一、理论应用题

写出下列软工的理论和其针对的领域应用。

1. Chief programmer team, egoless approach 应用于什么领域，各有什么特点？结合自己的项目实践，谈谈你们是如何做的？（10分）

答：

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6人，6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6人是使用Egoless Approach的团队，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。

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。特单是：可以是交流最小化、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汇总的项目团队组织领域。

2. Incremental development, iterative development 应用于什么领域，各有什么特点？结合自己的项目，举例说明你们的实践？（10分）

答：

Incremental development即增量开发，指系统需求按照功能分成若干子系统，开始建造的版本是规模小的、部分功能的系统，后续版本添加包含新功能的子系统，最后版本是包含全部功能的子系统集。

Iterative development即迭代开发，指系统开始就提供了整体功能框架，后续版本陆续增强各个子系统，最后版本使各个子系统的功能达到最强。

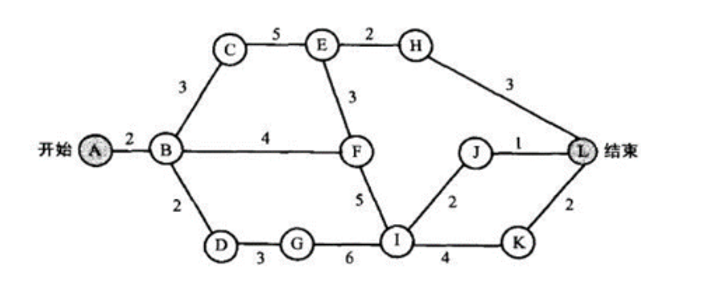
二者应用于软件开发中的阶段化开发模型领域。

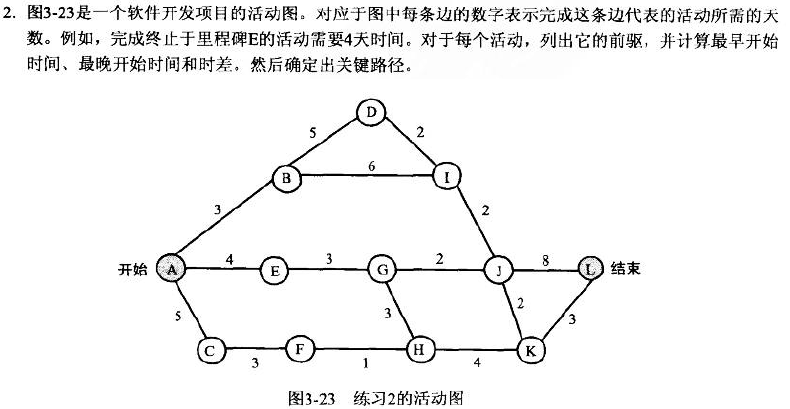
我们的项目实践SDUOJ小组，属于每个人员工程经验较少，但是程序算法编码基础强、知识学习能力强的小组，所以我们划分各自负责的模块后，自主学习，每周定时组会汇报学习进度和编码进度。尽管这样，但是我们仍不能在项目开始之初就定好整体的功能框架，后续逐渐增强各自功能，同时我们也不能保证每次增量开发增加的都是规模小、功能鲁棒的子系统，所以我们组采用的是增量开发和迭代开发相结合的开发模型。好处是能及时开拓新功能的开发和上线（增量部分），并且对于发布的部分能及时修复使用过程中发现的Bug（迭代部分）。

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。

二、分析题

1. 下图是一个软件开发项目的活动图，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。（10分）

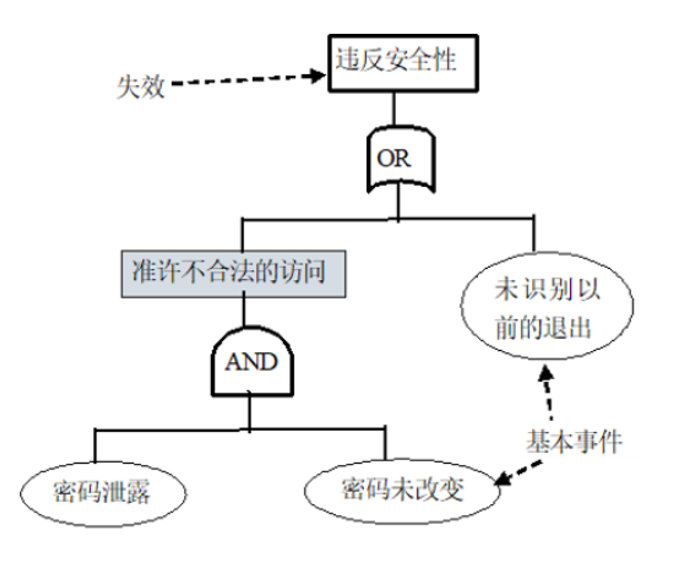


课本题：

| **活动** | **最早开始时间** | **最晚开始时间** | **时差（冗余时间）** |
| --- | --- | --- | --- |
| AB | 1 | 1 | 0 |
| BD | 4 | 4 | 0 |
| BI | 4 | 5 | 1 |
| DI | 9 | 9 | 0 |
| IJ | 11 | 11 | 0 |
| AE | 1 | 4 | 3 |
| EG | 5 | 8 | 3 |
| GJ | 8 | 11 | 3 |
| JL | 13 | 13 | 0 |
| AC | 1 | 5 | 4 |
| CF | 6 | 10 | 4 |
| FH | 9 | 13 | 4 |
| GH | 8 | 11 | 3 |
| HK | 11 | 14 | 3 |
| JK | 13 | 16 | 3 |
| KL | 15 | 18 | 3 |

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=20。

2. 下图是违反安全性Fault-tree故障树，请由它设计生成Cut-set-tree割集树。（5分）



三、综合设计题

1. 结合自己的项目开发时间，从体系结构、OO设计原则等（至少选两个案例）方面论述你在提高软件指令（易重用、维护等）方面所做的重构工作。（15分）

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：

① 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

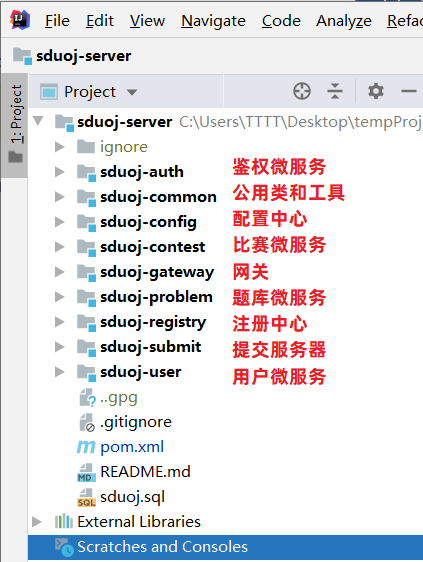
背景：

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：

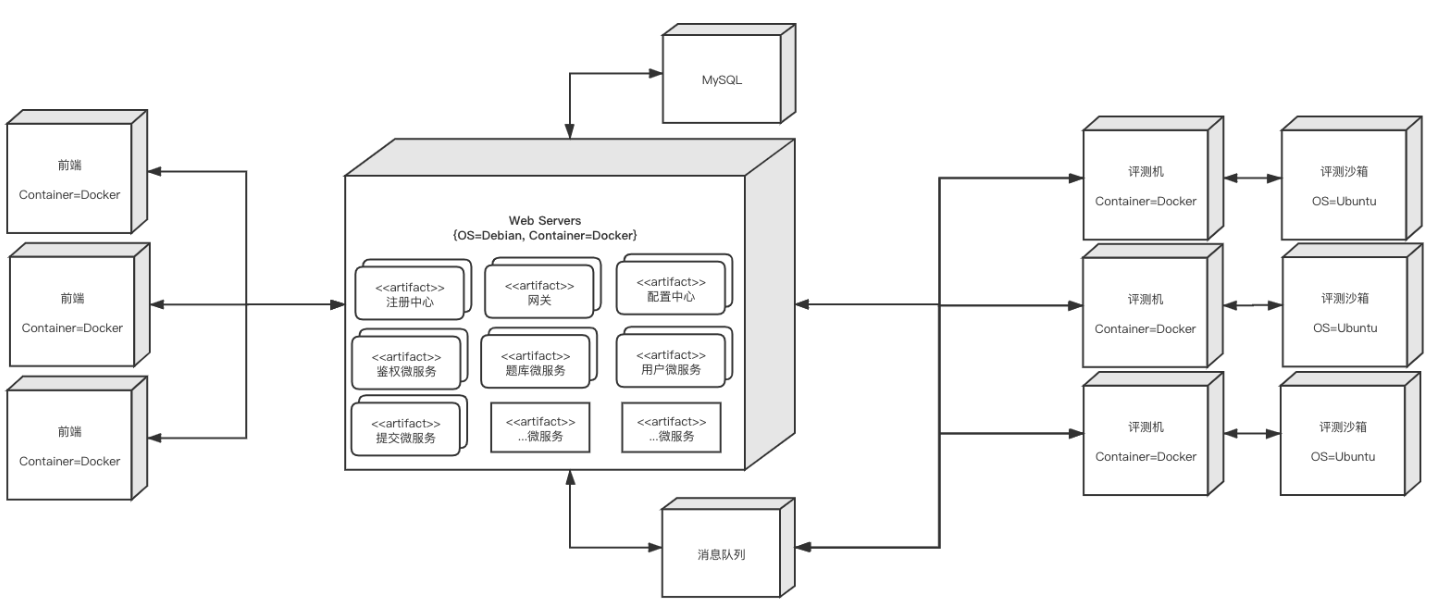
使用Spring Cloud提供的微服务组件，其中 Eureka 是服务注册中心，所有的微服务都要申请到它那里，进行服务注册；APIGateWay是网关，所有的请求都要经过它，它会将请求转发到具体的微服务；ConfigServe是配置中心，通过它可以轻松区分开发和测试环境，更新配置只需要推送到Git仓库，即可立即刷新，而无需重启服务器。

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

（2）使用面向对象设计原则（OO设计原则）所做的重构工作：

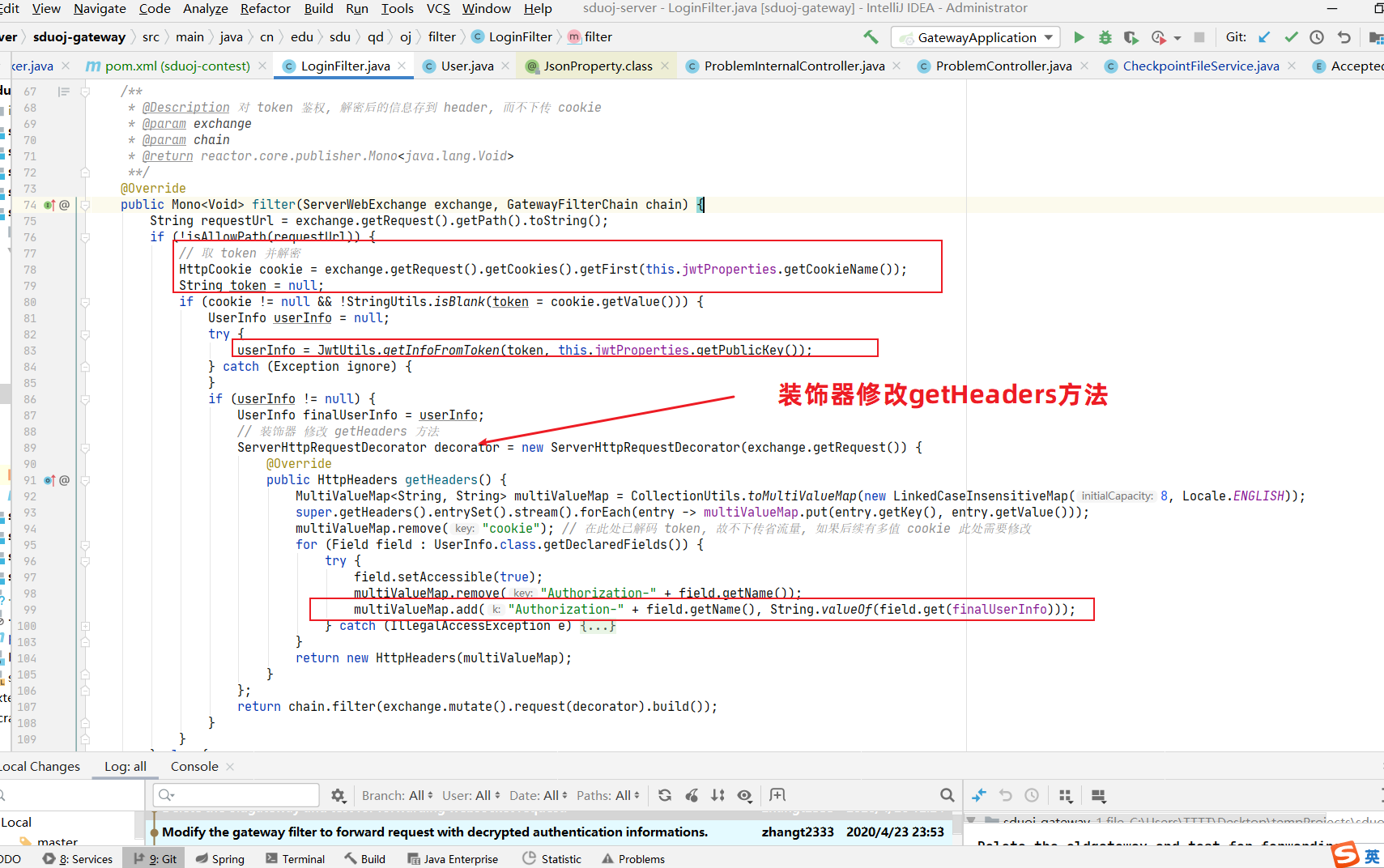
① 使用装饰器模式装饰网关转发流，在网关层解决权限鉴别问题，简化代码

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