Chapter01

1.SE的定义、目的、方法及作用

SE：在将有关软件开发与应用的概念科学体系化的基础上，研究如何有计划、有效率、 经济地开发和利用能在计算机上正确运行的软件理论和技术工程的方法学，以及一些开发和维护软件的方法、过程、原则等。它是一个系统工程，既有对技术问题的综合分析，也有对开发过程和参与者的管理。
SE 的方法：面向对象模式，结构化模式，基于过程的模式等。
SE 的作用：付出较低的开发成本，达到要求的软件功能，取得较好的软件性能，开发的软件易于移植，需要较低的维护费用，能按时完成开发工作，及时交付使用。
// 开发模式（paradiam）

2.说明错误、缺陷、失败的含义与联系。（请举例说明）

错误(error)：是在软件开发过程中人为产生的错误（需求说明中的错误，代码中的错误）。
故障(fault)：软件功能实现过程中产生的问题，是错误导致的结果，是软件中一个错误的表现（一个错误可能产生多个故障，静态存在）。
失效(failure)：系统违背了它应有的行为（在系统交付前或交付后被发现，动态存在）。
联系：人为原因导致程序错误；该错误编译到系统中导致系统故障；用户使用该系统时，因故障导致失效。故障是系统内部视图，从开发者的角度看待问题；失效是系统外部视图， 从用户角度看到的问题。而且并不是所有的故障会导致失效，只要不执行故障代码，或者不进入某个特定状态，那么故障就不会使代码失效。

3.软件质量应从哪几个方面来衡量？论述之。简答

产品(product)的质量
用户：从失效的数目和类型等外部特性进行评价，如果软件具有足够的功能，并且易于学习和使用；或者虽然难以学习和使用，但是由于功能值得这些付出，用户就断定软件是高质量的。
开发者：从故障的数目和类型等内部特征来作为产品质量的依据。
过程(process)的质量
有很多过程都会影响到最终的产品质量，只要有活动出了差错，产品的质量就会受到影响；开发和维护过程的质量与产品的质量是同等重要的。
商业(business)环境背景下的质量
(1) 技术价值与商业价值的联系与区别：
技术价值：技术指标（速度，正确的运行时间，维护成本等）。 商业价值：机构对软件是否与其战略利益相吻合的一种价值评估。 误区：技术质量不会自动转化为商业价值。
(2) 目标
将技术价值和商业价值统一起来，改进过程所带来的商业价值。
// 软件系统的系统组成。

4.现代软件工程大致包含的几个阶段及各个阶段文档。

（1）需求分析：包括问题定义、可行性研究、需求分析【《SRS》即《软件需求规格 说明书》】与复审（所有人）。
（2）系统设计：包括用户界面的设计【《SAD》即《软件系统结构图》：如何制作软 件】与复审（开发者与客户）。
（3）程序设计：包括模块功能算法与数据描述设计【相关文档】与复审（开发者）。
（4）程序实现：包括编程与 debug【源代码和注释】与复审（开发者、码农）。

（5）单元测试：模块功能测试与性能测试【测试报告】与复审（测试团队）。
（6）集成测试：按照结构图进行测试【测试报告】与复审（测试团队）。
（7）系统测试：按《SRS》对系统总体功能进行测试与复审（开发者与客户）。
（8）系统提交：交付产品【用户手册和操作手册】与复审。
（9）系统维修：修改软件的过程，为改错或满足新需求【维修报告】与复审（维修团 队）。

//使现代SE实践发生变化的（七个）关键因素是什么?

5.什么是软件过程？软件过程的重要性是什么？包含几个阶段?

软件开发活动中的各种组织及规范方法（课件定义）
软件开发过程描述了软件产品从概念到实现、交付、使用和维护的整个过程，因此，有时把软件开发过程称为软件生命周期。
（1）它强制活动具有一致性和一定的结构。
（2）过程结构允许我们分析、理解、控制和改进组成过程的活动，并以此来指导我们的活动。
（3）它使我们获取经验并把经验传授给他人。

6.什么是重用、抽象等现代软件工程主要概念?

抽象(abstraction)：基于某种层次归纳水平的问题描述。它使我们将注意力集中在问题的关键方面而非细节。
分析、设计方法和符号描述系统：
使用标准表示来对程序进行描述。利于交流，利于建模并检查其完整性和一致性，利于 对需求和设计部件进行重用。
用户界面原型化(prototyping)：
建立系统的小型版, 通常具有有限的关键功能,以利于用户评价和选择，证明设计或方法的可行性。
软件体系结构：定义一组体系结构单元及其相互关系集来描述软件系统。 单元分解的方法
软件过程：软件开发活动中的各种组织及规范方法。
重用或复用(reuse)名词解释：重复采用以前开发的软件系统中具有共性的部件, 用到新的开发项目中去 (注: 这里的重用绝不仅仅是源代码的重用)。
测度或度量(measurement)：通用的评价方法和体系，有助于使过程和产品的特定特性更加可见，包括量化描述系统、量化审核系统。
工具和集成环境：通过框架比较软件工程环境提供的服务，以决定其好坏。工具：由于厂商很少针对整个开发生命周期，因此对于工具的比较集中于小的活动集，例如测试或设计。

Chapter02

1.什么是软件过程？软件过程的重要性是什么？软件生命周期?

软件开发活动中产生某种期望结果的一系列有序任务，涉及活动、约束和资源。

1.通用性，一致性，结构性。一致性和结构性可以使我们知道是否已经做好了工作，还能使别人以同样的方式做工作，因而具有相对通用性。
2.自我指导。总结经验，完善规范，指导新版本

3. 过程改进；流程改善 process improvement

软件开发过程描述了软件产品从概念到实现、交付、使用和维护的整个过程，因此，有时把软件开发过程称为软件生命周期。

2.瀑布模型及各阶段文档，优缺点?

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。

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。
缺点简答（并提问，这种缺点出现的原因）：
（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。
（2）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。
（3）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）。

3.原型的概念与用途。

一种部分开发的产品，用来让用户和开发者共同研究，提出意见，为最终产品定型。
原型可以理解为小样，在某一阶段产品定型前先做一些小样，通过对各种样品的评价和分析，并最终为产品定型。

V 模型与瀑布模型的区别：
（1）V 模型使得隐藏在瀑布模型中的迭代和重做活动更加明确。
（2）瀑布模型关注文档和制品，V 模型关注活动和正确性。

4.论述分阶段开发模型的含义, 其基本分类及特点是什么?

系统被设计成部分提交, 每次用户只能得到部分功能, 而其他部分处于开发过程中。
循环时间: 软件开发时整理需求文档时间与系统提交时间之差

增量开发：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。
迭代开发：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。
将增量开发和迭代开发相结合：一个新发布的版本可能包含新功能，并对已有功能做了改进

5.螺旋模型四个象限的任务及四重循环的含义？ 名词解释

螺旋模型每次迭代有四个任务，依次是计划、目标/可选方案、风险评估、开发与测试。
螺旋模型共有四次迭代，依次是操作概念、软件需求、软件设计、开发与测试。每一次迭代都根据需求和约束进行风险分析，以权衡不同选择，并且在确定选择之前，通过原型化验证可行性和期望度。

//------ 习题2， 3。
// 在所有的软件开发过程模型中，你认为哪些过程给予你最大的灵活性以应对需求的变更?

6.什么是UP， RUP，进化式迭代等市场流行的过程模型?

统一过程（UP）：它是用例驱动的、以基本架构为中心的、迭代式和增量性的软件开发过程框架，它使用对象管理组织（OMG）的UML 并与对象管理组织（OMG）的软件过程工程原模型（SPEM）等相兼容。
“统一过程”将重复一系列生命期，这些生命期构成了一个系统的开发期寿命。每个生命期都以向客户推出一个产品版本而结束。
每个周期包括四个阶段：开始阶段、确立阶段、构建阶段和移交阶段。每个阶段可以进一步划分为多次迭代。
RUP（Rational Unified Process），统一软件开发过程，统一软件过程是一个面向对象且基于网络的程序开发方法论。
进化式迭代开发（Iterative development）是统一开发过程(RUP)的关键实践。开发被组织成一系列固定的短期小项目。每次迭代都产生经过测试、集成并可执行的局部系统。每次迭代都具有各自的需求分析、设计、实现和测试。随着时间和一次次迭代，系统增量式完善

Chapter03

1.什么是项目进度？活动？里程碑？项目成本?

项目进度：名词解释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。
活动：项目的一部分，一般占用项目进度计划中的一段时间
里程碑（Milestone）指特定的时间点，标志着活动的结束，通常伴随着提交物。（如一般性文档，功能模块的说明，子系统的说明和展示，精确度的说明和展示，可靠性，安全性，性能说明或展示文档）

2.如何计算软件项目活动图的关键路径？（习题2，3）冗余时间？最早和最迟开始时间（课堂习题讲解）

一个选择，问哪个点在关键路径上
关键路径（Critical Paths）：从起点到终点总花费时间最长的路径，即这个项目的最短完成时间，因为如果这条路径无法完成那么整个项目都不能算完成。所以这条路径上的任务耽误一点都会影响最后项目完 成时间。

// 软件团队人员应该具备的能力是什么?

3.软件项目团队组织的基本结构?

一个选择题
(1) 主程序员负责制（Chief Programmer Team）
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。
优势：
使交流最小化迅速做出决定
缺点：
创造性低
对主程序员要求高，个人主观性强
(2) 忘我方法（Egoless Approach）
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。
(3) 项目组织的结构化
结构化较强的团队：

按时完成任务，单工作比较循规蹈矩，项目普通但是功能完备。适合人员较多，项目稳 定性和一致性高，使用较正规的结构。
结构化较弱的团队：
不能按时完成任务但是创造性强，涉及大量的不确定性因素时采用较为民主的方法和相 关的团队结构

//专家估算法的大致含义？算式估算法的大致含义？

4.试述COCOMO模型的三个阶段基本工作原理或含义。

COCOMO 模型的关键在于针对项目开发的不同阶段来设置工作量的衡量标准，逐步细化， 逐渐准确。
E = bSc m(X)

在阶段一，项目通常构建原型以解决包含用户界面、软件和系统交互、性能和技术成熟 性等方面在内的高风险问题。这时，人们对正在创建的最终产品可能的规模知之甚少，因此COCOMOⅡ用应用点来估算规模。
在阶段二，即早期设计阶段，已经决定将项目开发向前推进，但是设计人员必须研究几种可选的体系结构和操作的概念。同样，仍然没有足够的信息支持准确的工作量和工期估算， 但是远比第一阶段知道的信息要多。在阶段二，COCOMOⅡ使用功能点对规模进行测量。
在阶段三，即后体系结构阶段，开发已经开始，而且已经知道了更多的信息。在这个阶 段，可以根据功能点或代码行来进行规模估算，而且可以较为轻松地估算很多成本因素。

5.什么是软件风险？ 了解主要风险管理活动？有几种降低风险的策略?

概念：软件生产过程中不希望看到的，有负面结果的事件。

一个选择，问：哪个 辅助项目小组进行风险应对的策略制定（大体是这个意思）一脸蒙，我觉得哪个都对策略有辅助作用
风险评价：风险识别，风险分析，风险优先级分配
风险控制：风险降低，风险管理计划，风险化解。
避免风险（Avoiding the risk）一个判断：改变功能和性能需求，使风险没机会发生。比如用 C 语言的程序有内存泄漏的风险改用 Java，避免风险。
转移风险（Transferring the risk）：通过把风险分配到其他系统中，或者购买保险以便在风险成为事实时弥补经济上的损失。
假设风险（Assuming the risk）：用项目资源，接受并控制风险。比如在开发时主动有意识地进行测试。

6.弄懂活动图基本原理（参考课本），找出课后练习题图3.23和3.24的关键路径。

Chapter04

1.需求的含义是什么?

定义：对来自用户的关于软件系统的期望行为的综合描述，涉及系统的对象、状态、约束、功能等。

2.需求阶段作为一个工程，其确定需求的过程是什么？简答

①原始需求获取：客户给出的需求
②问题分析：理解需求并通过建模或模型化方式进行描述
③规格说明草稿：利用符号描述系统将定义规范化表示
④需求核准：开发人员与客户进行核准
⑤软件规格说明（SRS）

3.举例说明获取需求时，若有冲突发生时，如何考虑根据优先级进行需求分类。

①必须要被满足的需求
②非常值得做但是不是必须的需求
③可选的需求（可做可不做）

// 如何使需求变得可测试？（sidebar4.4)

4.需求文档分为哪两类?

(1) 需求定义
完整罗列了客户期望的需求
(2) 需求规格说明（SRS）
将需求重述为关于要构建的系统将如何运转的规格说明。

5.什么是功能性需求和非功能性需求/质量需求？ 设计约束？过程约束？如何区分?

①功能需求名词解释：描述系统内部功能或系统与外部功能的交互作用，涉及系统输入应对、实体状态变化、输出结果、设计约束、过程约束等。
②设计约束：已经做出的设计决策或限制问题解决方案集的设计决策。涵盖物理环境、接口、用户等方面。
③过程约束：对用于构建系统的技术和资源的限制，涵盖资源、文档等方面。
④非功能需求：描述软件方案必须具备的某些质量特征，例如系统性能、安全性、响应 时间等。
// 需求的特性？（正确性、一致性、完整性）。
正确性；一致性；无二义性；完备性；可行性；相关性；可测试性；可跟踪性

6.了解DFD图 数据流图 的构成及画法。

这一部分需要知道各种图的组成要素和用途。综合起来考了个选择。大体就是选择什么图来表达什么。

数据流图：描述数据进入、转换、离开系统，重点在于数据流，而不是控制流。
状态图：寻找主要的状态、确定状态之间的转换，细化状态内的活动与转换，用复合状态来展开细节。
用例图：
执行者：可能使用这些用例的人或外部程序。
用例：对系统提供的功能（或称系统的用途）的一种描述。
类图
定义：描述了系统中的类及其相互之间的各种关系，其本质反映了系统中包含的各种对 象的类型及对象间的静态关系（关联、子类型等关系）
包图：
介绍：包图也存在类图里面的继承、引用等依赖关系，也包含接口，接口与包之间用带 小圆圈的实线相连。
序列图：

介绍：序列图中的对象可以是并发执行的，每一个对象有自己运行的线程控制着。这时，需 要通过激活、异步消息、同步控制和活动对象来表示。序列图有两种，一种是描述特定对象 之间生存期中消息通信的所有情节，称作一般序列图；一种是描述消息通信的个别情节的实 例序列图，如果需要描述所有的情节，则需要多个实例序列图。
部署图
介绍：部署图描述了系统在运行时的物理结构、配置和关系，涉及处理器、设备、通讯 等硬件单元和软件部件。部署图的描述是基于代表硬件单元的节点之上的。
// 在需求原型化方面，什么是抛弃型原型？什么是演化型原型？
// 用DFD图简单描述ATM机的工作原理（主要功能和数据流）（习题7）

Chapter05

1.什么是软件体系结构？设计模式？设计公约？设计？ //概念设计？技术设计？

体系结构 Architecture:一种软件解决方案，用于解释如何将系统分解为单元， 以及单元如何相互关联，还包括这些单元的所有外部特性。
设计模式 design pattern:一种针对单个软件模块或少量模块而给出的一般性解决方案，它提供较低层次的设计决策。它是一个共同的设计结构的关键方面，包括对象和实例， 角色和协作，责任分配，
设计公约 Design Convention:一系列设计决策和建议的集合，用于提高系统某方面的设计质量。当一种设计公约发展成熟时，将会被封装成设计模式或体系结构风格，最后可 能被内嵌为一种程序语言结构
设计 Design:将需求中的问题描述转变成软件解决方案的创造性过程

2.软件设计过程模型的几个阶段？

(1) Modeling 建模:尝试可能的分解，根据需求描述的系统的关键特性等确定软件体系结构风格
(2) Analysis 分析:分析初步的体系结构，主要关注软件系统的质量属性性能、安全性、可靠性等、各种约束等等。关注系统级别决策
(3) Documentation 文档化:确定各个不同的模型视图。
(4) Review 复审:检查文档是否满足了所有需求。
(5) final output: SAD:Software Architecture Document 软件体系结构文档， 用来和开发团队中其他人员交流系统级别设计决策的有力工具。

// 三种设计层次极其关系？
//什么是模块化？什么是抽象？

3.论述设计用户界面应考虑的问题。

(1)设计界面要注意解决的要素：
①隐喻：可识别和学习的基本术语、图像和概念等
②思维模型：数据、功能、任务的组织与表示
③模型的导航规则：怎样在数据、功能、活动和角色中移动及切换
④外观：系统向用户传输信息的外观特征
⑤感觉：向用户提供有吸引力的体验的交互技术
(2) 文化问题：需要考虑使用系统的用户的信仰、价值观、道德规范、传统、风俗和传 说。两种解决方法：
①使用国际设计/无偏见设计，排除特定的文化参考或偏见②采用定制 界面，使不同用户看到额界面
(3) 用户偏爱：为具有不同偏好的人选择备选界面

4.----模块独立性----耦合与内聚的概念及各个层次划分?

老师给的模拟题问的内聚，我感觉应该考耦合，结果一个选择题三个小空问分别是哪种聚合。欲哭无泪

耦合度是指两个软件之间的相互关联程度，耦合程度取决于模块之间的依赖关系的多少，可以划分为紧密耦合、松散耦合和非耦合。模块之间的依赖关系有：一个模块引用另一个模块、模块间传递数据量、某个模块控制其他模块的数量。为了使模块可以独立设计和修改，应尽可能减少耦合度。模块耦合有六个级别，从高到底依次为：
(1) 内容耦合 content：一个模块实际上修改了另一个模块，被修改的模块完全依赖于修改他的模块。可能的情况有：一个模块修改另一个模块内部数据项或代码，或分支转移 到另一个模块。如 goto 语句
(2) 公共耦合 common：不同模块可以从公共数据存储区来访问和修改数据。
(3) 控制耦合 control：一个模块通过传递参数或返回代码来控制另一个模块的活动
(4) 标记/特征耦合 stamp：使用一个复杂的数据结构进行模块间传递消息，并且传递的是该数据结构本身。比如将一个数组传递给另一个模块，数组仅用于计算而非控制
(5) 数据耦合 data：模块间传递的是数据值，这是最受欢迎的一种耦合。如一个数值被当做参数传递给另一个模块，这个数值在另一个模块中只会参与计算而非控制。
(6) 非耦合 uncoupled：模块相互之间没有信息传递，如两个毫无关系的方法，但是一般完全没有耦合是不现实的。

内聚度是指模块内部各组成成分（如数据、功能、内部模块）的关联程度，内聚度越高， 模块各成分间相互联系越密切，与总目标越相关。内聚分为低内聚和高内聚。
(1) 偶然内聚 coincidental：模块各部分不相关，只为方便或偶然性原因放入同一模块。比如强行放入一个类中没有任何关系的方法
(2) 逻辑内聚 logical：模块中各部分只通过代码的逻辑结构相关联， 会共享程序状态和代码结构，但相对于数据、功能和目标的内聚比较弱。比如因为有相同的某一个计算步骤而放在 一起的两个没有关系的计算。
(3) 时间内聚 temporal：部件各部分要求在同一时间完成或被同一任务使用而形成联系。比如初始化模块中需要完成变量赋值、打开某文件等工作。
(4) 过程内聚 procedurally：要求必须按照某个确定的顺序执行一系列功能，模块内功能组合在一起只是为了确保这个顺序。其与时间性内聚相比优点在于其功能总是涉及相关 活动和针对相关目标，如写数据->检查数据->操作数据这一过程
(5) 通讯内聚 communicational：各部分访问和操作同一数据集，如将来自于同一传感器的所有不相干数据取出这一模块
(6) 顺序内聚 sequential：各部分有输入输出关系，操作统一数据集，并且操作有顺序
(7) 功能内聚 functional：理想情况，各部分组成单一功能，且每个处理元素对功能都是必须的，每个元素执行且只执行设计功能，如一个简单的输出程序
(8) 信息内聚 information：功能内聚的基础上调整为数据抽象化和基于对象的设计

5.举例说明耦合与内聚的基本分类。以及各个分类的含义与特征

见上

6.软件过程中复审的概念，设计复审的重要性。

复审定义：检查文档是否满足所有功能及质量需求。
(1) 验证 verification：确保设计遵循良好的设计原则，设计文档满足阅读者的需要。验证检查某样东西是否符合之前已定好的标准，就是要用数据证明我们是不是在正确的制造产品。更注重过程正确性，强调做得正确

(2)确认 validation：确认设计能够满足用户需求。确认检查软件在最终的运行环境上是否达到预期的目标，就是要用数据证明我们是不是制造了正确的产品。更注重结果正确 性，强调做的东西正确。
(3) 验证更多是从开发商角度来做评审、测试来验证产品需求、架构设计等方面是否和 用户要求一致，确认更多是从用户的角度或者可以是模拟用户角度来验证产品是否和自己想 要的一致。
重要性：
(1) 复审中批评和讨论是"忘我"的，能将开发人员更好地团结在一起，提倡并增强了成 员之间的交流
(2) 在评审过程中故障的改正还比较容易，成本还不高，在这时候发现故障和问题会使每一个人受益。

Chapter06

// 什么是面向对象？OO有几个基本特征？如何使用高级语言实现这些基本// 特征？掌握并使用高级语言的OO基本编程方法和技巧。

1.什么是设计模式?

名词解释是一套被反复使用的（多数人知晓并经过分类编目的）代码设计经验的总结，使用设计模式目的是为了可重用代码、让代码更容易被他人理解并且保证软件质量。

2.了解OO设计的基本原则?

①单一职责原则
②重用原则
③开闭原则
④替换原则
⑤依赖倒置原则
⑥接口隔离原则
⑦迪米特法则

3.了解OO开发有何优势?

1.语言的一致性：采用相同的语义结构（类、对象、接口、属性、行为）描述问题和解决方案
2.全开发过程的一致性：从需求分析和定义、高层设计、底层设计到编码和测试等，所有的过程都采用相同的语义结构

4.OO开发过程有几个步骤?

OO 需求分析和定义+OO 高层设计+OO 底层设计+OOP+OO 测试

5.掌握用例图的组成和画法，用例的几个要素的含义。 //掌握用例图的实例解析方法

用例图：表示一个用户、外部系统或其他实体和在开发系统的关系
①用例
描述系统提供的特定功能，用椭圆表示：
②执行者
和系统交互的实体（用户、设备或其他），用小人表示：
③包含
对已定义用例的复用，用以提取公共行为，用带箭头的实线表示：

④扩展
对一个用例的扩展使用，用以下图标

6.用例模型相关建模步骤是什么?

7.用例图、类图等针对面向对象的项目开发的意义是什么?

8.熟悉类图中各个类之间的基本关系分类及其含义。 //状态图的含义及用途。

继承（inheritance）
关联 （双向关联、单向关联、自关联，多重性关联）一个判断，关于多重度
聚合 聚合(Aggregation)关系中的成员对象是整体对象的一部分，但是成员对象可以脱离整体对象存在
组合 组合(Composition)关系也表示部分与整体的关系。但是组合关系中整体对象可以控制成员对象的生命周期，一旦整体对象不存在，成员对象也将不存在，两者是共生关系。
依赖。依赖(Dependency)关系是一种使用关系，大多数情况下依赖关系体现在某个类的方法 使用另一个类的对象作为参数
接口与实现
在 java 和 C#中都引入了接口，接口中通常没有属性，而且所有的操作都是抽象的。接口之间也有继承和依赖关系，但是接口设计很重要的一种关系是实现关系，即类实现了接口

9.绘制类图最常用的方法及步骤是什么?

10.熟悉用例图、类图、状态图等的组成和画法。

11了解UML其他图示结构的基本用途。

Chapter07

//为什么说编码工作是纷繁复杂甚至令人气馁?

1.一般性的编程原则应该从哪三个方面考虑?

（1）控制结构：当设计转变成代码时，我们希望保留组件的控制结构，在隐含调用的 面向对象设计中，控制是基于系统状态和变量而变化的。
（2）算法：在编写代码时，程序设计通常会制定一类算法，用于编写组件。
（3）数据结构：编写程序时，应该安排数据的格式并进行存储，这样的数据管理和操 作才能简明易懂。

//论述编码阶段实现某种算法时所涉及的问题。

2.在编写程序内部文档时，除了HCB外，还应添加什么注释信息？注意什么?

（1）头注释块（header comment block，HCB)
将一组注释信息放在每个构件的开始部分，包含构件名，作者，配置在整个系统设计的哪个部分上，何时编写和修改的，为什么要有该构件，构件是如何使用数据结构，算法和控制的。
（2）其他程序注释包含：
a. 可以对程序正在做什么提供逐行的解释。

b. 将代码分解成表示主要活动的段，每个活动再分解成更小的步骤。
c. 随着时间进行修改的记录。
（3）有意义的变量名和语句标记
命名时尽量用有意义的变量名进行命名
（4）安排格式以增强理解
注意缩进和间隔来反映基本的控制结构。

3.什么是极限编程(XP)? 以及派对编程?

极限编程是敏捷过程的一种具体形式，提供敏捷方法最一般原则的指导方针。XP 的支持者强调敏捷方法的 4 个特性：交流、简单性、勇气以及反馈。
交流是指客户与开发人员之间持续地交换看法；
简单性激励开发人员选择最简单的设计或实现来处理客户的需要；勇气体现在尽早地和经常交付功能的承诺；
在软件开发过程中的各种活动中，都包含反馈循环。例如，程序员一起工作，针对实现设计的最佳方式，相互提供反馈；客户和程序员一起工作时，以完成计划的任务。
结对编程名次解释属于主要的敏捷开发方法，开发方式是两个程序员共同开发程序，且角色分工明确：一个负责编写程序，另一个负责复审和测试，两个人定期交换角色。
优点：提高生产率和质量，但证据不充分，模棱两可
缺点：会抑制问题求解的基本步骤，扰乱对问题的关注

Chapter08

1.了解 产生软件缺陷的原因?

(1)软件本身，系统处理大量的状态，复杂的公式，活动，算法等；
(2)客户不清晰的需求；
(3)其他原因，如项目的规模，众多的参与者导致的复杂性。
// 将软件缺陷进行分类的理由?
知道正在处理的是什么类别的故障对于我们具体的测试，以及之后的故障 改正都是有很大帮助的。

2.有几种主要的缺陷类型? （故障和缺陷换个说法）

(1)算法故障(algorithmic fault)：由于处理步骤中的某些错误，使得对于给定的输入，构件的算法或逻辑没有产生适当的输出。
(2)计算故障(computation fault)或精读故障(precision fault)：一个公式的实现是错误的，或者计算结果没有达到要求的精度。
(3)文档故障(documentation fault)：文档与程序实际做的事情不一致。名次解释
(4)压力故障(stress fault)或过载故障(overload fault)：对队列长度、缓冲区大小、表的维度等的使用超出了规定的能力。
(5)能力故障(capacity fault)或边界故障(boundary fault)：系统活动到达指定的极限时，系统性能会变得不可接受。
(6)计时故障(timing fault)或协调故障(coordination fault)：几个同时执行或仔细定义顺序执行的进程之间细条不适当。
(7)吞吐量故障(throughput fault)或性能故障(performance fault)：系统不能以需求规定的速度执行。
(8)恢复故障(recovery fault)：当系统失效时，不能表现得像设计人员希望的或客户要求的那样。
(9)硬件和系统软件故障(hardware and system software fault)：当提供的硬件或者系统软件实际上并没有按

照文档中的操作条件或步骤运作时。
(10)标准和过程故障(standards and procesure fault)：代码没有遵循组织机构的标准和过程。

3.什么是正交缺陷分类?

定义：被分类的任何一项故障都只属于一个类别，则分类方案是正交的。如果一个故障 属于不止一个类，则失去了度量的意义。

4.测试的各个阶段及其任务? （图8.3）

(1)模块测试(module testing)、构件测试(component testing)或单元测试(unit testing)：将每个程序构件与系统中的其他构件隔离，对其本身进行测试。
(2)集成测试(integration testing)：验证系统构件是否能够按照系统和程序设计规格说明中描述的那样共同工作的过程。
(3)功能测试(function test)：对系统进行评估，以确定集成的系统是否确实执行了需求规格说明中描述的功能，其结果是一个可运转的系统。
(4)性能测试(performance test)：测试系统的软硬件性能是否符合需求规格说明文档。其结果是一个确认的系统。
(5)验收测试(acceptance test)：确定系统是按照用户的期望运转的。
(6)安装测试(installation test)：确保系统在实际环境中按照应有的方式运转。
(7)系统测试(system test)：功能测试、性能测试、验收测试和安装测试统称为系统测试。
// 测试的态度问题？（为什么要独立设置测试团队？）

5.掌握测试的方法----黑盒、白盒的概念?

黑盒 将测试的对象看作是一个不了解其内容的闭盒，我们的测试就是向闭盒提供输入的数据，并记录产生的输出。测试的目标是确保针对每一种输入，观察到的输出与预期的输出相匹配。

优点：黑盒测试免于受强加给测试对象内部结构和逻辑的约束。更偏向于功能性的测试。
缺点：黑盒法以 SRS 为依据，有一定的盲目性和不确定性，不可能揭示所有的错误。
没办法总是使用这种方式进行完备的测试。不容易找到具有代表性的测试用例证明所有情况 下功能都正确。
白盒：将测试对象看作一个白盒，然后根据测试对象的结构用不同的方式进行测试。

优点：可以测试一个模块的细节。
缺点：该法以模块内部逻辑为依据，当内部逻辑过于复杂时，则不能给出好的或合适的 测试用例。有时候，对于大量递归、循环和分支的构件，想要测试完所有的分支也是不现实 的。

6.什么是单元测试?

//什么是走查和检查?
这玩意考了个判断，检查比走查更严谨认真?

7.黑盒白盒方法各自的分类？测试用例的设计和给出方法。

8.黑盒白盒方法的分类，各种覆盖方法等。（课件等）

黑盒：

等价分类法：将输入域划分为若干等价类。每一个测试用例都代表了一类与它等价 的其他例子。如果测试用例没有发现错误，那么对应的等价例子也不会发生错误。有效等价 类的测试用例尽量公用，以此来减少测试次数，无效等价类必须每类一个用例，以防止漏掉 可能发现的错误。
边界值分析法：在等价分类法中，代表一个类的测试数据可以在这个类的允许范围 内任意选择。但如果把测试值选在等价类的边界上，往往有更好的效果，这就是边界值分析 法的主要思想。
错误猜测法：猜测程序中哪些地方容易出错，并据此设计测试用例。更多的依赖于 测试人员的直觉和经验。
因果图法：适用于被测试程序有很多输入条件，程序的输出又依赖输入条件的各种 组合的情况。
白盒：一道综合题

语句覆盖 + 判定(分支)覆盖一个判断 + 条件覆盖：要求判定中的每个条件均按照“真”、“假”两种结果至少执行一次。
条件组合覆盖：要求所有条件结果的组合都至少出现一次(比如 A&&B，两个条件，那么就有四种条件的组合)。

9.如何面对一个命题，设计和给出测试用例的问题。（课件）

------课堂练习的测试题目和讲解内容

10.集成测试及其主要方法的分类？（驱动模块、桩模块的概念）

构件驱动程序：在测试最底层的构件时，因为没有现成的已测试的构件调用底层的待测试构件，所以我们需要编写特定的代码来辅助集成。构件驱动程序就是调用特定构件并向其 传递测试用例的程序。（即代替上层模块的调用程序）
桩(stub)这玩意考了个名词解释f...k：一种专用程序，用于模拟测试时缺少构件时的活动。桩应答调用序列，并传 回输出数据，使测试能够正常的进行下去。（即代替下层模块的应答程序）
// 传统测试和OO测试有何不同？OO测试有何困难？

// 测试计划涉及的几个步骤？ （了解）
(1)制定测试目标
(2)设计测试计划
(3)编写测试用例(4)测试测试用例
(5)执行测试
(6)评估测试结果

课件例题：
某城市的电话号码由 3 部分组成。这 3 个部分的名称与内容分别是:
地区码：空白或 3 位数字；前缀：非‘0’或‘1’开头的 3 位数字；后缀：4 位数字。
假定被测程序能接受一切符合上述规定的电话号码，拒绝所有不符合规定的号码，请使 用等价类的思路设计测试用例。

Chapter09

1.系统测试的主要步骤及各自含义？（图9.2）

(1)功能测试——系统功能需求
(2)性能测试——其他软件需求
(3)验收测试——客户需求规格说明书
(4)安装测试——用户环境

// 什么是系统配置？软件配置管理？ // 基线？（或见课件）
// 什么是回归测试？

2.功能测试的含义极其作用？

测试需求设计的功能性需求。
有很高的故障检测概率（因为一项功能测试只面向一小组组件）。

3.功能测试的基本指导原则？

简答
(1)高故障检测概率；
(2)使用独立于设计人员和程序员的测试小组；
(3)了解期望的动作和输出；
(4)既要测试合法输入，也要测试不合法输入；
(5)制定停止测试的标准。

4.性能测试的含义与作用？

性能测试与需求的质量有密切的关系，需求文档需要足够完备才能确保性能测试的成功 进行。因此需求的质量通常可以反映在性能测试的容易度上。
性能测试所针对的是非功能需求。它需要确保这个系统的可靠性、可用性与可维护性。性能测试由测试小组进行设计和执行并将结果提供给客户。

5.性能测试的主要分类？

压力测试(短时间内加载极限负荷，验证系统能力)
容量测试(验证系统处理巨量数据的能力)
配置测试(测试各种软硬件配置(最小到最大))
兼容性测试(如果它与其他系统交互时)
回归测试(如果这个系统要替代一个现有系统时)
安全性测试
计 时 测 试
环 境 测 试
质 量 测 试
恢 复 测 试
维 护 测 试
文档测试
人为因素测试/可使用性测试
// 什么是可靠性、可用性和可维护性？

6.确认测试概念，确认测试分类？（基准测试和引导测试）

(1)基准测试
由用户准备典型测试用例，在实际安装后的系统运作并由用户对系统执行情况进行评估。
(2)引导测试(课件译)/试验性测试(课本译)
在假设系统已经永久安装的前提下执行系统。它依赖系统的日常工作进行测试，相对基准测试不是非常的正式与结构化。

7.什么是alpha测试？β测试？

α测试：在向客户发布一个系统之前，先让来自自己组织机构或公司的用户来测试这个系统。在客户进行实际的试验性测试前先试验这个系统。
β测试名词解释：客户的试验成为β测试。

8.什么是安装测试？

在用户环境中配置系统，以测试可能因为开发环境与用户环境的不同而导致的问题。

总的来说，老师给的纲要对名词解释和简答题覆盖几乎是100%，选择和判断由于其灵活性，有的并没有直接体现在纲要里。 大题不算太难。
我只背诵了这份整理。于是经历了 名解简答，谁都打不过；选择判断，谁都打不过。

复习建议：
1.按照老师纲要整理知识点，全文背诵。
2.注意对概念的理解，应对选择判断