4.21作业

1. 基于代码实例FileSourceExample以及data数据源打印出inflow大于4且LogStore为LogStore-1的

projectname。

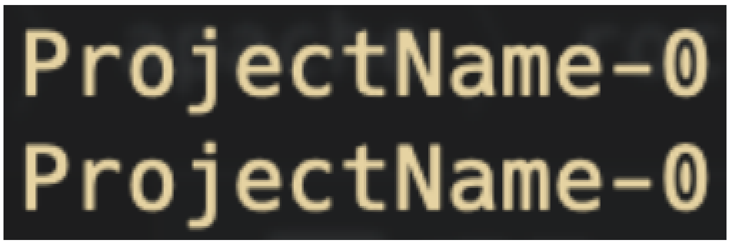
源文件内容：

```
{"InFlow":"1","ProjectName":"ProjectName-0","LogStore":"LogStore-0","OutFlow":"0"}
{"InFlow":"2","ProjectName":"ProjectName-1","LogStore":"LogStore-1","OutFlow":"1"}
{"InFlow":"3","ProjectName":"ProjectName-2","LogStore":"LogStore-2","OutFlow":"2"}
{"InFlow":"4","ProjectName":"ProjectName-0","LogStore":"LogStore-0","OutFlow":"3"}
{"InFlow":"5","ProjectName":"ProjectName-1","LogStore":"LogStore-1","OutFlow":"4"}
{"InFlow":"6","ProjectName":"ProjectName-2","LogStore":"LogStore-2","OutFlow":"5"}
{"InFlow":"7","ProjectName":"ProjectName-0","LogStore":"LogStore-0","OutFlow":"6"}
{"InFlow":"8","ProjectName":"ProjectName-1","LogStore":"LogStore-1","OutFlow":"7"}
{"InFlow":"9","ProjectName":"ProjectName-2","LogStore":"LogStore-2","OutFlow":"8"}
{"InFlow":"10","ProjectName":"ProjectName-0","LogStore":"LogStore-0","OutFlow":"9"}
```

代码处理：

```java
public class FileSourceExample {
 public static void main(String[] args) {
 DataStreamSource source = StreamBuilder.dataStream("namespace",
"pipeline");
 source.fromFile("data.txt", true)
 .map(message -> message)
 .filter(message -> ((JSONObject)message).getInteger("InFlow") > 4 &&
 ((JSONObject)message).getString("LogStore").equals("LogStore-
1"))
 .map(message -> ((JSONObject) message).getString(key: "ProjectName"))
 .toPrint(1)
 .start();
 }
}
```

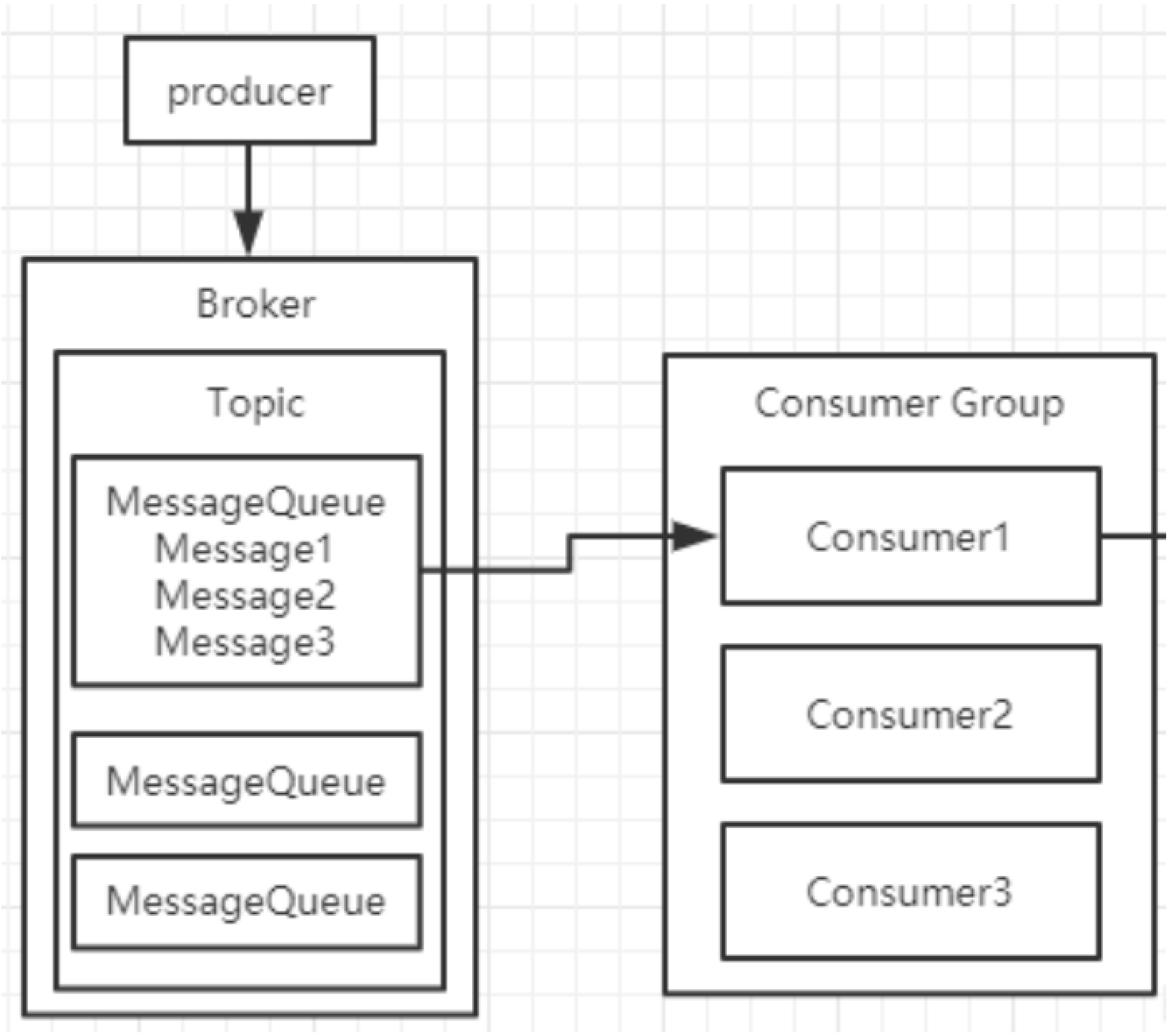
结果：



```
ProjectName-0
ProjectName-0
```

2. 如果消息处理需要基于**事件时间**进行处理，那么对于乱序的消息窗口需要添加什么额外的设计?

答：只要**将消息按照顺序都放入同一个MessageQueue中**，最后就能被同一个消费者顺序消费了。


producer
Broker
Topic
MessageQueue
Message1
Message2
Message3
MessageQueue
MessageQueue
Consumer Group
Consumer1
Consumer2
Consumer3