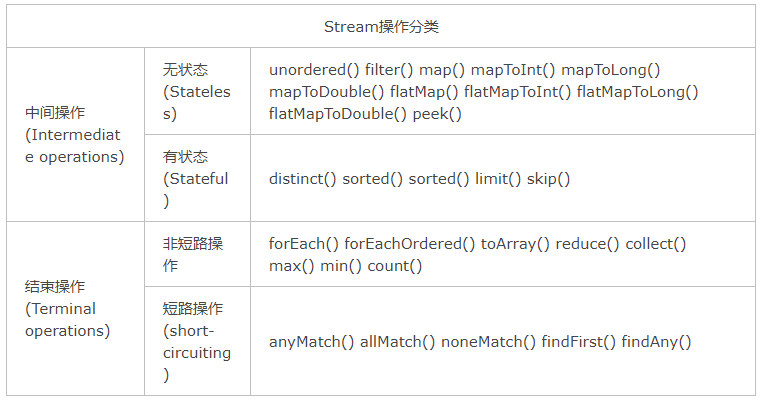
**介绍**



Stream的核心：Stream、Sink。

Sink的核心方法：begin()、end()、cancellationRequested()、accept()。

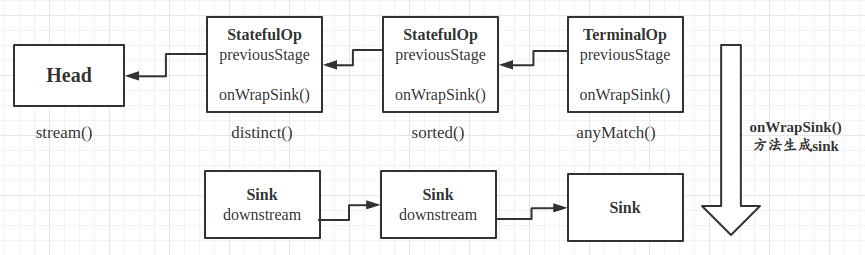
每个阶段都会产生一个新的stream对象，并通过重写AbstractPipeline的opWrapSink()方法，返回一个sink对象去记录当前对stream的操作。而不会去遍历list。

直到遇到结束(Terminal)操作。

例：

list.stream().distinct().sorted().anyMatch(x -> "a".equals(x));

TerminalOp未继承AbstractPipeline，未重写onWrapSink方法。

本例中的sink方法执行顺序：

distinct.begin() -> sorted.begin() -> distinct.accept() -> sorted.accept() -> distinct.end() -> sorted.end() -> anyMatch.begin() -> anyMatch.accept() -> anyMatch.end()

1. stream()方法生成Head-Stream对象。
2. distinct()生成StatefulOp-Stream对象，并重写opWrapSink()方法通过sink记录distinct需要对元素进行的操作，通过AbstractPipeline的previousStage成员记录Head对象，形成一个链。
3. sorted()生成StatefulOp-Stream对象，并重写opWrapSink()方法通过sink记录sorted需要对元素进行的操作，通过previousStage记录distinct生成的对象。
4. anyMatch()方法生成MatchOp -TerminalOp对象，并通过MatchSink- BooleanTerminalSink记录anyMatch需要的操作。
5. 通过previousStage循环调用opWrapSink()方法，直到获取到distinct()封装的sink，并在循环的过程中通过ChainedReference -Sink中的downstream成员变量将各阶段的sink串联起来。
6. 调用第一个sink的begin()方法。
7. 循环并调用第一个sink的accept()方法。
8. 调用第一个sink的end方法。
9. 后续的sink通过downstream串联起来，实现了递归的调用。直到结束(Terminal)操作才开始正式元素的遍历。