ab dem 1. Semester werde ich so den Ablauf von Methodenaufrufen grundsätzlich vermitteln, um später, z.B. bei Trees und Streams darauf aufzubauen.

Anregungen zur Verbesserung sind immer willkommen.

Literatur

Accelerated Learning (2011). www.acceleratedlearning.com, www.alcenter.com

AlgoRythmics (2011). http://www.youtube.com/user/AlgoRythmics#p/u

Hubwieser (2007). *Didaktik der Informatik*, Springer

Schröder (2002). *Lernen - Lehren – Unterricht. Kapitel Aktivierender Unterricht*. Oldenbourg

Kontakt Informationen

Prof. Dr.-Ing. Markus Dahm
FH Düsseldorf
Fachbereich Medien

WWW: www.medien.fh-duesseldorf.de/dahm
E-Mail: markus.dahm@fh-duesseldorf.de