**Pipes & Filters Architecture**

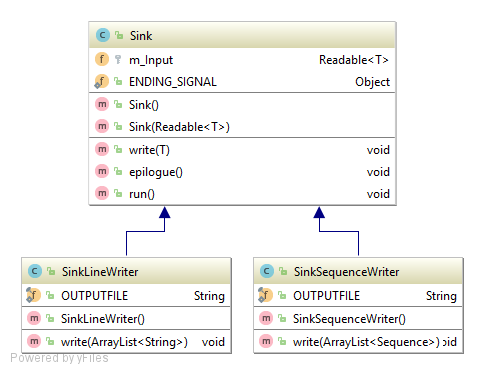
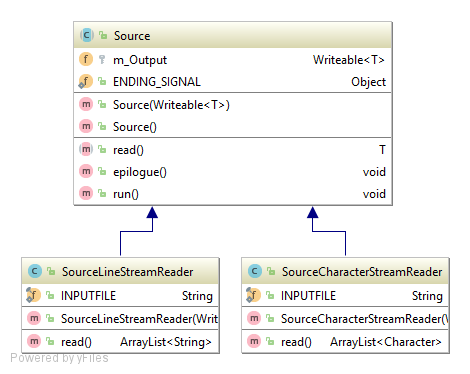
**Julia Schatzmann und Christina Tschol**

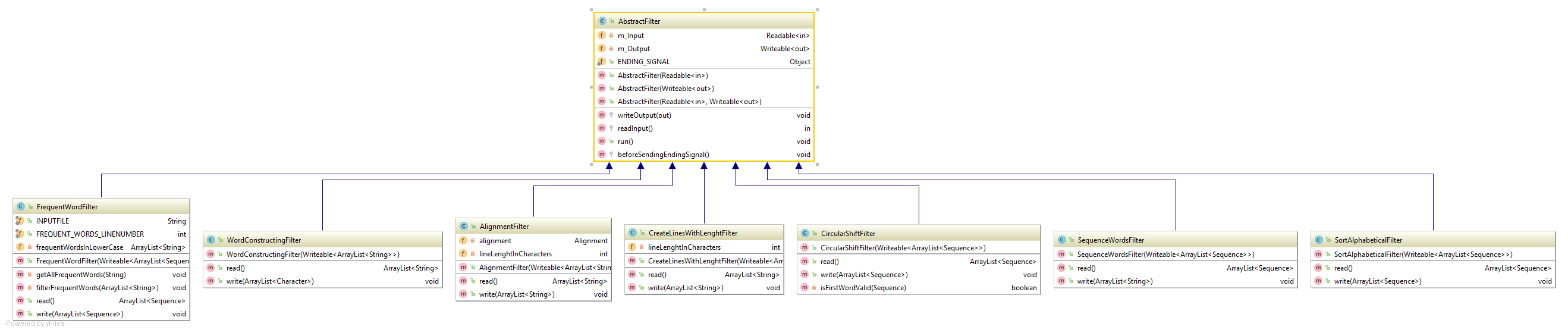
**Dokumentation**

## Design Dokumentation

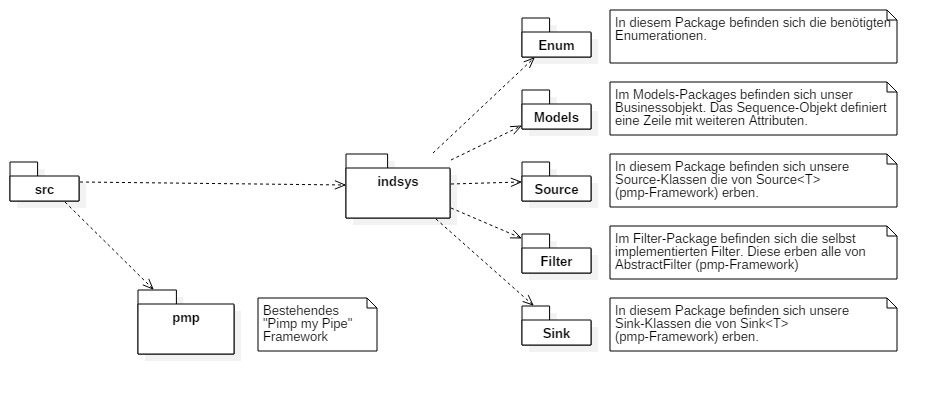
**Klassendiagramme**

Source-Readers:

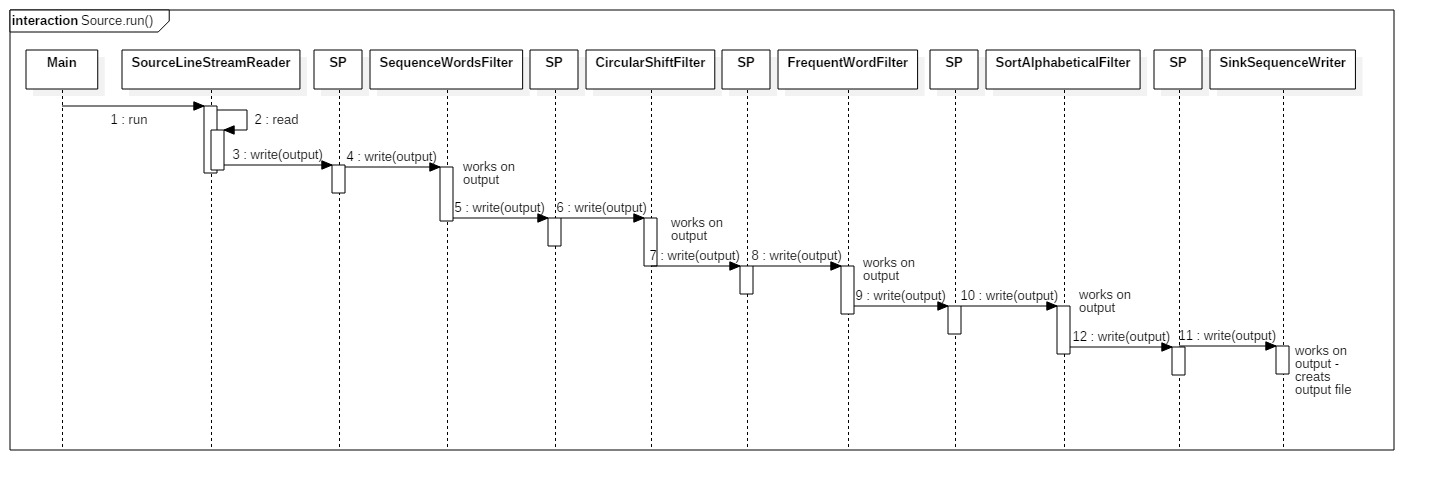
Sink-Writers:

Filters:

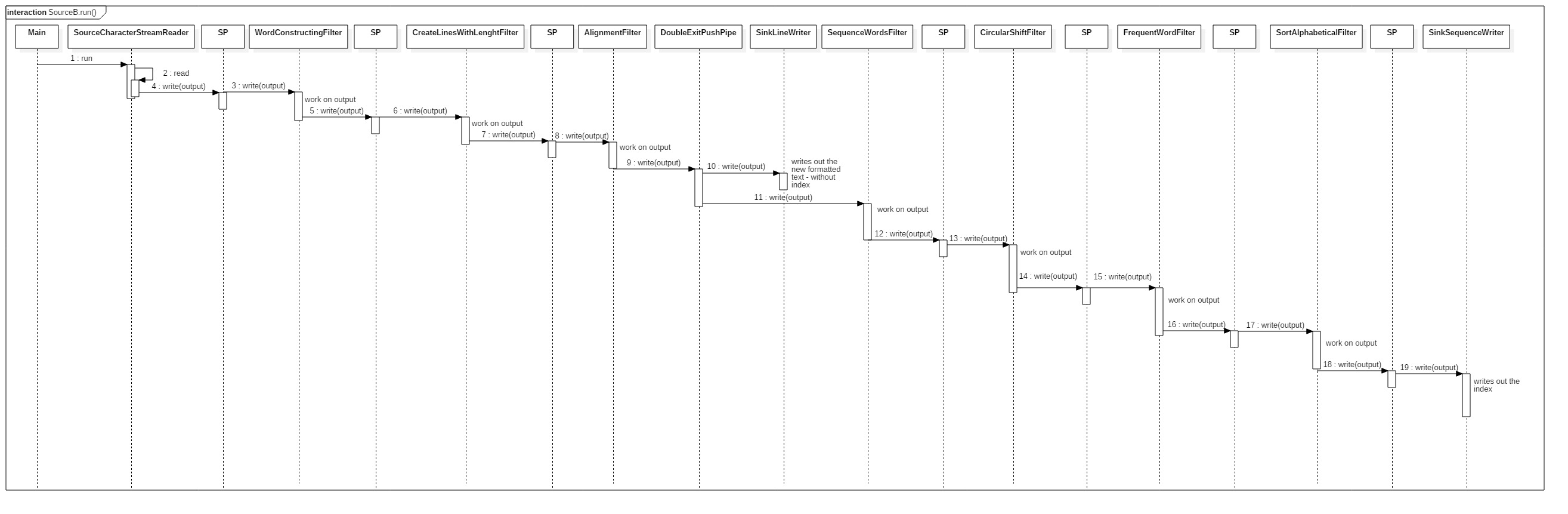
**Paketdiagramm:**

****

**Sequenzdiagramm Aufgabe A)**



**Sequenzdiagramm Aufgabe B)**



## Beschreibung

Für die Umsetzung konnte ein vorgegebenes pmp-Framework verwendet werden. Dort wurde vor allem das Composite-Pattern ausgenutzt. Nach dem der Benutzer eingegeben hat, in welchem Modus er das Programm mit ggf. Attributen ausführen möchte, wird der „Sourcereader“ aufgerufen (die „run“-Methode). Anschließend wird jedem Filter eine Pipe und jeder Pipe einen Filter übergeben. Somit haben wir einen „aktiven“ Push-Filter realisiert, bei dem die Source durch alle Filter „durchgepushed“ wird, bis dann zum Schluss der „Sinkwriter“ die Darstellung/Ausgabe übernimmt.

## Uninteressante Wörter nicht beachten:

### Um uninteressante Wörter nicht im Index abzubilden (als erstes Wort), haben wir einen eigenen Filter dafür implementiert. Bei der Erstellung ([Konstruktoraufruf](https://www.java-forum.org/thema/konstruktoraufruf.82519/)) des Filters wird ein File eingelesen, in dem häufig vorkommende Wörter befinden, und gespeichert. Nachdem der CircularShiftFilter den Output mit den „Zeilen-Objekten“ weitergegeben hat, wird im FrequentWordFilter jedes Zeile-Objekt geprüft ob das erste Wort der Zeile ein uninteressantes ist. Wenn ja wird das Objekt entfernt.