# Lab 3 – Pipes and Filters

## How to start the project

To start the project open it in IntelliJ, go to the gradle section on the right, select application and click on run.

To switch between pull and push pipeline, simply go to the main class and change the value of **`USE_PUSH_PIPELINE`** to true or false.

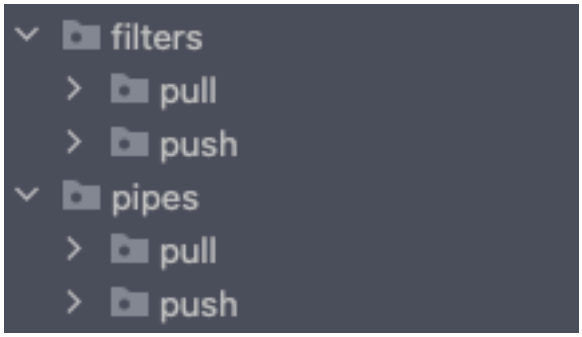
```
public class Main extends Application {
    9 usages
    private final static int VIEW_WIDTH  = 860;
    9 usages
    private final static int VIEW_HEIGHT = 540;

    1 usage
    private final static int SCENE_WIDTH  = VIEW_WIDTH * 2;
    1 usage
    private final static int SCENE_HEIGHT = VIEW_HEIGHT * 2;

    1 usage
    private final static boolean USE_PUSH_PIPELINE = true;
```

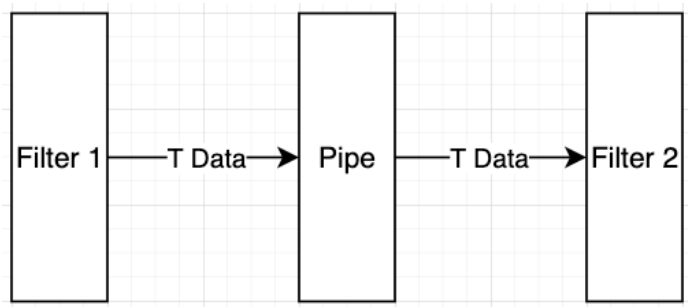
## Structure

We divided the pipes and filters in two packages, and in these packages, we divided them in push and pull packages.



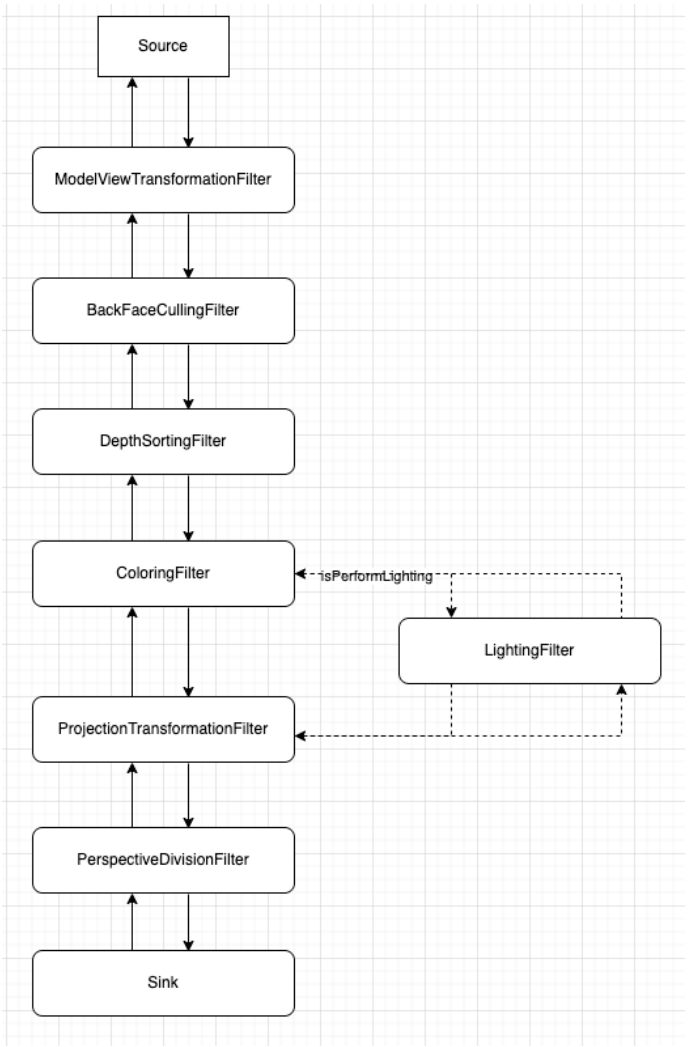
## Pipes

A pipe is in between two filters, where it gets data from the previous filter and gives it to the next. The interface uses generics as input and output, so it can be used for all filters of the application.




## Pullpipe

The arrows represent the pipes.




The pullpipe needs to know its predecessor and must call the **`pull()`** and **`hasNext()`** method of the predecessor.

The **`hasNext()`** is needed because the filters ask their predecessor if they have data to pull. Also the sorting needs this method to collect all the faces.

## Pushpipe

The arrows represent the pipes.

The pushpipe needs to know its successor and must call the **`push()`** method of the successor.