第一章

1、请谈一下关于软件工程的整体认识和印象。

软件工程是指用系统化，模块化的工程性方法解决软件开发中的问题。简单来说，软件工程就是使用一种系统化的方法，解决软件开发中的各种困难。解决耦合度高等等的问题。

软件工程的目的是为了规范软件开发，推出高质量的软件产品。

使用软件工程可以降低开发成本，达到要求的软件功能、提高软件性能、提高软件的可移植性、降低维护费用、按时完成工作并交付使用。从而使系统开发更高效，成本更低，耦合度较低。

2、系统分析师和系统构架设计师有何区别?

系统分析师的主要工作是，分析系统的功能，能把用户的需求分解，根据用户的具体需求写出规格说明书，并给出系统的开发计划，开发的各个阶段等等。并且能评估和选用适合的开发方法和工具。可以说，系统分析师，主要的任务是分析系统，能给出系统在开发前的一些预先准备和规划。

系统架构设计师的主要工作是能够根据用户需求，涉及出正确，合理的软件架构及相关接口，并确保系统有良好的性能。并编写对应的设计文档。此外，还要对架构进行描述、分析和评估。

3、应用软件工程(实施分阶段原理等)会增加系统工作量吗?

软件工程学科其本身就是为了应对软件质量危机而出现的，对于现代成熟的工业软件来讲，如果完全不应用软件工程，会导致产品质量下降，日后维护等产生的工作量会更多，总体工作量看似降低，实则增加；但同时，如果使用了不合理的软件工程开发规范，简单项目使用了过于复杂的规范，也会让团队困在繁琐的规范中，产生不必要的工作量，同样会增加系统工作量。

综上，应用软件工程，使用合理的软件工程开发规范，从长远看一定会使总体工作量降低，不会增加系统工作量。

练习题2

问题：给出一个问题分析的例子,其中问题部分相对简单,但是解决问题的困难在于子问题之间的相互联系。

答：比如对于一个现代的直播系统来说，整体来看，直播的需求和功能较为明确，我们很容易做出一个单体的，前后端不严格分离的直播系统，实际上6年前的b站直播也是这么做的。

但是，单体服务架构会严重制约系统的发展，在监控告警，资源弹性，故障隔离等方面都会出现严重的问题。简单的例子就是，比如B站弹幕系统被打挂了，在单体服务架构下就会全面崩溃，会拖垮其他业务。微服务化势在必行！

另外的问题就是微服务化是复杂的，我们需要把如主播系统，礼物系统等从一个单体服务架构里去拆分出来，要做到的重要特性是一个子系统挂了不会影响另外的子系统。我们要怎么处理这其中所带来的问题，因为原本系统是一体式的，我们必然要处理拆分出来的子系统之间的消息互通关系，甚至要做到另外一个子系统下线期间，其他子系统应该做什么，要保证自己的正常运行。

如果还是以B站为例的话，查看他们项目组的技术报告可以发现他们是通过自行基于Swoole开发了自己的一套微服务化架构，重新封装协议，使用弹性流量池等一系列自研技术才解决了这个问题，这足以说明微服务化的复杂性。

总的来说，现代软件工程中，开发解决一个问题的软件是相对简单的，但当服务扩展的时候不可避免的需要进行微服务化。微服务化是困难的，因为我们往往要处理各个子系统之间的通讯交流困境，也要处理各系统之间的依赖关系，不能有过重的依赖导致微服务架构退化。

练习题3

问：解释错误、故障和失效之间的区别。举出一个关于错误的例子，并且这个错误导致了需求、设计、代码的故障。举出在需求中存在故障的例子，并且这个故障导致了失效。举出在设计中存在故障的例子，并且这个故障导致了失效。举出在测试数据中存在故障的例子，并且这个故障导致了失效。

答：错误是指在软件开发过程中人为产生的错误，比如代码中的错误。

故障是指软件功能实现过程中产生的问题，故障是由错误导致的，故障是软件中错误的表现。

失效是指系统的功能实现错了，或者系统的功能失效，即没有实现好应用的功能。

例子：还是以B站视频为例，对获取视频的CDN镜像获取不当，这个错误导致了系统故障【用户访问视频速度奇慢无比】，而系统故障导致了B站播放视频功能的失效。

第二章

1、如何称得上一名优秀的程序员?

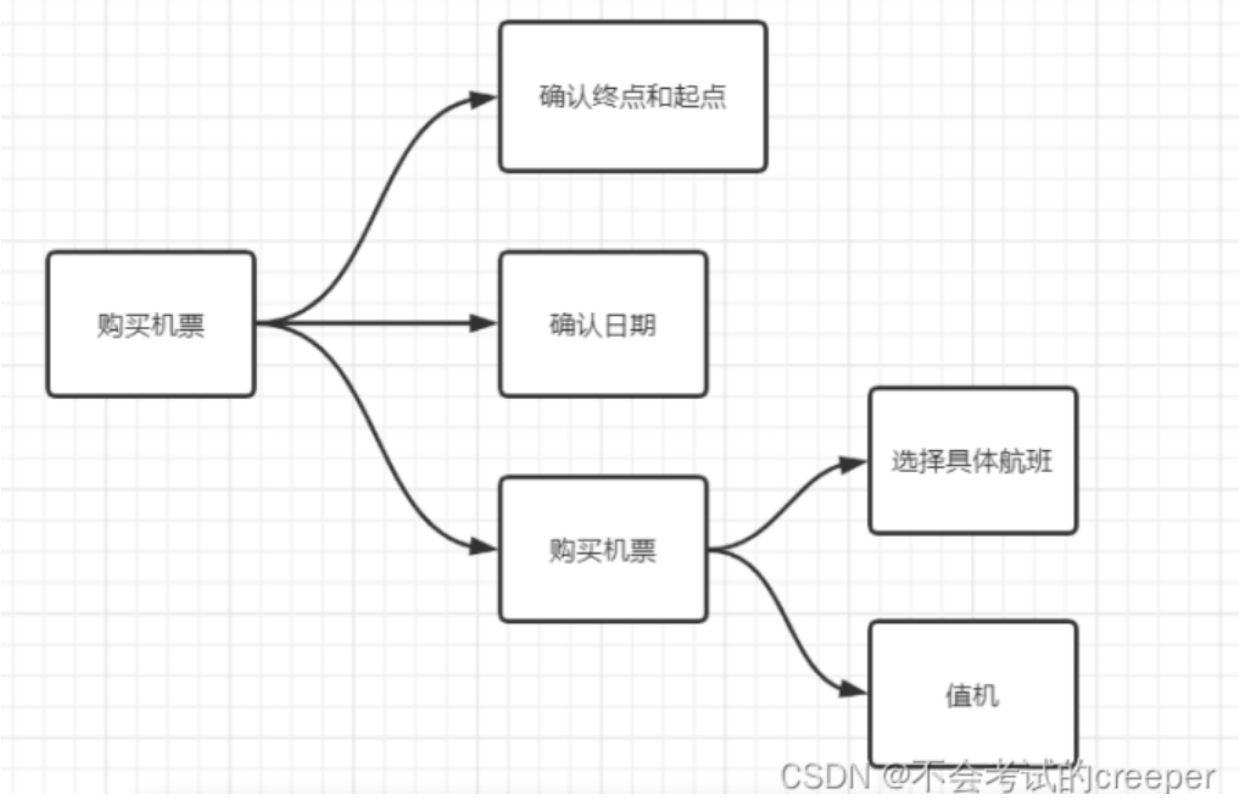
一名优秀的程序员不仅仅需要把代码写好，同时要在写代码的时候需要遵循相应的代码规范，并写好注释，方便后期维护。程序员还需要具有良好的沟通能力，由于现在企业中开发项目使用的是团队协作的方式进行开发，如何与队友协作，共同完成项目成为一个难题。处理了良好的代码能力和沟通能力之外，程序员还需要对项目的进展和预期进行估计，对项目需要的资金和时间进行相对准确的估计。

2、系统架构师应该具备的素质是什么?

系统架构师需要具备的素质是，很好地了解系统的所有需求，知道整个系统需要用什么架构，对实现各个功能的技术栈相当收悉。系统的开发周期大概需要多长，需要投入多少资金。除此之外，系统架构师还需要给出系统开发前的一些具体规划。

练习题4

问：画一个图，试描述为一次商务旅行购买一张飞机票的过程。



练习题11

问：考虑本章介绍的过程。哪些过程在你对需求变化做出反应时给了你最大的灵活性?

系统设计和程序设计，在需求发生变化时给了我最大的灵活性。

在系统设计和程序设计时，本身就要先考虑到未来可能产生的变动，在系统架构和代码实现上要留有一定的余地，带来可拓展性。

当需求发生变化时，我们需要对现有的软件进行修改。一个在可接受内的改动我们可能不需要更改原有架构，因为可拓展性可以让我们使用迭代手段来应对此次修改。但如果改动比较大，可能需要增加新的子系统来完成修改。

但在迭代到一定次数时，代码架构难免出现偏差，重新返回系统设计和程序设计进行重构也是必要的。

综上所述，系统设计和程序设计能在需求发生变化时，让项目团队获得最大的灵活性。但在迭代过多时，代码重构的代价可能也是昂贵的。

第三章

自定义问题

问：如果一个案例中涉及到合同管理，项目管理控制和项目沟通等诸多方面，在项目实际运行过程中，出现了甲方随意变更、不配合验收、甲乙双方沟通存在障碍.等情形，试问如何从合同管理、过程控制和项目沟通管理三个方面来应对?

答：合同管理：在与用户签订合同之前，需要签订相应的合同。在合同中要明确规定甲方随意变更，不配合验收等情形造成的后果由用户本人自行承担。在合同签订之后，一些细节可能随着系统更新需要调整，这些具体的细节同样需要在合同中规定清楚。

过程控制：在项目进行的过程中，要随时记录，跟进最新的状态，并写好对应的需求文档，在项目开发的过程中，还可以让用户提出意见，对于这一阶段的产品有什么其他的需求，再后续的阶段可以再加上。

项目沟通管理：在项目尚未开始时，需要指定项目进度和与用户沟通的时间表，在项目进行的过程中，需要及时与用户沟通进度。如果系统不满足用户需求，应及时更改。

练习题2

题目：图3-23是一个软件开发项目的活动图。对应于图中每条边的数字表示完成这条边代表的活动所需的天数。例如，完成终止于里程碑E的活动需要4天时间。对于每个活动，列出它的前驱，并计算最早开始时间、最晚开始时间和时差。然后确定出关键路径。

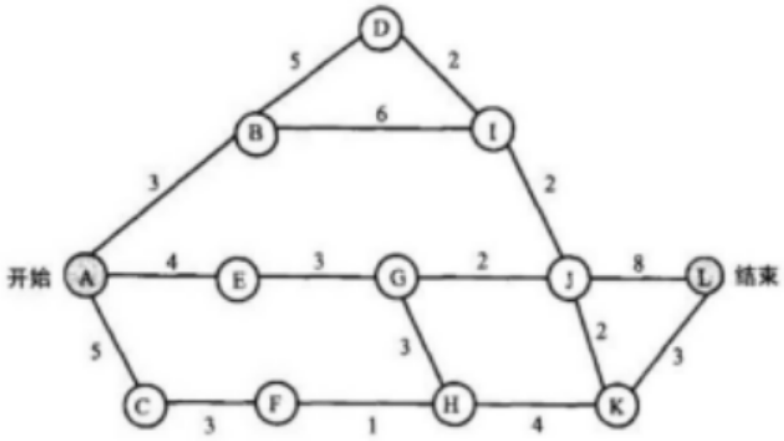


图3-23 练习2的活动图

（1）每个活动的前驱

A活动没有前驱，B活动的前驱是A，C活动的前驱是A，D活动的前驱是B，E活动的前驱是A，F活动的前驱是C，G活动的前驱是E，H活动的前驱是G和F，I活动的前驱是B和D，J活动的前驱是I和G，K活动的前驱是J和H，L活动的前驱是J和K

（2）每个活动的最早开始时间，最晚开始时间，时间差

关键路径为：A->B->D->I->J->L

练习题3
题目：图3-24是一张活动图。找出其关键路径

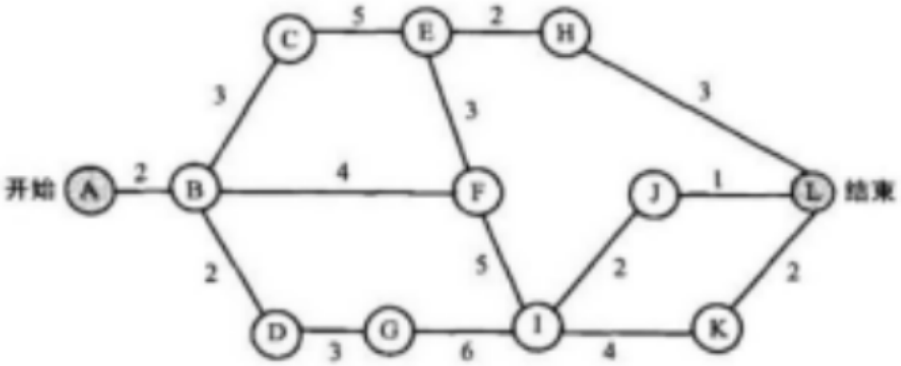


图3-24 练习3的活动图

关键路径为：A->B->C->E->F->I->K->L

练习题12

很多项目经理根据过去项目中程序员的生产率来计划项目的进度，生产率通常根据单位时间的单位规模来测量。例如，一个组织机构可能每天生产300行代码或每月生产1200个应用点。用这种方法测量生产率合适吗?根据下列事项讨论生产率的测度:

●用不同的语言实现同样的设计，可能产生的代码行数不同。

●在实现开始之前不能用基于代码行的生产率进行测量。

●程序员可能为了达到生产率的目标而堆积代码。

1）使用不同语言进行设计，产生的代码量差异巨大。如果使用C语言或C++，甚至是Java代码量很大，但如果使用python语言或者Go，则代码量较少。因此不能简单地按照语言的行数判断生产率。

2）在实现开始之前，虽然已经对工程量进行了估计，但项目的难度和具体的难点可能还未知。比如开发一个OS内核的速度显然要慢于开发一个简单的web项目，以程序员过去写代码的速度估计本项目的速度不一定合适。

3）当程序员没办法按时写出达到数量的代码时，可能为了完成任务而应付，导致很多无用或冗余的代码，比如堆积大量无用的if else语句。这对系统的代码架构甚至性能来说是一个问题

第四章

1、如何拒绝需求分析过程中某些不合理的用户需求?

分情况进行处理:

1.用户对于系统了解不够时，适当增添使用文档即可。

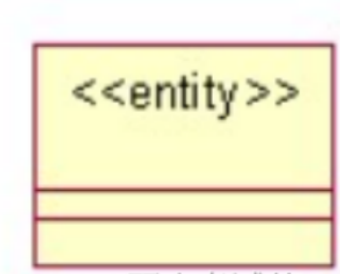
2.用户对于新功能的要求明显违反了现有逻辑，即很有可能会有BUG产生，需要用通俗易懂的语言向用户解释清楚。

3.用户所需的新功能合理但改动难度大，比如说需要对软件架构进行调整等，可以根据开发前所签订合同进行相关处理，也要将如果改动将产生的工期变动，价格变化等告知客户。

总的来说，当用户提出不合理的需求，需要向用户说明清楚造成的影响。双方需要多沟通，互相了解对方的难处。

2、分析阶段类的大致种类有边界类、实体类、控制类。请简述其具体含义(包括画法)
实体类：

实体类是用于对必须存储的信息和相关行为建模的类。实体对象（实体类的实例）用于保存和更新一些现象的有关信息，例如：事件、人员或者一些现实生活中的对象。实体类通常都是永久性的，它们所具有的属性和关系是长期需要的，有时甚至在系统的整个生存期都需要。



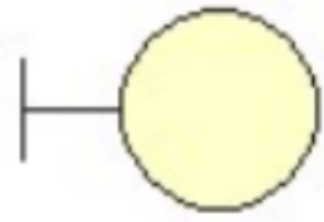
边界类：

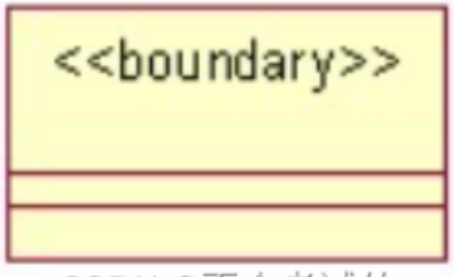
边界类是系统内部与系统外部的业务主角之间进行交互建模的类。边界类依赖于系统外部的环境，比如业务主角的操作习惯、外部的条件的限制等。它或者是系统为业务主角操作提供的一个GUI，或者系统与其他的系统之间进行一个交互的接口，所以当外部的GUI变化时，或者是通信协议有变化时，只需要修改边界类就可以了，不用再去修改控制类和实体类。业务主角通过它来与控制对象交互，实现用例的任务。

边界类调用用例内的控制类对象，进行相关的操作。

一个系统可能会有多种边界类：

用户界面类 - 帮助与系统用户进行通信的类
系统接口类 - 帮助与其他系统进行通信的类
设备接口类 - 为用来监测外部事件的设备（如传感器）提供接口的类





控制类：

控制类用于对一个或几个用例所特有的控制行为进行建模，它描述的用例的业务逻辑的实现，控制类的设计与用例实现有着很大的关系。在有些情况下，一个用例可能对应多个控制类对象，或在一个控制类对象中对应着对个用例。它们之间没有固定的对应关系，而是根据具体情况进行分析判断，控制类有效将业务逻辑独立于实体数据和边界控制，专注于处理业务逻辑，控制类会将特有的操作和实体类分离，者有利于实体类的统一化和提高复用性。

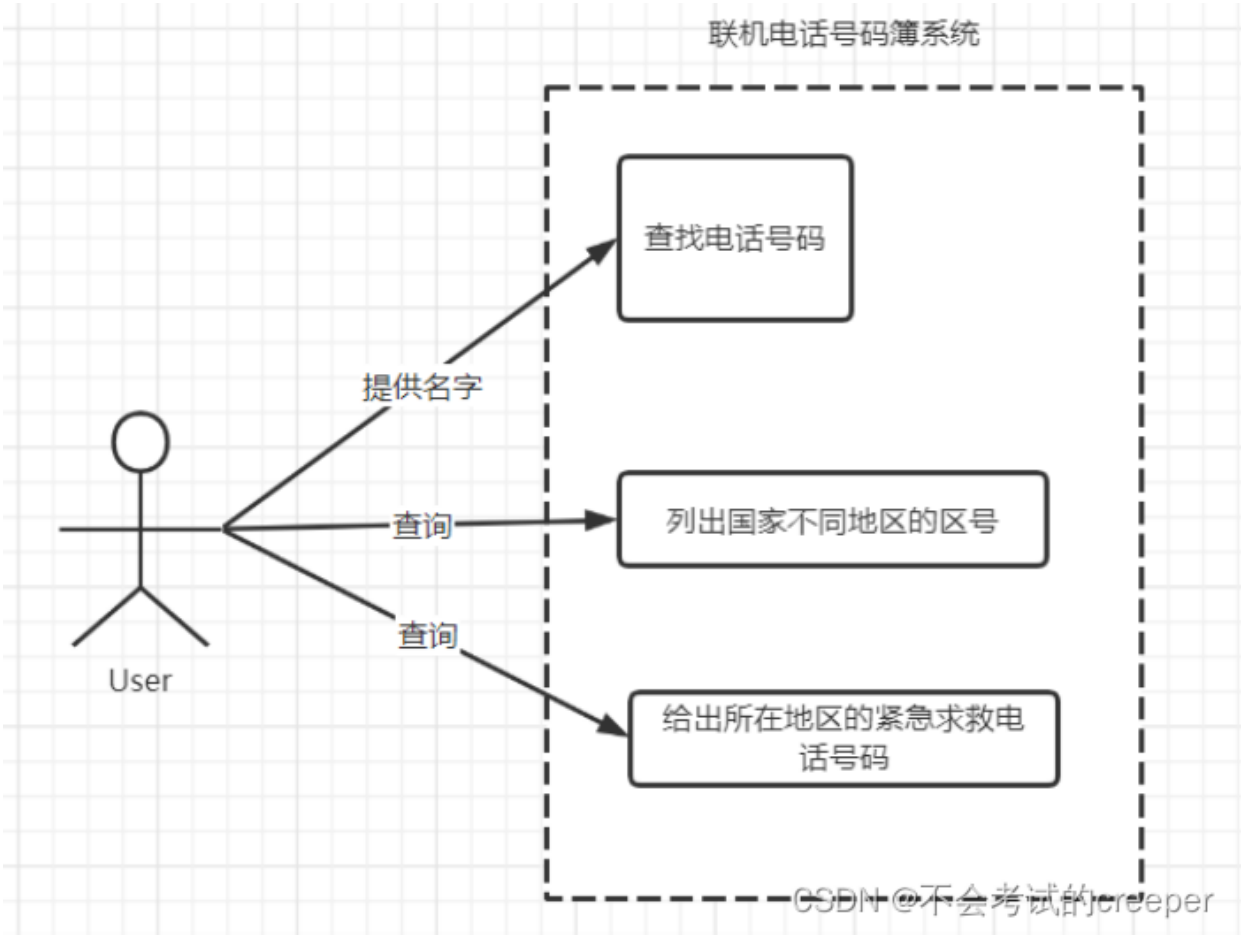
当业务主角通过边界类来执行用例的时候，产生一个控制类对象，在用例被执行完后，控制类对象会被销毁。

当然如果用例的逻辑较为简单，可以直接利用边界类来操作实体类，而不必再使用控制类。或者用例的逻辑较为固定，业务逻辑固定不会改变。也可以直接在边界类实现该逻辑。【类似于MVC三层中的做法】

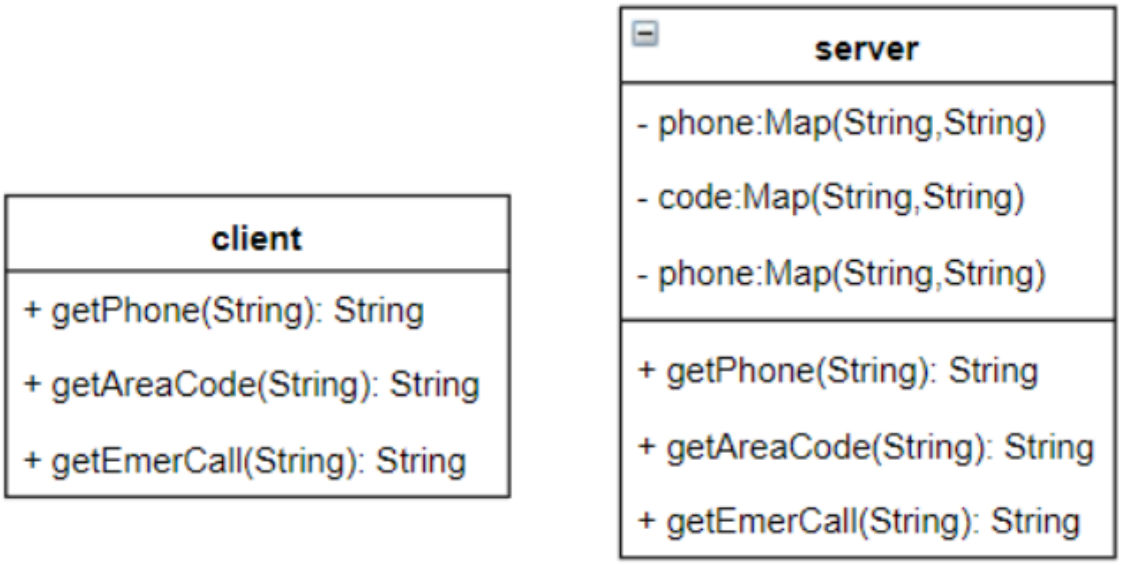
练习题12

问：针对用联机电话号码簿替换电话公司给你的号码簿这样的问题。编写一套UML模型(用例图、MSC图、类图)。给出名字时，要求该号码簿应该能够提供电话号码;它还应该能够列出国家不同地区的区号，并给出你所在地区的紧急求救电话号码。

答：用例图：


联机电话号码簿系统
查找电话号码
提供名字
查询
列出国家不同地区的区号
查询
User
给出所在地区的紧急求救电话号码
CSDN @不会考试的Creeper

UML类图：



这里只给出两个client和server，实际上是前后端分离的系统，实际上server和client内聚不可能这么简单，就不详细给出了。

网上实例
问：小米校招产品作业解读:设计一款网络日记APP。(画像课堂作业)

答：

(一) 背景及需求分析

背景：

在当代人繁忙的生活下，许多人已经没有时间再停下来细细去写一本日记了，在万事万物都走上线上的时刻，日记也应该可以走上我们的手机或者IPad，让繁忙的现代人也能抽空享受这份美好，记录生活的点滴。

需求分析：

1.审美需求

页面，字体等应该可以更换，对图片等应该有较高兼容性。

2.保密/分享需求

个人日记可以是私人性质的，也可以是分享性质的，在这方面是类似于QQ空间中的，仅自己可见/所有人可见。

3.音频，影像记录需求

电子日记可以跳脱于传统日记，对传统难以保存的音频，影像等插入电子日记中，获得更丰富的日记体验。

4.保存需求

实时保存，防止用户临时退出。同时，提供对写过的日记全量的储存，存储用户的日记回忆。

5.图形多样性需求

用户在使用网上日记APP时，有使用多种图形的需求，如课程表等。

（二）用户画像

（1）高中男生，希望记录自己的学习规划，每天的新的体会，对课程表，数学工具等有强烈需求

（2）女白领：希望记录自己的日常心情，对保密性，美观性要求较高

（3）网红：分享欲强烈，希望能把自己的生活点滴分享给大家

（三）确定方向

（1）UI的设计时尚而朴素，并提供多种主题供用户选择

（2）采用类似真实日记本的翻页形式记录日记，并且对背景颜色，字体等提供多种选择，支持用户自定义背景图片。

（3）支持录音或导入音像到日记中，产生丰富的日记效果

（4）日记的样式支持公开和仅自己可见

第五章

练习题4

问：给出一个用原型开发但并没有节省大量开发时间的例子。

一般来说，如果基于原型开发，会节省大量的时间，这是因为，如果我们基于原型开发，可以先将快速开发形成的“软件样机”以可视化的形式展示给用户。

但基于原型开发不一定能节省开发时间，因为不是所有原型都可以得来那么简单，比如一个编译器等等，即如果在原型开发中做了大量的精细化工作，那么原型开发可能会比较耗时。这是由于原型开发的内容一般要全部抛弃导致的。

练习题5

问：列举出最适合原型设计的系统的特征。

在原型设计时，产品经理需要先明确系统对应的界面等，设计优于开发。

适合原型设计的系统有以下特征：

第一是软件数据流明确，系统简单；这是因为具有该特征的系统往往易于模拟，不会需要大量的细节化工作。

第二是涉及面狭窄的小系统；强调系统小的特性，大型系统必然出现难以模拟的困境，同时强调涉及面不要太广，涉及面广的时候往往会直接击垮团队。

上一章第12题的延续:继续进行设计阶段的工作

上一章已经给出了UML和用例图。我们现在对后端流程进行进一步设计。

查找号码功能：我们考虑号码库过于庞大的情况，由名字查找号码又难以从数据结构上加快速度。那我们不妨模拟一个缓存区，把经常命中的号码放在缓存中，期望以此加快查询速度。

列出区号：按照国家，地区分类存储数据，加索引，提升速度。

给出所在地求救电话：在这里使用GPS是必要的，或者让用户手动输入所在地，返回求救电话即可。

第六章

练习题1

皇家汽车服务站经理Manny准备扩展他们的服务，使之包括洗车服务。这将是一个自动化系统。客户选择洗车的方式并注明车的类型。系统计算费用并在控制面板上显示应付金额。然后客户支付洗车费。支付结束后，如果洗车正忙的话，系统就会提示客户必须等待，否则，系统提示客户将车驶进洗车间。请问，如何对图6-21所示的类图进行修改才能提供这种新服务。

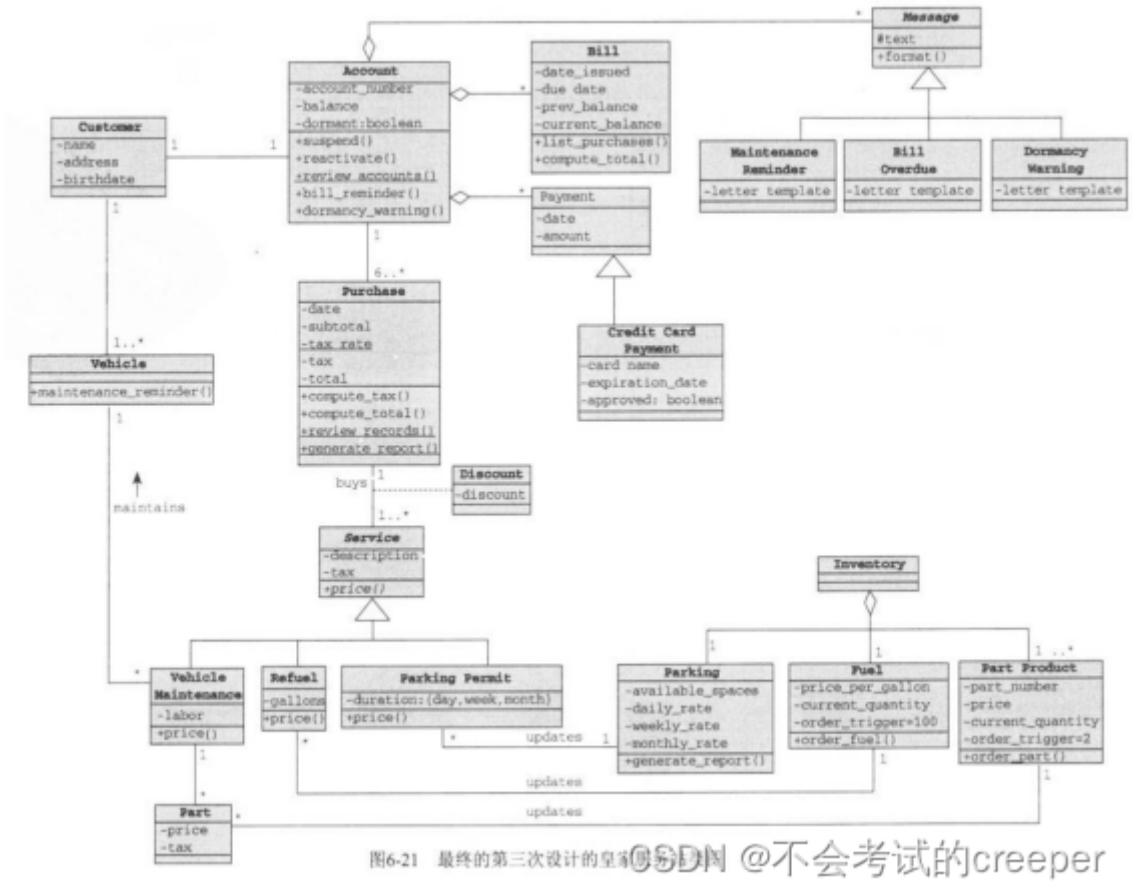


图6-21 最终的第三次设计的皇家服务站类图

首先定义一个洗车服务类，实现service接口

WashingCarService
-washType（洗车方式）
-carType（汽车类型）
+price()

接着定义一个确定洗车库存的类，实现Inventory接口

CarFactory
-message（如果洗车正忙，向用户发送的消息）
-washType
-carType
-price
+washingCars()

上述两个类显然和别的service和inventory关系相同

这样直接将上述两个类嵌入原本UML即可

第八章

1、 模拟面试问答:如果让您来带领一个测试团队，您会做哪些工作?

如果让我来带领一个团队，第一步刚到时应该是对每位成员进行一定的了解，通过聊天，查看简历等方式去了解他们的性格特点和能力特点，根据他们所擅长的去安排不同工作。

不妨从接受一个测试项目开始讲起，首先自己要对这个项目有个总体的感知，抓住项目的重点，整体衡量项目的各个模块及开发人员的情况，做到心中有数，同时，要统一部署好项目环境，这也是对后面打好基础。完成上面两项以后就开始实际部署工作，这个过程中要制定合理的测试计划，合理分配个人任务，同时安排的时候也要预留解决难题和整合系统的时间。在完成任务的过程中，注重沟通、协调、跟进把关，协调商定具体方案，在质量上要由有把关，先易后难，有针对性有计划性的完成任务。当然，在测试工作完成后，还要进行必要的复盘总结，进一步提高团队凝聚力和战斗力。

2、 对于需求文档，如何进行测试?(从什么方面考虑?有什么样的原则?)

主要包括:

正确性：对照原始需求检查需求规格说明书。

必要性：不能回溯到出处的需求项可能是多余的。

优先级：恰当地划分并标识。

明确性：不使用含糊的词汇。

可测性：每项需求都必须是可验证的。

完整性：不能遗漏必要和必需的信息。

一致性：与原始需求一致、内容前后一致。

可修改性：良好的组织结构使需求易于修改。

练习题7

问：图8-22说明了一个软件系统中的构件层次。分别描述用自底向上方法、自顶向下方法、改进的自顶向下方法、一次性集成方法、三明治方法和改进的三明治方法对构件进行集成测试的顺序。

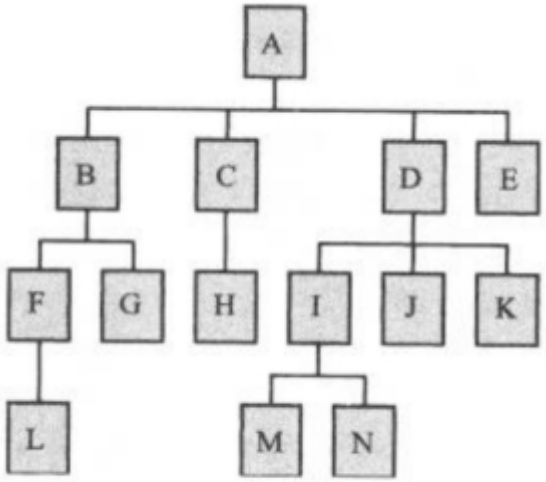


图9-22 构会层次的例子

自底向上：LMN -> LMNFGHIJK -> LMNFGHIJKBCDE -> LMNFGHIJKBCDEA

自顶向下：A -> ABCDE ->ABCDEFGHIJK -> ABCDEFGHIJKLMN

改进的自顶向下：A -> B,C,D,E-> ABCDE -> F,G,H,I,J,K -> ABCDEFGHIJK -> L,M,N -> ABCDEFGHIJKLMN

一次性集成：每个模块分别测试，而后全部集成（先测A,B,C....N，之后再即成为ABCD.....MN）

三明治方法：【我们不妨把BCDE看成目标层】

A,L,M,N -> LF,G,H,IMN,J,K -> BFGL,CH,DIJKMN -> ABCDEFGHIJKLMN

改进的三明治方法：【我们不妨把BCDE看成目标层】

A,L,M,N -> LF,G,H,IMN,J,K ->B,C,D,E ->BFGL,CH,DIJKMN -> ABCDEFGHIJKLMN

第九章

练习题5

问：在第4章中，我们讨论了需求的可测试性的必要性。解释为什么可测试性对性能测试是重要的。举例说明你的解释。

性能测试指通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。而可测试性是一个系统能被测试的难易程度，这个难易程度是用可控制性、可观察性、有效性、简单性、稳定性、信息化几个方面来评价的。

那么对于性能测试来说，可测试性是非常重要的。比如在压力测试下，我们能不能有可预测的结果——稳定性，能不能知道导致压力瓶颈的是什么部件——可观察性，等等。如果一个软件可测试性较差，那压力测试必然是如履薄冰的，测试了也难以知道哪里出错等等。

练习题6

问：字处理系统、工资单系统、自动银行连线柜员系统、水质监控系统、核电站控制系统分别需要哪种类型的性能测试?

字处理系统：计时测试，人为因素测试，恢复测试，质量测试

工资单系统：计时测试，安全性测试，人为因素测试，恢复测试，质量测试

自动银行连线柜员系统：安全性测试，计时测试，压力测试，质量测试

水质监控系统：环境测试，兼容性测试，质量测试，质量测试，恢复测试，配置测试，维护测试，文档测试

核电站监控系统：容量测试，环境测试，兼容性测试，安全性测试，计时测试，恢复测试，配置测试，维护测试，文档测试

练习题8
问：一个制导系统将安装在一架飞机上。当设计安装测试时必须考虑什么问题?

答：首先常规的，我们要进行回归测试，证明系统实际安装完成后系统仍然是完备的。再者特殊的，我们要验证任何可能受到运行场地所影响的功能和非功能特性，比如我们的系统在高空下能否正常运转，这要考虑高空下物理传感器是否有我们未处理的错误等等。

练习题12
问： 一个工资单系统的设计要求在该公司工作的每名员工都有一个雇员信息记录。每个星期对雇员记录更新一次，记录该雇员本星期工作的小时数。每两星期打印一次汇总报表，显示从会计年度开始的总工作小时数。每月每名雇员的该月收入被转到他的银行账户中。针对照本章所介绍的每一种类型的性能测试，介绍它们是否适用于该系统。

答：

简单分析题目：

1、该公司工作的每名员工都有一个雇员信息记录

2、每个星期对雇员记录更新一次，记录该雇员本星期工作的小时数【这应该是自动的，由打卡机或其他提供数据】

3、每两星期打印一次汇总报表，显示从会计年度开始的总工作小时数【这也是自动的，数据来源打卡机和其他记录，还要处理打印机的硬件接口】

4、每月每名雇员的该月收入被转到他的银行账户中【自动的，要处理银行的下游接口】

总结，这不像一个需要每个员工都访问的系统，访问量不大，数据量有上升的可能，但自动化特征明显，需要可靠的运行。

性能测试：

压力测试：主要压力来源是周期性的234，实际上做的时候很可能采用队列缓冲形式，而不会让其突发，不需要压力测试。

容量测试：公司有扩张的可能，且所有记录随着时间成倍增长，数据量有超出承受范围的可能，需要容量测试。

配置测试：一定的软硬件性能是必要的，特别是数据存储方面，需要配置测试。

兼容性测试：需要和银行等软件接口交互，也需要和打印机等硬件交换，显然需要兼容性测试。

回归测试：看实际开发情况。

安全性测试：涉及到公司财务，需要安全性测试。

计时测试：可以检测周期性任务的完成时间，计时测试是可选项，主要看客户需求。

环境测试：无特殊环境，不需要环境测试。

质量测试：系统将长期自动运转，需要较高质量。需要质量测试。

恢复测试：停电等事情时有发生，需要恢复测试。

维护测试：若有提供软件自动修复等工具，则需要维护测试。

文档测试：对象非专业人员，显然需要文档，需要文档测试。

人为因素测试：对象非专业人员，可能发生各种意外情况，需要人为因素测试