Lösungsansatz

Für die Lösung der Aufgabe wurden Filter und Pipe Klassen erstellt. Dabei wurde auf eine strikte Trennung der Push – Pipeline und er Pull – Pipeline geachtet.

**Push – Pipeline**

Für die Push – Pipeline wurden die Packages *at.fhv.sysarch.lab3.pipeline.pushPipeline* und *at.fhv.sysarch.lab3.pipeline.pushFilter* erstellt. In den Packages befinden sich die erstellten Klassen und Interfaces für die Push – Pipeline.

Für die Pipeline wurde gegen ein Interface programmiert. Dabei wurde auf einen generischen Ansatz gesetzt. Dadurch ist es im späteren Verlauf der Pipeline möglich auch mehrere Daten zu übertragen, was auch nötig ist, um die Faces zusammen mit einer entsprechenden Farbe weiterzuleiten. Um die ein Face zusammen mit einer Farbe in einem Objekt zu bündeln, wurde die Klasse *Pair* im Package *…pipeline.data* verwendet.

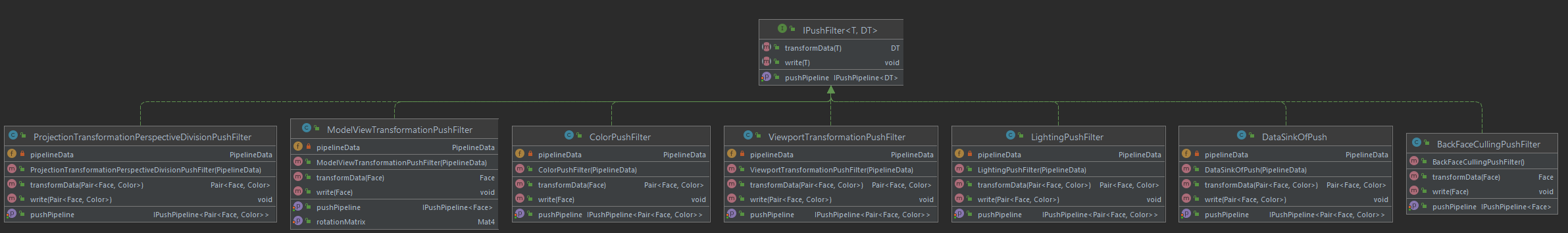
Ein Bild, das Text, draußen, Screenshot enthält.

Automatisch generierte Beschreibung

Die Pipeline hat den nächsten Filter, in welchen sie schreiben, gespeichert.

Für die Filter wurde ebenfalls gegen Interfaces programmiert. Dabei wurde auch ein generischer Ansatz verwendet. Da der zweite generische Datentyp beim Start der Pipeline nicht bekannt ist, wurde ein Wildcard verwendet.

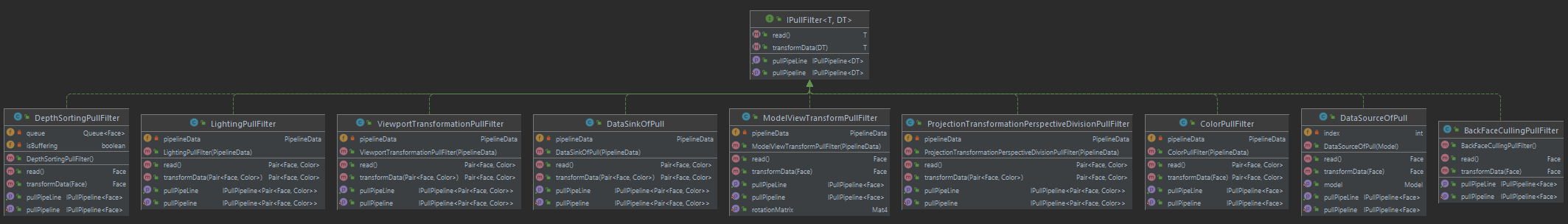
Die Filter haben Jeweils die Pipeline, in welche sie schreiben ebenfalls gespeichert. Diese können nach dem Erzeugen der Filter gesetzt werden. Jeder Filter hat zusätzlich noch eine „transform“ Methode, in welcher die Umwandlung bzw. die Bearbeitung der Daten stattfindet.

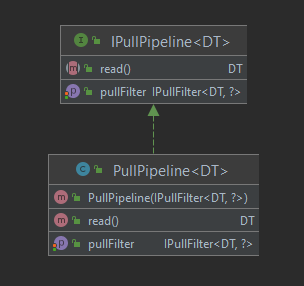


In der Push – Pipeline Factory werden die Filter und die Pipes erstellt und miteinander verbunden. Bei der Rendering Methode werden dabei immer die Daten in den ersten Filter geladen.

**Pull – Pipeline**

Bei der Pull – Pipeline gibt es nicht besonders viel unterschied, die Bearbeitung der Daten ist grundsätzlich identisch jedoch kann das Depthsorting besser eingesetzt werden. Für das Aktualisieren der View wird die Sink getriggert damit diese aus der vorherigen Pipe die Daten liest. Das setzt den gesamten Prozess in Gang. Bei der Pull – Pipeline gibt es noch eine Data Source welche die Daten zurückliefert. Wenn das Ende des Datenstromes erreicht ist, wird ein Marker Face gesendet. Die Filter prüfen in den jeweiligen Methoden ob ein ende des Streams erreicht ist um nicht in eine Null – Pointer Ausnahme zu laufen.





Grundsätzlich können die Filter auch für Pull und Push Pipeline verwendet werden. Dazu müsste sie nur das dementsprechende Interface implementieren. Jedoch wurden die Systeme strikt getrennt, um das Beispiel einfacher zu halten. Dadurch wird auch das Verständnis für die einzelnen Pipes besser.