《软件工程》

期末考试答卷

张腾 201700301030 菁英班

SDUOJ小组 后端架构与编码

授课老师：梁文革

目录

[理论应用题1. 2](#_Toc44404126)

[理论应用题2. 4](#_Toc44404127)

[理论应用题3. 6](#_Toc44404128)

[分析计算题1. 7](#_Toc44404129)

[分析计算题2. 9](#_Toc44404130)

[综合设计题1. 10](#_Toc44404131)

[综合设计题2. 13](#_Toc44404132)

[综合设计题3. 16](#_Toc44404133)

# 理论应用题1.

请画出传统软工生命周期模型与现代软工开发（如敏捷开发）模型，论述两者的特点和不同。论述自己的实践项目开发属于哪种生命周期模型？

答：

传统软工生命周期模型，即瀑布模型，模型图如下：



瀑布模型的优点：

（1）它的简单性使得开发人员很容易向不熟悉软件开发的客户作出解释。

（2）每一个过程活动都有与其相关联的里程碑和可交付产品，以便于项目经理评估项目进度。

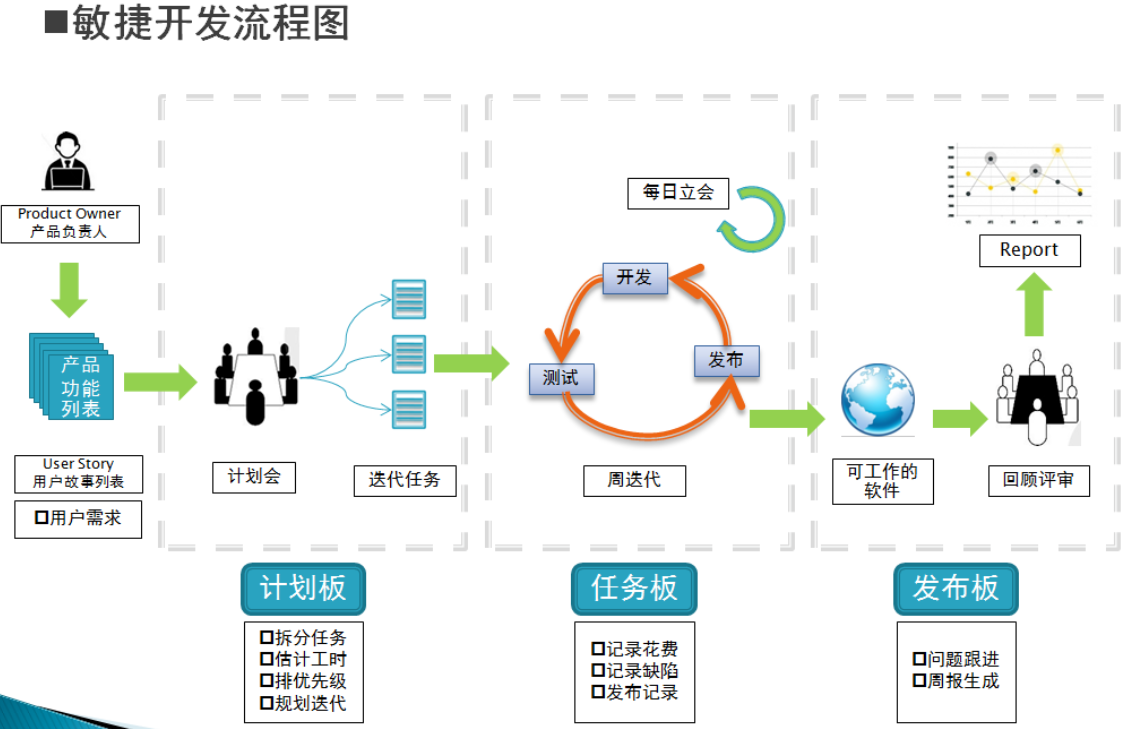
（3）瀑布模型是最基础的模型，很多其他更复杂的模型实际上是在瀑布模型的基础上的润色，如加入反馈循环以及额外的活动。

瀑布模型的缺点：

（1）除了一些理解非常充分的问题之外，实际上软件是通过大量的迭代进行开发的。软件是一个创造的过程, 不是一个制造的过程。软件变动时, 该模型无法处理实际过程中的重复开发问题。

（2）文档转换有困难。它说明了每一个活动的产品（例如，需求、设计或代码），但没有揭示一个活动如何把一种制品转化为另外一种制品（例如，从需求文档转化为设计文档）。

现代软工开发模型，取敏捷开发模型，敏捷开发的流程图如下：



敏捷方法的特点（原则）：

（1）个体和交互的价值胜过过程和工具。

（2）可以工作的软件胜过面面俱到的文档。

（3）客户合作胜过合同谈判。

（4）响应变化胜过遵循计划。

传统软工生命周期模型（瀑布模型）与现代软工开发模型（敏捷方法）的不同：

（1）传统软工生命周期模型十分庞大，而敏捷方法打破了这种局面。

（2）敏捷方法强调灵活性在快速有效的软件生产中所发挥的作用，而瀑布模型强调线性的安排每一个阶段，将开发阶段描述为从一个阶段瀑布般地转换到另一个阶段。一个开发阶段必须在另一个开发阶段开始之前完成。

（3）敏捷方法强调尽可能早的，持续的对有价值的软件的交付活动，以客户满意。瀑布模型强调线性的安排每一个阶段，保证最终交付。

SDUOJ小组的实践项目开发属于敏捷方法。

我们的项目实践SDUOJ小组，属于每个人员工程经验较少，但是程序算法编码基础强、知识学习能力强的小组，所以我们划分各自负责的模块后，自主学习，每周定时组会汇报学习进度和编码进度（体现了敏捷开发中强调个体和交互的价值）。尽管这样，但是我们仍不能在项目开始之初就定好整体的功能框架，后续逐渐增强各自功能，同时我们也不能保证每次增量开发增加的都是规模小、功能鲁棒的子系统，所以我们组不能采用瀑布模型，我们采用的是敏捷开发。好处是能及时开拓新功能的开发和上线，并且对于发布的部分能及时修复使用过程中发现的Bug。

举例：我负责SDUOJ项目的后端，后端采用面向服务（SOA）的架构，在我开发SDUOJ的题库微服务时，同时设计着比赛微服务模块的架构，此时评测端的人员开发完评测功能，转到管理端开发，我必须要在相应的题库微服务中开发并透出管理员权限的接口，由于题库微服务的框架已搭好，增加新的API接口这属于迭代开发；我在开发题库微服务的同时在设计着比赛微服务模块的架构，这属于增量开发。产品每周例会，汇报和不断迭代。

# 理论应用题2.

请论述德米特法则，结合自己的项目举例说明如何应用？

答：

德米特法则又叫作最少知识原则，一个类对于其他类知道的越少越好，就是说一个对象应当对其他对象有尽可能少的了解，只和朋友通信，不和陌生人说话。

德米特法则认为，任何一个对象或者方法，它应该只能调用下列对象：

（1）当前对象本身（this）

（2）以参量形式传入到当前对象方法中的对象

（3）当前对象的实例变量直接引用的对象

（4）当前对象的实例变量如果是一个聚集，那么聚集中的元素也都是朋友

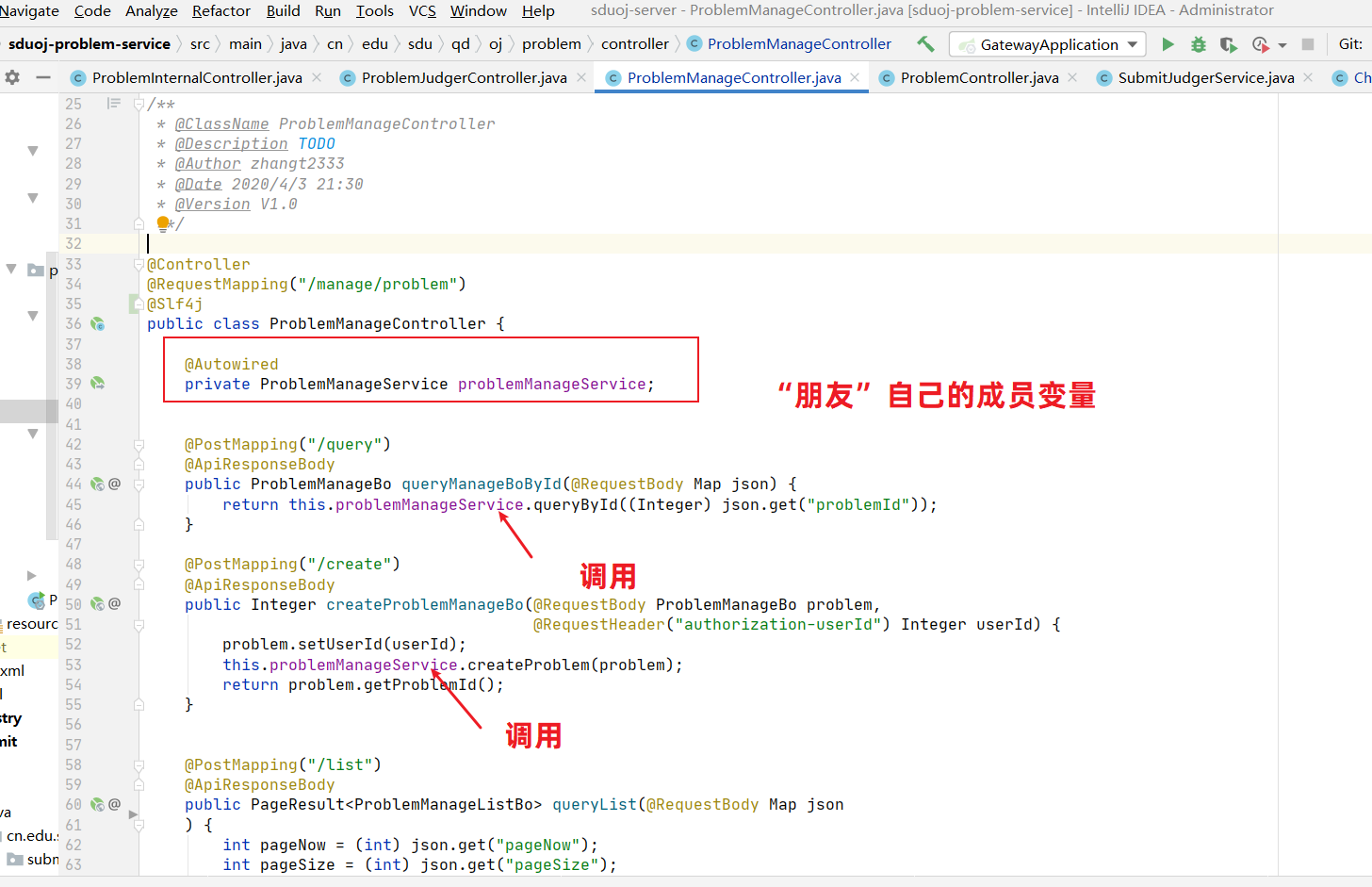
（5）当前对象所创建的对象

德米特法则的核心观念就是类间解耦，弱耦合，只有弱耦合了以后，类的复用率才可以提高，其要求的结果就是产生了大量的中转或跳转类，类只能和朋友交流，朋友少了你业务跑不起来，朋友多了，你项目管理就复杂。

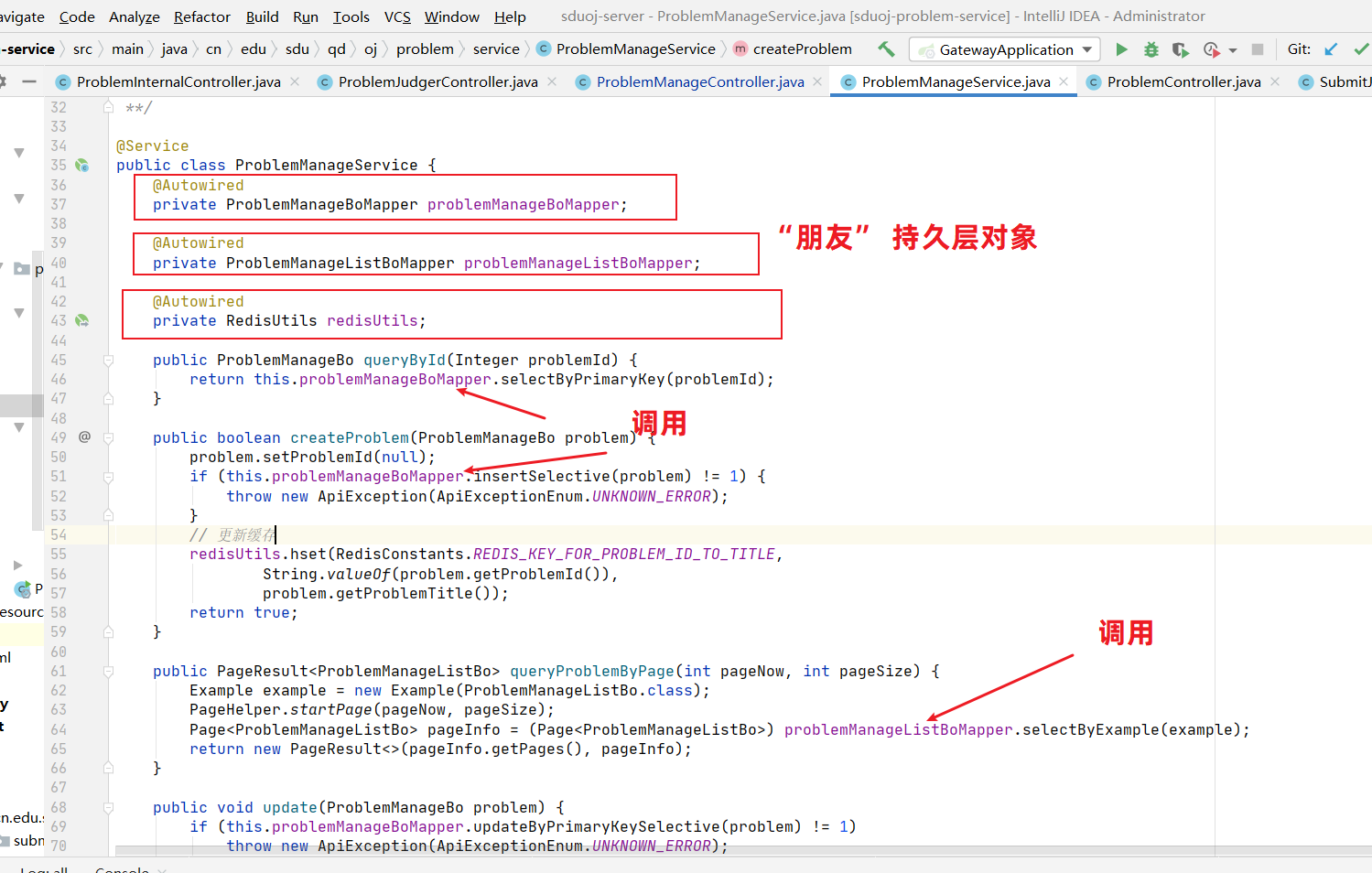
或者具体到编码层面上来说，德米特法则就要求类“小气”一点，尽量不要对外公布太多的public方法和非静态的public变量， 尽量内敛， 多使用private、package-private、protected等访问权限。

我在SDUOJ小组中负责开发后端和设计系统架构，所以下面将从这方面论述。

在我们的项目的后端设计中，请求链路是采用最典型的MVC模型进行处理，一个请求抵达微服务后，被Controller层接收，可以看到Controller持有的Spring依赖注入的Service层对象是被private关键字修饰的，并且根据德米特法则，在Controller层，仅调用它的朋友Service对象，对其他对象尽可能少的了解。



在调用了Service层对象后，根据德米特法则，在Service层实现，也仅调用它的朋友持久层Mapper对象，去操作数据库，此时对其他对象尽可能少的了解。



# 理论应用题3.

请写出几种软件开发团队的组织结构，论述各自的特点。结合自己的小组论述理想的团队组织是怎样的？

答：

软件开发团队的组织结构有Chief Programmer Team、Egoless Approach等。

Chief Programmer Team即主程序员负责制，由一个主程序员负责系统设计和开发，其他的成员向其汇报，主程序员对每一个决定有绝对决策权。特点是：可以使交流最小化、迅速做出决定，缺点是创造性低、对主程序员要求高，个人主观性强。

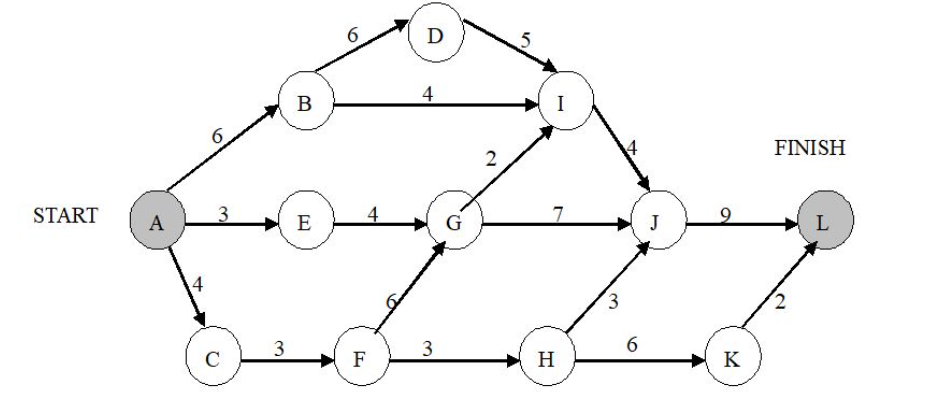
Egoless Approach即忘我方法，每个成员平等的承担责任，而且过程与个人是分开的；批评是针对产品和结果的，不针对个人的。

二者应用于计划和管理项目中的项目团队组织领域。

我们的项目实践SDUOJ小组，共5人，5个人全部属于程序算法编码基础强，但是大多工程经验较弱的成员类型。我本人作为后端开发，难免要协调前端和评测端，所以也负责系统设计和开发，但SDUOJ团队谈不上是Chief Programmer Team，我也谈不上是主程序员。我们5人是使用Egoless Approach的团队，这也是我们理想的团队组织，每个人负责一部分，比如前端、后端、评测端，每人平等地承担责任，自主学习，每周定时组会汇报学习进度和编码进度，遇到决策困难和设计困难会在组会上讨论，调研，最终投票做出决定，由具体人员负责开发。属于交流充分，减少开发周期反复联调的项目实践方法。

# 分析计算题1.

下图是一个软件开发项目的活动图，边长代表天数。请分析标出每一个活动的最早开始时间、最晚开始时间和时差。然后找出关键路径和其总长度。如果图中有两条关键路径，这两条关键路径的长度是否相等？请修改图中某一活动的时间，使得图中再出现一条关键路径，并标出。

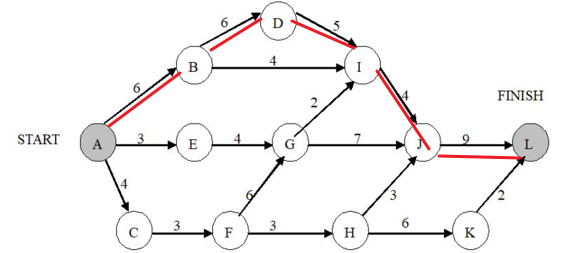


答：

未修改活动时间的结果：

| **活动** | **最早开始时间** | **最晚开始时间** | **时差（冗余时间）** |
| --- | --- | --- | --- |
| AB | 1 | 1 | 0 |
| AE | 1 | 8 | 7 |
| AC | 1 | 2 | 1 |
| BD | 7 | 7 | 0 |
| BI | 7 | 14 | 7 |
| EG | 4 | 11 | 7 |
| CF | 5 | 6 | 1 |
| DI | 13 | 13 | 0 |
| FG | 8 | 9 | 1 |
| GI | 14 | 16 | 2 |
| GJ | 14 | 15 | 1 |
| IJ | 18 | 18 | 0 |
| FH | 8 | 16 | 8 |
| JL | 22 | 22 | 0 |
| HJ | 11 | 19 | 8 |
| HK | 11 | 23 | 12 |
| KL | 17 | 29 | 12 |

由上述表格可知，AB、BD、DI、IJ、JL 活动的时差为 0，即为关键节点，因此关键路径为 A->B->D->I->J->L= 30。

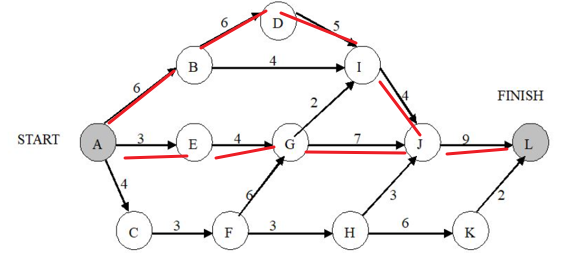


如果图中有两条关键路径，这两条关键路径的长度一定相等。

修改AE这条边，改为10，此时结果如下：

| **活动** | **最早开始时间** | **最晚开始时间** | **时差（冗余时间）** |
| --- | --- | --- | --- |
| AB | 1 | 1 | 0 |
| AE | 1 | 1 | 0 |
| AC | 1 | 2 | 1 |
| BD | 7 | 7 | 0 |
| BI | 7 | 14 | 7 |
| EG | 11 | 11 | 0 |
| CF | 5 | 6 | 1 |
| DI | 13 | 13 | 0 |
| FG | 8 | 9 | 1 |
| GI | 15 | 16 | 1 |
| GJ | 15 | 15 | 0 |
| IJ | 18 | 18 | 0 |
| FH | 8 | 16 | 8 |
| JL | 22 | 22 | 0 |
| HJ | 11 | 19 | 8 |
| HK | 11 | 23 | 12 |
| KL | 17 | 29 | 12 |

由上述表格中活动时差为0的关键节点，得到两条关键路径：A->B->D->I->J->L=30、A->E->G->J->L=30。

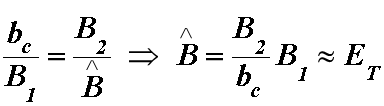


# 分析计算题2.

假定有两个小组测试同一个程序。小组1发现了26个故障，小组2发现了30个故障，其中，小组2发现的故障中有18个与小组1发现的故障相同。请利用Fault seeding原理，估算程序中的故障总数。

答：

在软件开发的过程中，利用测试的统计数据，估算软件的可靠性，以控制软件的质量是至关重要的。

采用Fault Seeding原理，设以小组1的测试结果为基准 （即相当于1中的植入故障）设，由题干得知，其中B1=26，B2=30，bc=18：

估算程序中的故障总数为：26\*30/18=43。

# 综合设计题1.

结合自己的软工项目实践,举例从体系结构、设计原则等方面进行的提高软件质量所做的重构工作。

对比修改前后两种设计方案的特点不同。

答：

我在SDUOJ小组中负责开发后端和设计系统架构，所以下面将从这方面论述。

（1）体系结构方面所做的重构工作：从单机单模块工程，到使用面向服务（SOA）的架构，转为多机多模块的工程

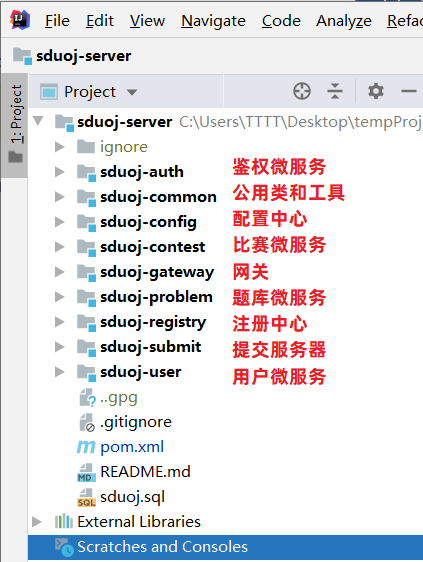
背景：

在开始架构SDUOJ时，后端使用Java的Spring Boot的单机单模块形式，部署上只能使用一台服务器部署，没有任何的容错和高并发高可用，并且在编码方面的修改容易”牵一发而动全身”，具有许多缺点，并不适合生产。

重构：

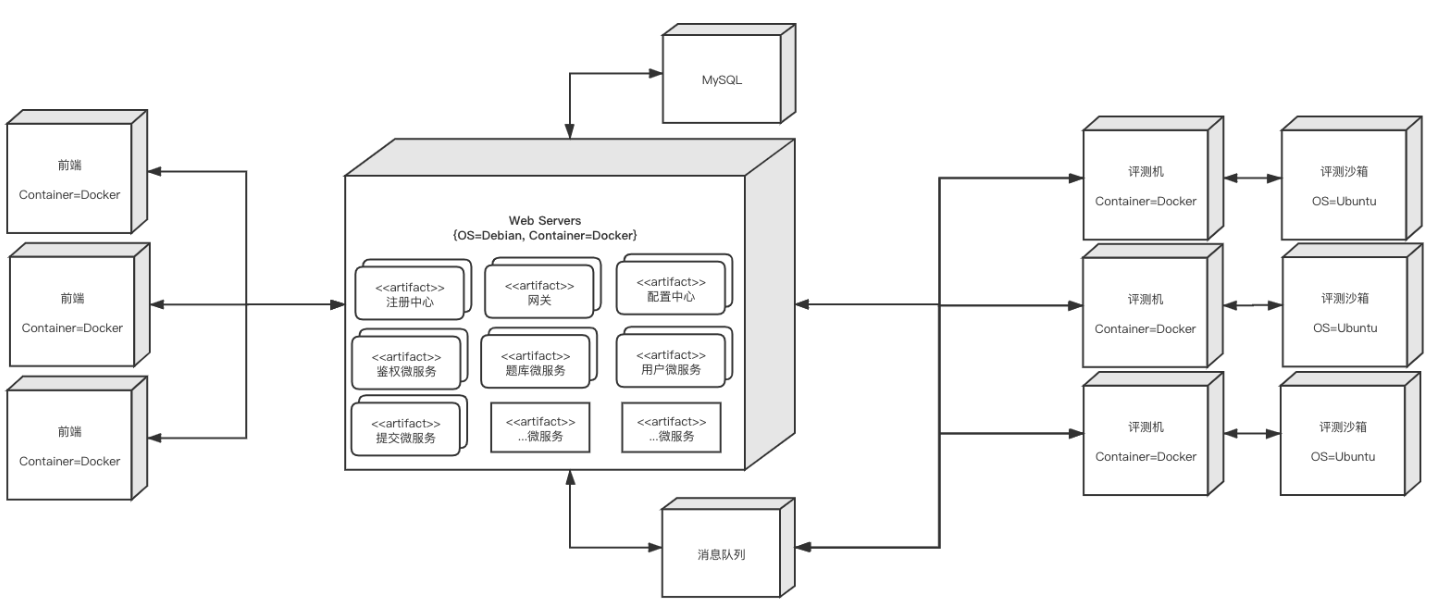
使用Spring Cloud提供的微服务组件，其中 Eureka 是服务注册中心，所有的微服务都要申请到它那里，进行服务注册；APIGateWay是网关，所有的请求都要经过它，它会将请求转发到具体的微服务；ConfigServe是配置中心，通过它可以轻松区分开发和测试环境，更新配置只需要推送到Git仓库，即可立即刷新，而无需重启服务器。

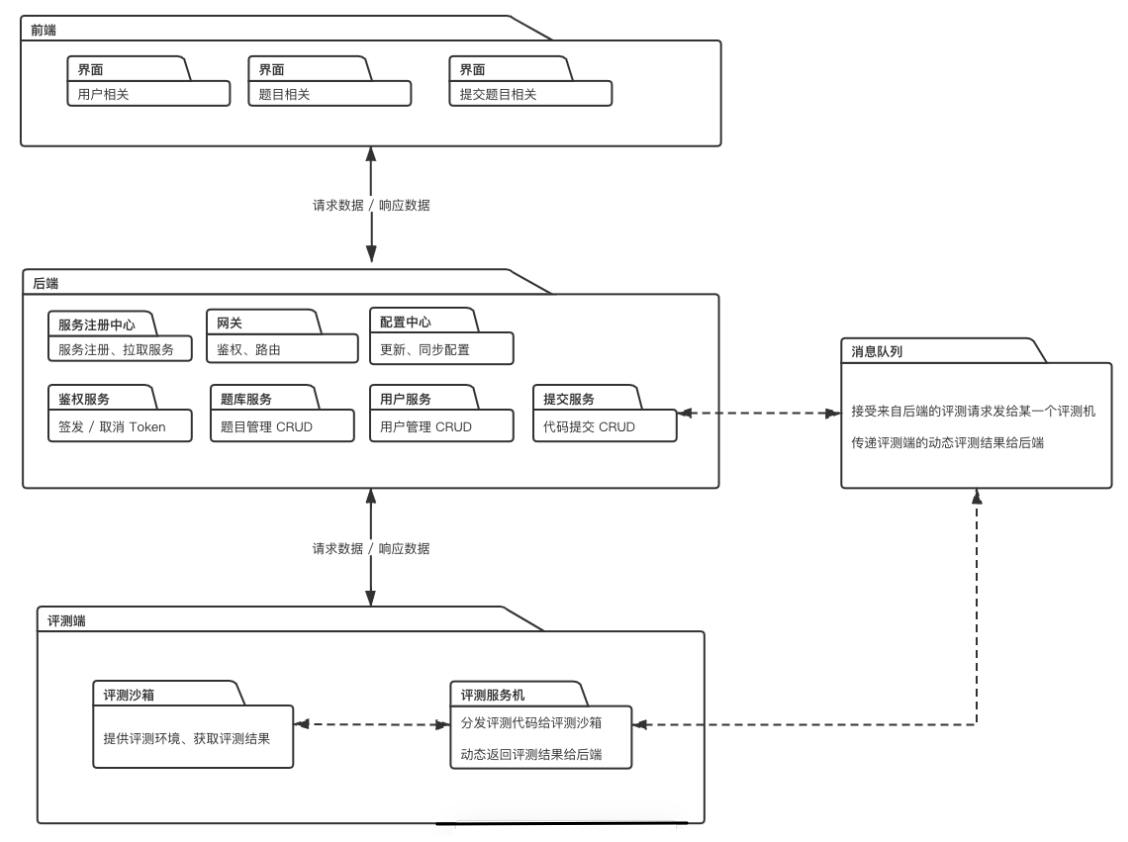
重构后的项目工程目录变成如下：



改变：

重构之后，各个微服务可以独立部署在多台服务器上，结合Docker部署和阿里云的脚本引擎，可以实现弹性动态扩容部署。在性能、可靠性、健壮性、易实用性上有极大的提高。





重构之后，模块与模块之间的编码独立，相互解耦。需要使用到对方的接口，则引入对方模块发布的二方包接口，使用远程RPC在系统内部进行调用。对于具体的实现细节是透明的。单个模块实现更好的内聚，在逻辑上都是与自己相关联的代码文件存放一起，在通讯上也实现了微服务内部解决绝大部分数据请求。

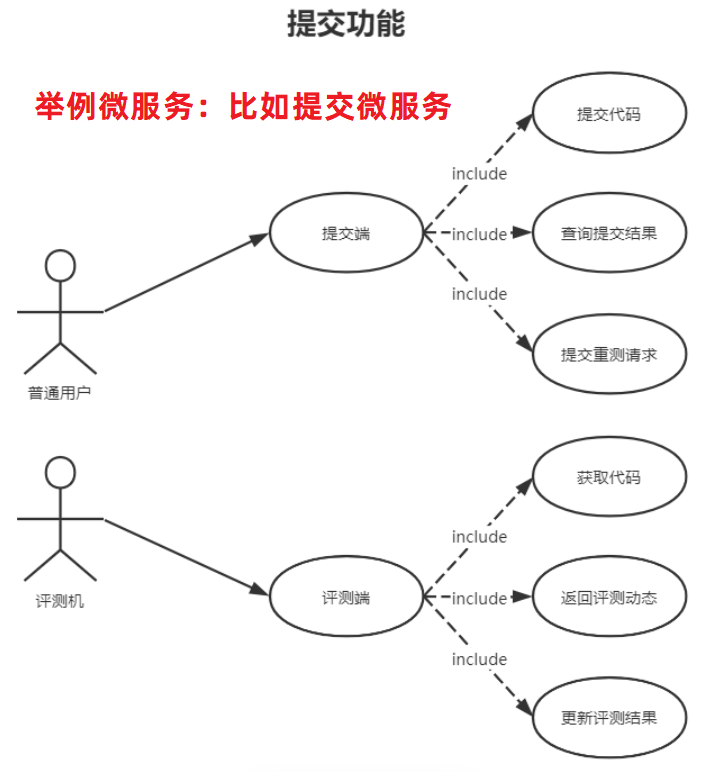
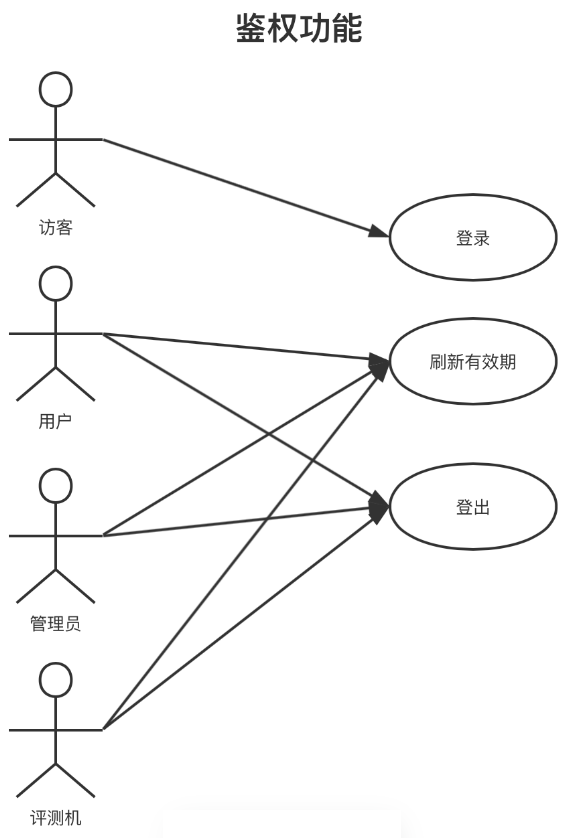
论证：

SDUOJ经历过一次核心组件更换，网关组件的更换，在题库实现”动态数据评测”时，我们引入了WebSocket技术，一开始选型的网关组件为Netflix开源的Zuul 1.x版本网关，这代网关使用的是阻塞的IO模型，所以不支持WebSocket，我们更换成了支持异步非阻塞的Spring Cloud官方研发的APIGateway组件，此时除了网关，其他端以及微服务的代码基本保持不变！

（2）使用面向对象设计原则（OO设计原则）所做的重构工作：使用装饰器模式装饰网关转发流，在网关层解决权限鉴别问题，简化代码

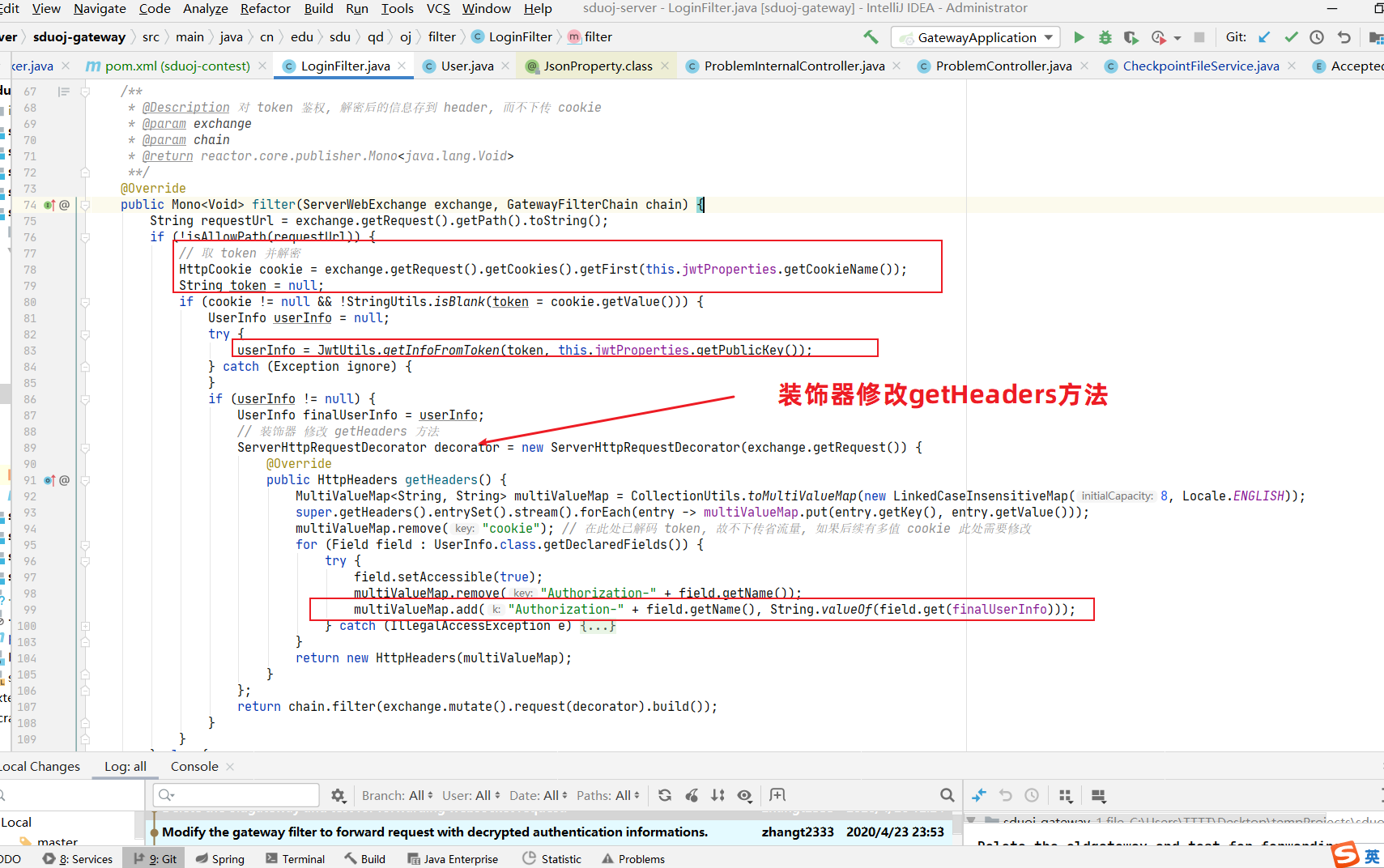
背景：

在SDUOJ中，鉴权模块持有秘钥负责校验用户账号密码并写出JWT鉴权信息到Cookie，而每个独立的微服务如果需要对调用接口的用户进行信息读取，需要持有公钥并对Cookie进行解密计算，这个过程中要消耗CPU，并且，为每个微服务透出公钥，有一定的安全风险。



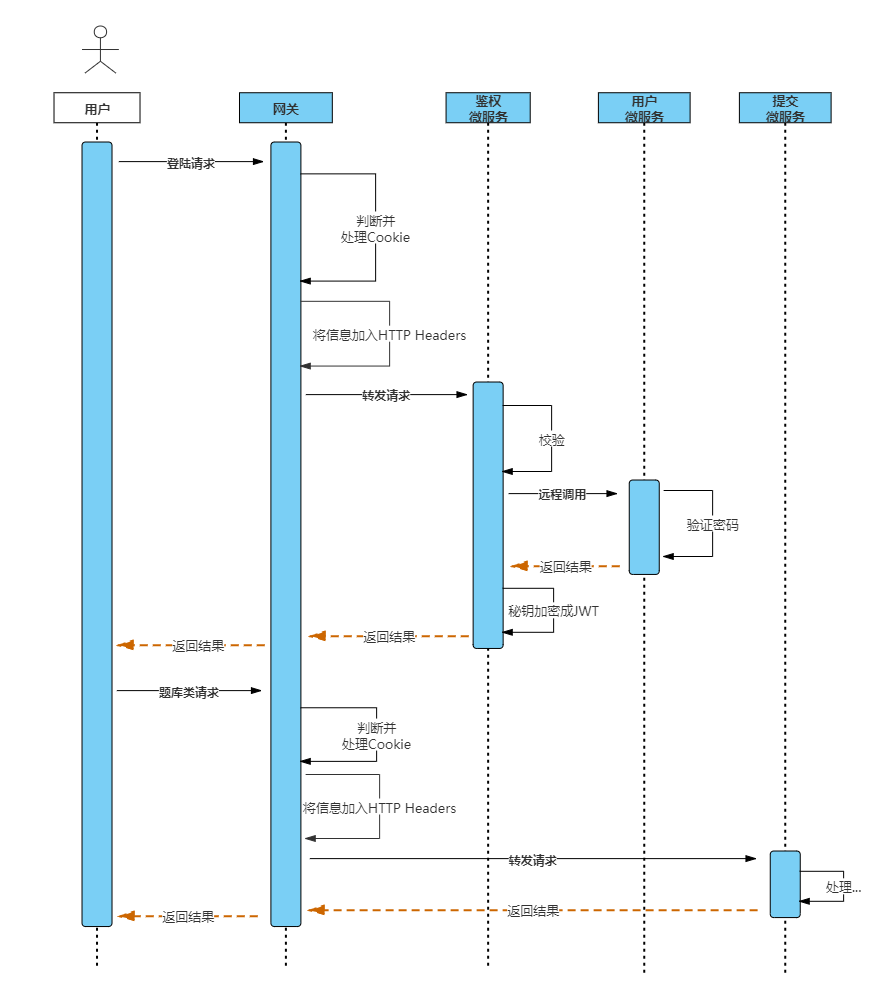
重构：

在更换了网关组件后，APIGateway倡导Spring 5新提出的响应式编程风格，其中提供了一种装饰器加入，可以代理网关的转发方法。只要网关持有公钥对Cookie中的JWT进行鉴定，并且把信息重新附加在HTTP Header中，接收到网关转发请求的微服务就不需要自己持有公钥并消耗CPU进行解密，而是将这部分集中在网关层完成。（代码改动如下）



改变：

只需要网关持有公钥，提高了一定的安全性。删除了各微服务处需要单独维护解密JWT代码的部分，相当于从各地抽取（Extract）出公用代码，提高了重用性和可维护性，比如将来要修改解密部分，增加信息或其他需求，可以改动一处，多处同步。



# 综合设计题2.

结合自己的软工实践，写出项目中应用的两种设计模式及其特点。

答：

我们项目中使用的设计模式有：

* 工厂模式
* 抽象工厂模式
* 单例模式
* 建造者模式
* 原型模式
* 策略模式
* 责任链模式
* 装饰器模式

我在SDUOJ小组中负责开发后端和设计系统架构，所以下面将从这方面论述。

我将列举装饰器模式和责任链模式，及其实践举例。

首先装饰器模式为允许向一个现有的对象添加新的功能，同时又不改变其结构。属于结构型模式，它是作为现有的类的一个包装。优点是装饰类和被装饰类可以独立发展，不会相互耦合，装饰模式是继承的一个替代模式，装饰模式可以动态扩展一个实现类的功能。缺点是多层装饰比较复杂。

其次是责任链模式，责任链模式为请求创建了一个接收者对象的链。这种模式给予请求的类型，对请求的发送者和接收者进行解耦。这种类型的设计模式属于行为型模式。在这种模式中，通常每个接收者都包含对另一个接收者的引用。如果一个对象不能处理该请求，那么它会把相同的请求传给下一个接收者，依此类推。优点是降低耦合度。它将请求的发送者和接收者解耦、简化了对象。使得对象不需要知道链的结构、增强给对象指派职责的灵活性。通过改变链内的成员或者调动它们的次序，允许动态地新增或者删除责任、增加新的请求处理类很方便。缺点是不能保证请求一定被接收、系统性能将受到一定影响，而且在进行代码调试时不太方便，可能会造成循环调用、可能不容易观察运行时的特征，有碍于除错。

实践举例：

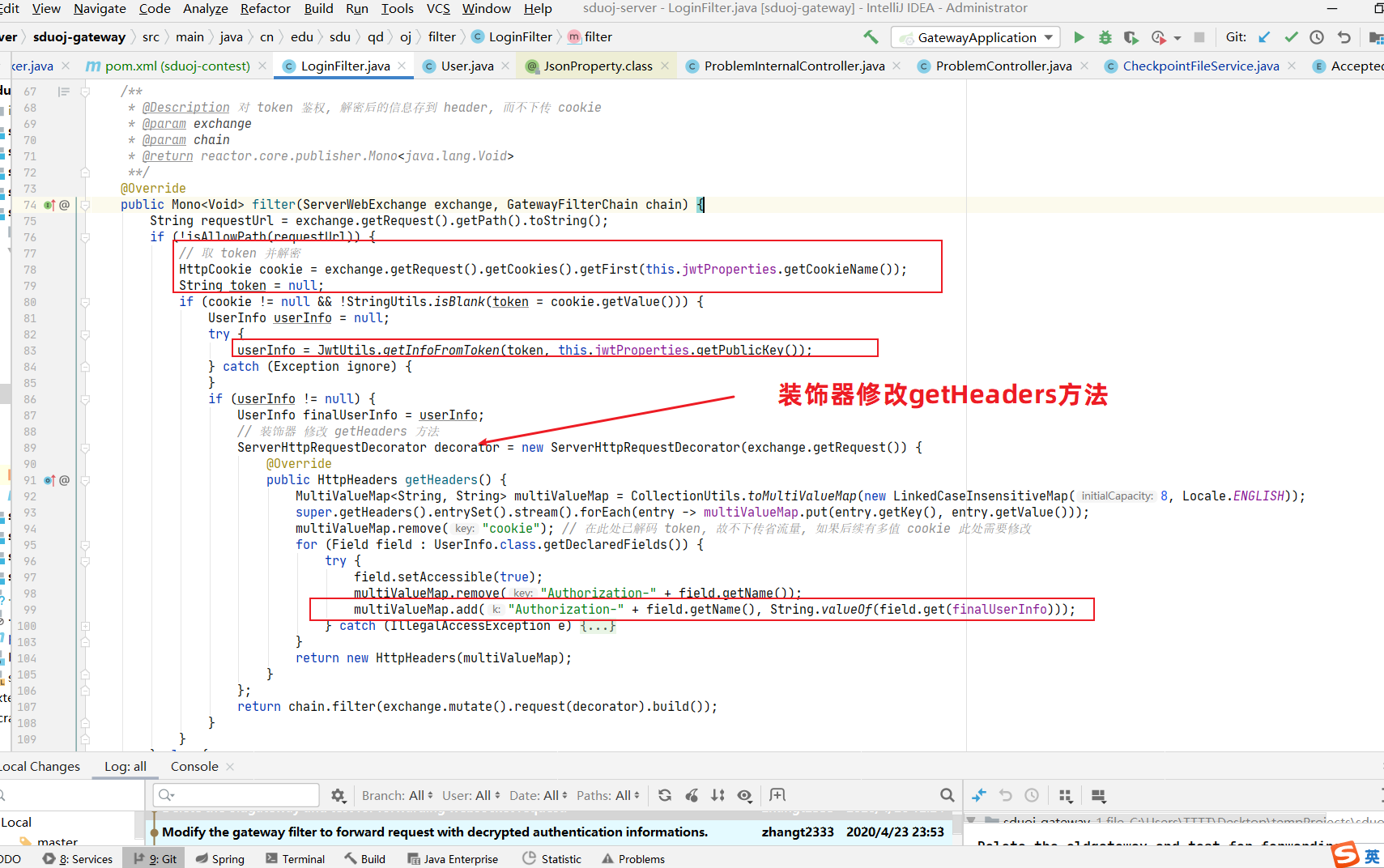
① 使用装饰器模式装饰网关转发流，在网关层解决权限鉴别问题，简化代码

背景：

在SDUOJ中，鉴权模块持有秘钥负责校验用户账号密码并写出JWT鉴权信息到Cookie，而每个独立的微服务如果需要对调用接口的用户进行信息读取，需要持有公钥并对Cookie进行解密计算，这个过程中要消耗CPU，并且，为每个微服务透出公钥，有一定的安全风险。

重构：

在更换了网关组件后，APIGateway倡导Spring 5新提出的响应式编程风格，其中提供了一种装饰器加入，可以代理网关的转发方法。只要网关持有公钥对Cookie中的JWT进行鉴定，并且把信息重新附加在HTTP Header中，接收到网关转发请求的微服务就不需要自己持有公钥并消耗CPU进行解密，而是将这部分集中在网关层完成。（代码改动如下）



改变：

只需要网关持有公钥，提高了一定的安全性。删除了各微服务处需要单独维护解密JWT代码的部分，相当于从各地抽取（Extract）出公用代码，提高了重用性和可维护性，比如将来要修改解密部分，增加信息或其他需求，可以改动一处，多处同步。

② 使用责任链模式在鉴权微服务使用用户微服务提供的功能，实现解耦

背景：

在SDUOJ中，鉴权模块持有秘钥负责校验用户账号密码并写出JWT鉴权信息到Cookie。其中校验用户账号密码，是该鉴权控制器对象无法解决的问题。

实现：

通过引入用户模块发布的二方包接口，调用远程RPC在系统内部进行调用。传递账号密码到用户微服务，



改变：

这体现了SDUOJ面向服务（SOA）的架构思想，各个微服务可以独立部署在多台服务器上，结合Docker部署和阿里云的脚本引擎，可以实现弹性动态扩容部署。在性能、可靠性、健壮性、易实用性上有极大的提高。单个模块实现更好的内聚，在逻辑上都是与自己相关联的代码文件存放一起，在通讯上也实现了微服务内部解决绝大部分数据请求。

# 综合设计题3.

本学期每次实验任务的最后一条要求属于软工哪个领域？结合自己的软工，写出其实践情况、意义、难点？

结合项目实践写出软件文档的作用。

答：

本学期每次实验任务的最后一条为：“记录项目及小组的最新进度及工作量。记录项目及小组每个人最新的工作的进度、里程碑、工作量的跟踪图或表。每周更新。每人向组长汇报自己工作。组长汇总进度里程牌，提交小组共同报告。”，属于软件工程的软工计划和管理项目领域以及编写程序中文档化领域。

实践情况：

SDUOJ使用飞书进行文档管理，组内五人每周六下午都会召开组会，对项目进度以及未来的流程进行讨论和总结，并汇总成当周的工作日志。同时成员之间互相交流，讨论并解决项目开发过程中所产生的疑难问题，记录本周项目完成了哪些，下周需要完成哪些功能。



意义：

 项目进度是对特定项目的软件开发周期的刻画。包括对项目阶段、步骤、活动的分解，对各个离散活动的交互关系的描述，以及对各个活动完成时间及整个项目完成时间的初步估算。 如果要开始一个项目，我们需要跟客户讲述一下我们要做哪些工作来实现这个项目，做多长时间，用多少花费。回答不上这些问题估计客户是不敢找我们干的，回答这些问题就需要我们有明确的实现项目的各个步骤活动的具体时间计划、干的时候十分清楚自己干到了哪一步、预估整体的预算是多少，凭什么值这个价。做到以上几点，客户才能安心的将项目托付给我们。

难点：

(1) 小组里的5个人的时间规划不一，比较难找到交集的片段空闲时间，所以要努力缩短讨论会议时间，在尽可能短的时间内，完成全部的规划和总结。

(2) 工作进度、工作量不好体现，每个人的工作都不相同，难以指定一个相同的量纲。

(3) 大家都不是熟练的软件开发者，都需要一定时间的磨合，并且也需要一定的时间才能上手写代码，很难快速的推进工作进度。

软件文档的作用：

（1）用来记录、描述、展示实施过程中一系列信息的处理过程，通过书面或图示的形式对项目活动过程或结果进行描述、定义、规定及报告

（2）项目文档有助于项目管理水平的提高，在SDUOJ项目文档中，体现了SDUOJ开发进度，实时调整每周目标。

（3）项目文档是项目成果的体现形式，在SDUOJ项目文档中，体现了SDUOJ的架构和开发进度。

（4）提高项目实施过程的能见度。

（5）提高项目的实施效率。

（6）便于项目成员之间的交流与合作，在SDUOJ的对齐文档中，体现了SDUOJ的团队成员的交流和合作。

（7）操作指南文档可帮助最终用户规范化操作，强化培训效果，在SDUOJ外部文档文档中，对用户有培训作用。

（8）有利于项目实施的监控作用