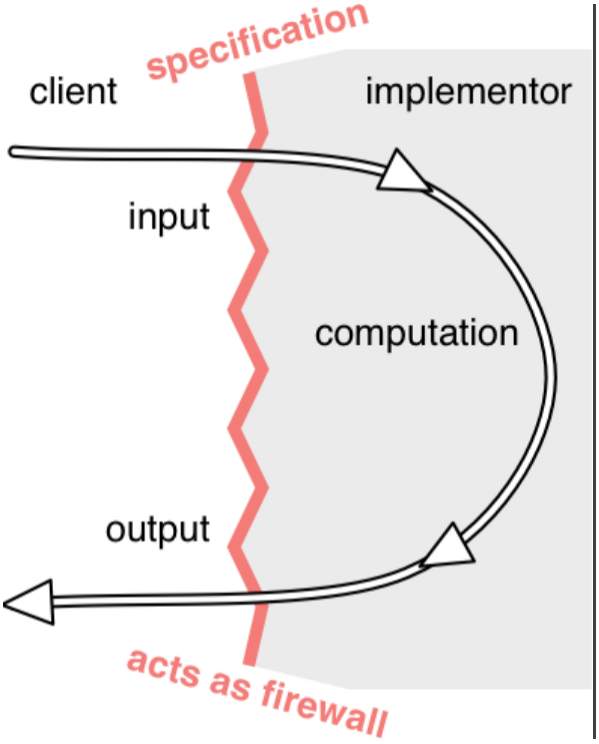


spec只讲做什么，不讲怎么做

行为等价性：两个函数符合同一个spec

前置条件：对客户端的约束，在使用方法时必须满足的条件

后置条件：对开发者的约束，方法结束时必须满足的条件

前置条件满足，则后置条件必须满足。

前置条件不满足，则方法可做任何事情。

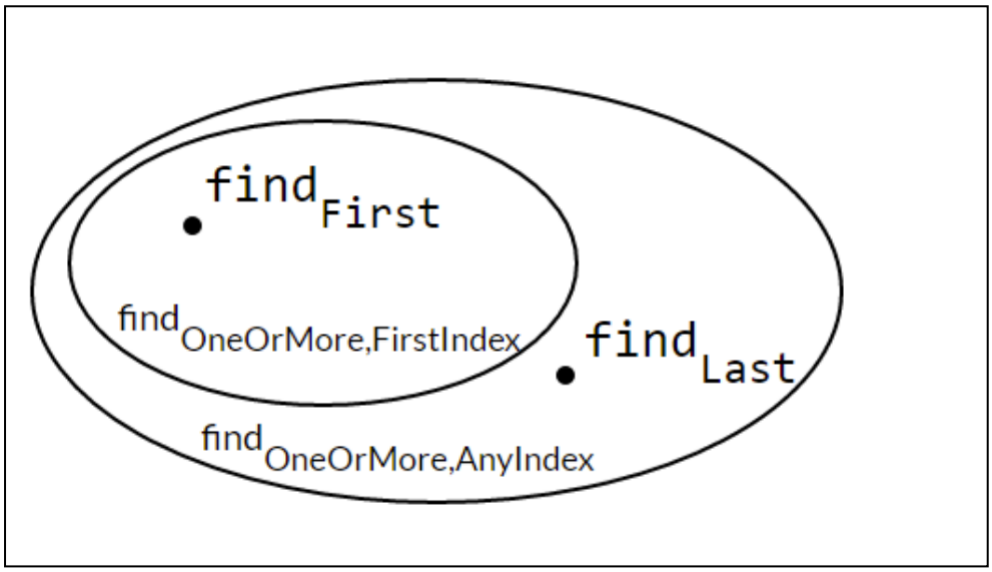
除非在后置条件里声明过，否则方法内部不应该改变输入参数

spec变强：更放松的前置条件+更严格的后置条件

确定的规约(Deterministic spec)：给定一个满足precondition的输入，其输出是唯一的、明确的

欠定的规约(Under-deterministic spec)：同一个输入可以有多个输出

更强的规约，表达为更小的区域



Spec描述的功能应单一、简单、易理解

在规约里使用抽象类型，可以给方法的实现体与客户端更大的自由度

在Java中参数类型尽量用接口类型，这样的话客户端的选择类型就广了

惯用做法是： 不限定太强的precondition，而是在postcondition中抛出异常：输入不合法

尽可能在错误的根源处fail，避免其大规模扩散

归纳：是否使用前置条件取决于

(1) check的代价；

(2) 方法的使用范围

1. 如果只在类的内部使用该方法(private)，那么可以不使用前置条件，在使用 该方法的各个位置进行check——责任交给内部client；
2. 如果在其他地方使用该方法(public)，那么必须要使用前置条件，若client端 不满足则方法抛出异常。