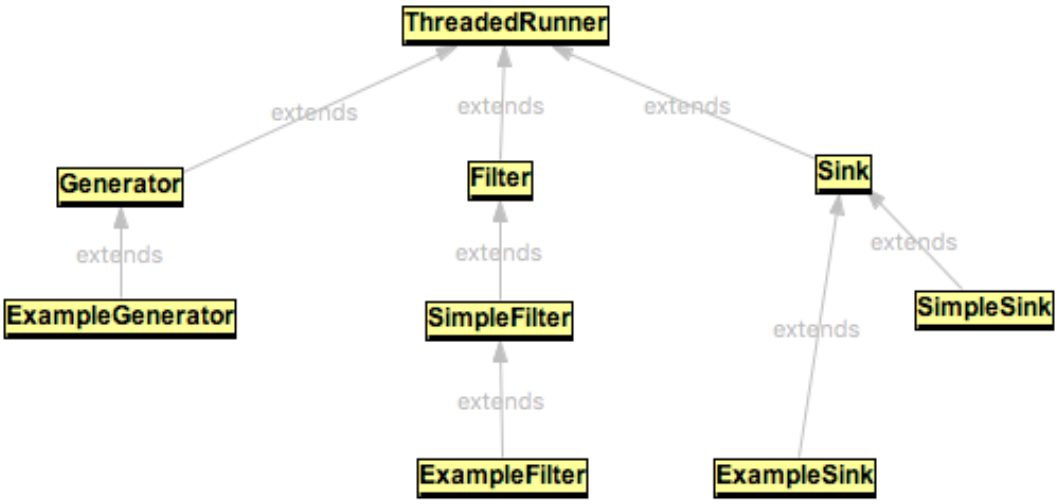
Pipe and Filter Pattern

1. Pipe and Filter 패턴

입력된 데이터 스트림을 변환하는 경우에 사용하며, 입력된 데이터를 처리하는 단계를 순차적으로 처리한다. 하나의 필터에서 처리된 결과물은 다음 필터의 입력물이 되며 각각의 처리 단계를 파이프로 연결한다.

2. Pipe and Filter 패턴 구조

각 필터는 각각의 쓰레드로 돌아가는 구조이다. 상속 구조를 나타내면 아래와 같다.



3. Pipe and Filter 패턴 사용

Pipe and filter를 사용하는 메인 함수는 다음과 같다[1].

```
public class ExampleRunner {
    public static void main(String[] args) {
        // create pipes
        final Pipe<Integer> in = new PipeImpl<Integer>();
        final Pipe<String> out = new PipeImpl<String>();

        // create components that use the pipes
        final Generator<Integer> generator = new ExampleGenerator(in);
        final Filter<Integer, String> filter = new ExampleFilter(in, out);
        final Sink<String> sink = new ExampleSink(out);

        // start all components
        generator.start();
        filter.start();
        sink.start();

        System.out.println("runner finished");
    }
}
```

[1] https://gist.github.com/roryokane/9606238

4. Pipe and Filter 패턴 활용

(1) 제공하는 소스는 각 필터를 지나가면서 타입을 정수에서 문자열로 변환할 뿐, 값을 변경하지 않는다. 필터를 지날 때 값이 10배로 커지도록 해보자.

(2) 필터는 단일한 역할을 수행해야 한다. 문자열로 변환하는 필터와는 별도로 값이 10배 커지는 필터를 만들어 두 개의 필터를 연결하여 (1)과 동일한 결과가 나오도록 해보자.

(3) 어떻게 파이프 앤 필터의 구조로 상기와 같은 작업이 가능한지 코드를 분석하여 설명해 보자.