code review，代码审查，代码走读

到这里为止，你的team里的一个同学，就已经把他要干的活儿全部都干完了

此时，他就应该push自己本地的代码到gitlab上去，然后在gitlab上对他自己的feature分支，提交一个Pull Request到develop分支上去

Pull Request的reviewer就是你，架构师本人，也可以是团队内其他的高工，如果你作为一个架构师，带了十几个人，二十人的团队，你手下有三四个高工，每个高工带了一个小团队，那么其实一般就是让高工负责去审查他们的代码就可以了，你负责审查高工的代码

Pull Request里面，是可以写备注的 ，这个备注的规范如下：

（1）标题：用一行文字，简短的概括说明，你本次开发好了哪些东西

（2）列表

1.完成了订单系统的开发

（1）包括XX模块

（2）包括XX模块

2.完成了调度系统的开发

（1）包括XX模块

（2）包括XX模块

单元测试报告：可以通过邮件去发送，然后在这里可以备注好，通过哪份邮件，已经发送过去了，对方可以查看

冒烟测试报告：邮件

静态代码扫描报告：邮件

接收到这个Pull Request的同学，就需要对他的代码去进行审查了

code review，要看哪些东西，众说纷纭，可以写的很详细，也可以写的很粗糙

这一块，基本上要定一个特别细的规范，很难的，负责review的那个人，自己时间很紧张，可能一般都没有太多的时间去给你看代码

所以你务必要确保说，在短时间内，这个负责审查代码的人，可以确保说，这个代码是ok的

我根据经验梳理出来一些，看别人的代码，重点要看什么：

（1）看懂别人的代码：基本上都按照编码规范，命名都很清晰，面向对象的划分都很清晰，注释很清晰，逻辑交互很清晰。如果说你觉得能看懂这个人的代码，而且代码设计的很优雅，未来的可扩展性很高，那么就可以。但是如果你发现有些地方的命名，或者是代码注释，不够清晰，导致你不好理解，看不懂，而已给建议。这块是在把控代码的质量。第一点，就是让人能看懂。 => 代码质量

（2）仔细审查一下业务逻辑是否准确，对照着详细设计文档来看，你作为一个小leader，你要负责确保说，你手下写出来的代码，是按照详细设计文档里的设计在写的，整个业务流程和业务逻辑都是准确的。看一下这个逻辑，是否有可能会在某些特殊情况下出现bug，也是根据你的经验和能力来的，你的经验和能力越强，这一步能看出来的问题就越多。 => bug

（3）看一看代码编写的细节有没有什么问题：这个，就根据你的经验来判断，比如说，你发现某一段代码，在for循环内部出现了每次循环都创建一个对象的，备注一下，建议将循环内创建对象的逻辑抽到循环外面来执行。我可以给出来很多很多的建议和细则，大家很难去执行的，还有很多，根据你自己的技术能力和技术经验来了，你的技术越强，经验越丰富，往往在这个环节，你可以提出很多的问题。io读取文件，你会建议说，是否可以重构为nio的方式。 => 对代码设计和技术实现的质量

（4）单元测试，初步浏览一遍，看看单元测试代码写的是否ok，如果出现一些不太规范不太好的地方，给一些建议，让单元测试代码重构一下。看一下单元测试覆盖率的报告，确保说，70%的覆盖率，核心包下的代码覆盖率是100%。

（5）冒烟测试，仔细看一遍，确保说，他设计的那些冒烟测试用例都执行了，这个就代表着说他做了这个事情

（6）静态代码扫描的报告，这个就是确保说，都是符合编码规范的，代码是规范的

有了规范，必须确保可以落地，可以执行，唯一能够依靠的就是check

在code review这个环节，就可以去check之前的所有流程，单元测试覆盖率是否达标，冒烟测试是否全部通过，静态代码扫描是否全部通过