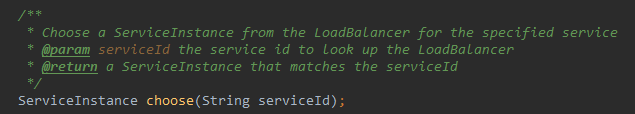
# 负载均衡策略

## 默认轮询

Ribbon默认的负载均衡策略是简单的轮询，我们可以测试一下：

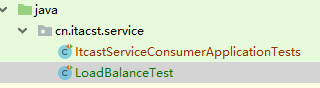
编写测试类，在刚才的源码中我们看到拦截中是使用RibbonLoadBalanceClient来进行负载均衡的，其中有一个choose方法，找到choose方法的接口方法，是这样介绍的：



## 测试类

现在这个就是负载均衡获取实例的方法。

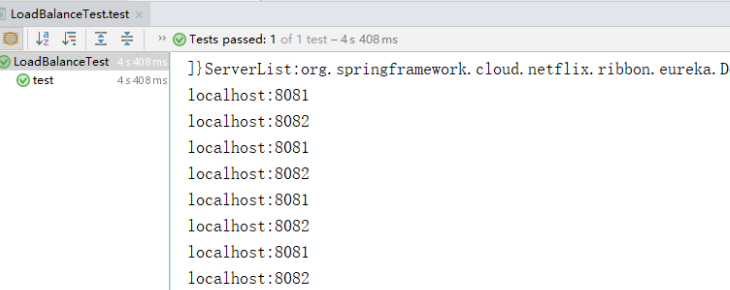
我们注入这个类的对象，然后对其测试：



### 测试内容：



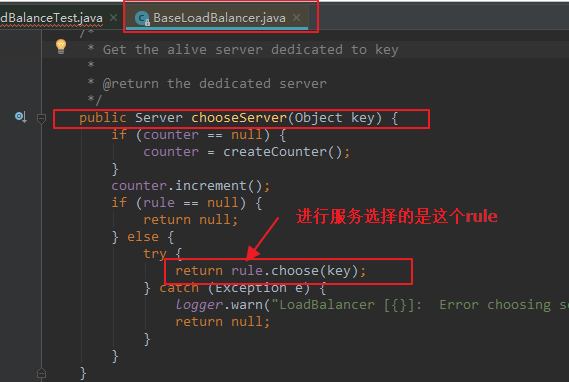
### 结果：



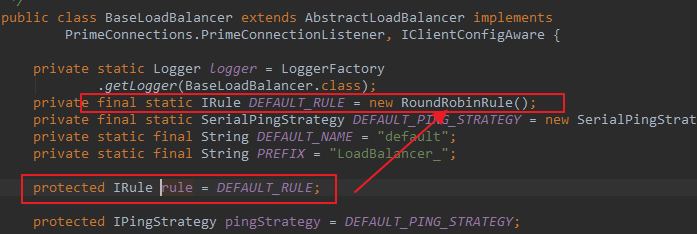
符合了我们的预期推测，确实是轮询方式。

## 我们是否可以修改负载均衡的策略呢？

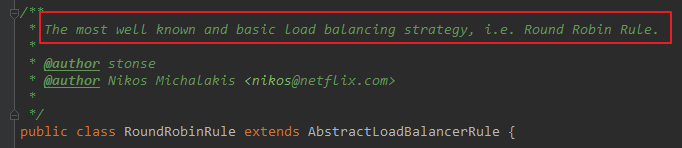
继续跟踪源码，发现这么一段代码：



我们看看这个rule是谁：

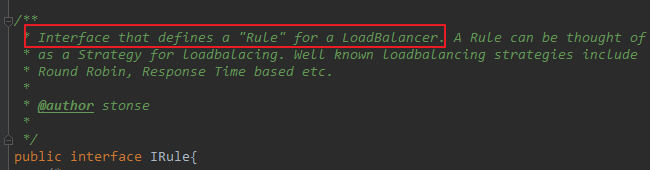


这里的rule默认值是一个RoundRobinRule，看类的介绍：



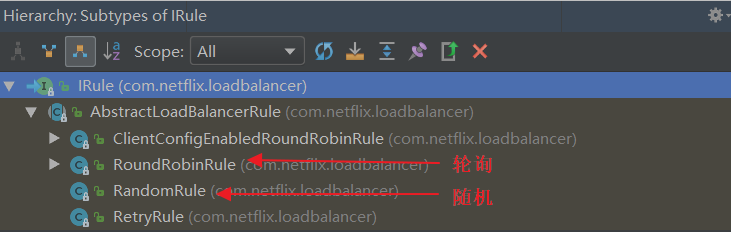
这不就是轮询的意思嘛。

我们注意到，这个类其实是实现了接口IRule的，查看一下：



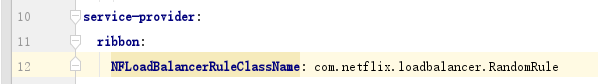
定义负载均衡的规则接口。

它有以下实现：



## Spring boot 配置

SpringBoot也帮我们提供了修改负载均衡规则的配置入口，在itcast-service-consumer的application.yml中添加如下配置：

 service-provider:

ribbon:

NFLoadBalancerRuleClassName: com.netflix.loadbalancer.RandomRule

格式是：{服务名称}.ribbon.NFLoadBalancerRuleClassName，值就是IRule的实现类。

再次测试，发现结果变成了随机：

