Ch0  
2021.3.13 Software crisis(软件危机)  
https://xuewen.cnki.net/R2014096100003379.html  
The severe impact and huge loss caused by the sharp contradiction between the complexity of computer software development and actual application requirements  
 什么是软件危机   
　　软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。
　　软件危机是落后的软件生产方式无法满足迅速增长的计算机软件需求, 从而导致软件开发与维 护过程中出现一系列严重问题的现象。 这些严重的问题阻碍着软件生产的规模化、商品化以及生产效率，让软件的开发和生产成为制约软件产业发展的“瓶径”。
  
 软件危机的表现  
　　(1)对软件开发成本和进度的估计常常很不准确。这种现象降低了软件开发组织的信誉。而为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质量，从而不可避免地会引起用户的不满。
　　(2)用户对“已完成的”软件系统不满意的现象经常发生。软件开发人员和用户之间的信息交流往往很不充分，“闭门造车”必然导致最终的产品不符合用户的实际需要。
　　(3)软件质量保证技术（审查、复审和测试） 没有坚持不懈地应用到软件开发全过程中。
　　(4)软件常常是不可维护的。由于开发过程没有统一的、公认的规范，软件开发人员按各自的风格工作，各行其是。很多程序中的错误是非常难改正的，实际上不可能使这些程序适应新的硬件环境，难适应用户要求增加的新的功能需求，软件的复用性不高。
　　(5)软件通常没有适当的文档资料。计算机软件不仅仅是程序，还应该有一整套文档资料。这些文档资料应该是在软件开发过程中产生出来的，而且应该是“最新式的”（即和程序代码完全一致的）。软件通常没有适当的文档资料，文档资料的作用是：管理和评价软件开发过程的进展情况，开发者与用户和开发者之间通信的工具，维护工具。
　　(6)软件成本在计算机系统总成本中所占的比例逐年上升。由于微电子学技术的进步和生产自动化程fe的不断提高，硬件成本逐年下降，然而软件开发需要大量人力，软件成本随着通货膨胀以及软件，规模和数量的不断扩大而持续上升。1985年美国软件成本占计算机系统总成本的比例90%。
　　(7)软件开发生产率提高的速度，远远跟不上计算机应用迅速普及深入的趋势。软件产品“供不应求”的现象使人类不能充分利用现代计算机硬件提供的巨大潜力。

Ch1  
2021.3.13 Software engineering(软件工程)  
https://baike.so.com/doc/5392576-5629410.html  
It is an emerging technology that enables the design and development of computer software to have the characteristics of general engineering processing to obtain cheap, reliable, and effective software. Including structural programming, portability and adaptability design, organization and management of software work, software archiving, maintenance, etc.  
 什么是软件工程   
　　软件工程是一门研究用工程化方法构建和维护有效的、实用的和高质量的软件的学科。它涉及到程序设计语言，数据库，软件开发工具，系统平台，标准，设计模式等方面。
　　在现代社会中，软件应用于多个方面。典型的软件比如有电子邮件，嵌入式系统，人机界面，办公套件，操作系统，编译器，数据库，游戏等。同时，各个行业几乎都有计算机软件的应用，比如工业，农业，银行，航空，政府部门等。这些应用促进了经济和社会的发展，使得人们的工作更加高效，同时提高了生活质量。
　　软件工程师是对应用软件创造软件的人们的统称，软件工程师按照所处的领域不同可以分为系统分析员，软件设计师，系统架构师，程序员，测试员等等。人们也常常用程序员来泛指各种软件工程师。
  
 软件工程的框架   
　　软件工程的框架可概括为：目标、过程和原则。
　　1、软件工程目标
　　生产具有正确性、可用性以及开销合宜的产品。正确性指软件产品达到预期功能的程度。可用性指软件基本结构、实现及文档为用户可用的程度。开销合宜是指软件开发、运行的整个开销满足用户要求的程度。这些目标的实现不论在理论上还是在实践中均存在很多待解决的问题，它们形成了对过程、过程模型及工程方法选取的约束。
　　软件工程活动是“生产一个最终满足需求且达到工程目标的软件产品所需要的步骤”。主要包括需求、设计、实现、确认以及支持等活动。需求活动包括问题分析和需求分析。问题分析获取需求定义，又称软件需求规约。需求分析生成功能规约。设计活动一般包括概要设计和详细设计。概要设计建立整个软件体系结构，包括子系统、模块以及相关层次的说明、每一模块接口定义。详细设计产生程序员可用的模块说明，包括每一模块中数据结构说明及加工描述。实现活动把设计结果转换为可执行的程序代码。确认活动贯穿于整个开发过程，实现完成后的确认，保证最终产品满足用户的要求。支持活动包括修改和完善。伴随以上活动，还有管理过程、支持过程、培训过程等。
　　2、软件工程过程
　　生产一个最终能满足需求且达到工程目标的软件产品所需要的步骤。软件工程过程主要包括开发过程、运作过程、维护过程。它们覆盖了需求、设计、实现、确认以及维护等活动。需求活动包括问题分析和需求分析。问题分析获取需求定义，又称软件需求规约。需求分析生成功能规约。设计活动一般包括概要设计和详细设计。概要设计建立整个软件系统结构，包括子系统、模块以及相关层次的说明、每一模块的接口定义。详细设计产生程序员可用的模块说明，包括每一模块中数据结构说明及加工描述。实现活动把设计结果转换为可执行的程序代码。确认活动贯穿于整个开发过程，实现完成后的确认，保证最终产品满足用户的要求。维护活动包括使用过程中的扩充、修改与完善。伴随以上过程，还有管理过程、支持过程、培训过程等。
　　3、软件工程的原则
　　软件工程的原则是指围绕工程设计、工程支持以及工程管理在软件开发过程中必须遵循的原则。软件工程的原则有以下四项基本原则：
　　1）选取适宜开发范型。该原则与系统设计有关。在系统设计中，软件需求、硬件需求以及其他因素之间是相互制约、相互影响的，经常需要权衡。因此，必须认识需求定义的易变性，采用适宜的开发范型予以控制，以保证软件产品满足用户的要求。
　　2）采用合适的设计方法。在软件设计中，通常要考虑软件的模块化、抽象与信息隐蔽、局部化、一致性以及适应性等特征。合适的设计方法有助于这些特征的实现，以达到软件工程的目标。
　　3）提供高质量的工程支持。“工欲善其事，必先利其器”。在软件工程中，软件工具与环境对软件过程的支持颇为重要。软件工程项目的质量与开销直接取决于对软件工程所提供的支撑质量和效用。
　　4）重视开发过程的管理。软件工程的管理，直接影响可用资源的有效利用，生产满足目标的软件产品，提高软件组织的生产能力等问题。因此，仅当软件过程得以有效管理时，才能实现有效的软件工程。
　　这一软件工程框架告诉我们，软件工程的目标是可用性、正确性和合算性；实施一个软件工程要选取适宜的开发范型，要采用合适的设计方法，要提供高质量的工程支撑，要实行开发过程的有效管理；软件工程活动主要包括需求、设计、实现、确认和支持等活动，每一活动可根据特定的软件工程，采用合适的开发范型、设计方法、支持过程以及过程管理。根据软件工程这一框架，软件工程学科的研究内容主要包括：软件开发范型、软件开发方法、软件过程、软件工具、软件开发环境、计算机辅助软件工程(CASE) 及软件经济学等。

Ch2  
2021.3.13 Software life cycle(软件生命周期)  
https://www.baike.com/wikiid/3905607932374949615?prd=home\_search&search\_id=dnooy5buu3k0&view\_id=55uwrbybnrg000  
Any computer software has to go through a process from the beginning of the development to the obsolescence. This starts from the software development plan, through the definition of requirements, analysis, design, production, testing, operation and maintenance, etc., until it is obsolete. Software life cycle  
 什么是软件生命周期   
　　软件生命周期又称为软件生存周期或系统开发生命周期，是软件的产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，这种按时间分程的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。但随着新的面向对象的设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减少。
　　生命周期的每一个周期都有确定的任务，并产生一定规格的文档（资料），提交给下一个周期作为继续工作的依据。按照软件的生命周期，软件的开发不再只单单强调“编码”，而是概括了软件开发的全过程。软件工程要求每一周期工作的开始只能必须是建立在前一个周期结果“正确”前提上的延续；因此，每一周期都是按“活动 ── 结果 ── 审核 ── 再活动 ── 直至结果正确”循环往复进展的。
  
 软件生命周期的七个阶段   
　　1、问题的定义及规划
　　此阶段是软件开发方与需求方共同讨论，主要确定软件的开发目标及其可行性。
　　2、需求分析
　　在确定软件开发可行的情况下，对软件需要实现的各个功能进行详细分析。需求分析阶段是一个很重要的阶段，这一阶段做得好，将为整个软件开发项目的成功打下良好的基础。"唯一不变的是变化本身。"，同样需求也是在整个软件开发过程中不断变化和深入的，因此我们必须制定需求变更计划来应付这种变化，以保护整个项目的顺利进行。
　　3、软件设计
　　此阶段主要根据需求分析的结果，对整个软件系统进行设计，如系统框架设计，数据库设计等等。软件设计一般分为总体设计和详细设计。好的软件设计将为软件程序编写打下良好的基础。
　　4、程序编码
　　此阶段是将软件设计的结果转换成计算机可运行的程序代码。在程序编码中必须要制定统一，符合标准的编写规范。以保证程序的可读性，易维护性，提高程序的运行效率。
　　5、软件测试
　　在软件设计完成后要经过严密的测试，以发现软件在整个设计过程中存在的问题并加以纠正。整个测试过程主要分单元测试、组装测试以及系统测试三个阶段进行。测试的方法主要有白盒测试和黑盒测试两种。在测试过程中需要建立详细的测试计划并严格按照测试计划进行测试，以减少测试的随意性。
　　6、系統轉換
　　將作業由舊系統轉換新系統，也就是說運行系統的方法，轉換方法有平行、分批、分發、立即、試驗五種轉換方法。
　　7、运行维护
　　软件维护是软件生命周期中持续时间最长的阶段。在软件开发完成并投入使用后，由于多方面的原因，软件不能继续适应用户的要求。要延续软件的使用寿命，就必须对软件进行维护。软件的维护包括纠错性维护和改进性维护两个方面。

Ch3  
2021.3.13 Software multiplexing(软件复用)  
https://xuewen.cnki.net/R2007060580000465.html  
A technology to improve software productivity and quality, using various relevant knowledge of existing software to build new software, so as to shorten the cost of software development and maintenance. Early software reuse was mainly code-level reuse. The reused knowledge specifically refers to the program, and later expanded to include domain knowledge, development experience, design procedures, architecture, requirements, design, code, and documentation.  
什么是软件复用  
　　软件复用是指在构造新的软件系统的过程中，对已存在的软件产品(设计结构、源代码、文档等)重复使用的技术。
  
软件复用的发展历史<sup id="\_ref-0" class="reference">[1]</sup>  
　　软件复用并不是一个新概念，它的首次提出可以追溯到20世纪60年代末期。截至目前，共经历了4个标志性的研究、发展阶段。
　　1．1968--1978年：萌芽、潜伏期
　　在1968年德国Garmish举行的NATO(北大西洋公约组织)软件工程会议上，Dough Mcilroy在其论文Mass Produce Software Components中提出软件复用的概念，希望通过代码复用实现软件开发的大规模生产。Mcilroy设想软件构件可根据它们的通用性、性能、应用平台进行分类，使复杂的软件系统可以像硬件设计一样，通过标准的构件进行识别、组装，这也是类构件软件复用思想的雏形，但是在以后的10年中，软件复用研究并未取得实质进展。
　　2．1979——1983年：再发现期
　　1979年，Lanergan发表论文，对其在Rayther Missice Divison中的一项软件复用项目进行总结，使得软件复用技术重新引起人们的关注。Lanergan项目小组分析了5000个COBOL源程序，发现设计和代码中有60％的冗余，因此可标准化并被复用。在此后的几年里，其他软件工程师也通过研究发现商业、金融等系统的大部分逻辑结构和设计模式都属于编辑、维护、报表等类型的模块，可通过对这些模块重新设计和标准化而得到较高的复用率。
　　3．1983——1994年：发展期
　　1983年，Hed BiggerstuffA和Alan Petis在美国的NewPort组织了第一次有关软件复用的研讨会。随后在1984年和1987年，美国IEEE Transaction on Software Engineering和IEEE Software分别出版了有关软件复用的专辑。1991年，第一届软件复用国际研讨会(IWSR)在德国举行，之后在1993年又举行了第二次研讨会。在此期间，欧洲实施了几个有关软件复用的重点项目，如ESF(Eureka Software Factory)，其主要目标是提供软件复用的工具支持。
　　4．1994年至今：成熟期
　　1994年的软件复用国际研讨会议改称软件复用国际会议，此时软件复用技术已引起了计算机科学界的广泛重视，越来越多的人投入到这一技术的研究中。面向对象技术的崛起给软件复用技术以新的希望，出现了类库、构件等新的复用方式，微软的ActiveX是其典型代表。软件复用研究重新成为热点，被视为解决软件危机、提高软件生产效率和质量的现实可行的途径。软件的构件化开发和软件复用已经成为互联网时代软件开发的大趋势。软件复用是解决“软件危机”的重要手段之一。无论是早期的结构化软件设计方法还是后来的面向对象的软件设计方法都在朝着这个方向努力。

Ch4  
2021.3.13 Quality planning(质量策划)  
https://wiki.mbalib.com/wiki/%E8%B4%A8%E9%87%8F%E7%AD%96%E5%88%92  
Quality planning is a part of quality management, committed to formulating quality goals and stipulating the necessary operating processes and related resources to achieve its quality goals. This standard not only puts forward specific requirements for the quality policy and quality objectives, but also provides a guarantee for the quality management system planning to achieve the quality objectives and achieve the standards, and to provide a guarantee for the planning results of the processes required by the quality management system.  
质量策划的涵义  
　　GB/T 19000—ISO9000族标准提出的基本工作方法是：首先制定质量方针，根据质量方针设定质量目标，根据质量目标确定工作内容（措施）、职责和权限，然后确定程序和要求，最后才付诸实施，这一系列过程就是质量策划的过程。质量策划是质量管理的一部分，致力于制定质量目标并规定必要的运行过程和相关资源以实现质量目标。
　　质量管理是指导和控制与质量有关的活动，通常包括质量方针和质量目标的建立、质量策划、质量控制、质量保证和质量改进。显然，质量策划属于“指导”与质量有关的活动，也就是“指导”质量控制、质量保证和质量改进的活动。在质量管理中，质量策划的地位低于质量方针的建立，是设定质量目标的前提，高于质量控制、质量保证和质量改进。质量控制、质量保证和质量改进只有经过质量策划，才可能有明确的对象和目标，才可能有切实的措施和方法。因此，质量策划是质量管理诸多活动中不可或缺的中间环节，是连接质量方针（可能是“虚”的或“软”的质量管理活动）和具体的质量管理活动（常被看作是“实”的或“硬”的工作）之间的桥梁和纽带。
  
质量策划的范围  
　　可以说，任何一项质量管理活动，不论其涉及的范围大小、内容多少，都需要进行质量策划。但是，GB/T19000—2000 族标准所要求的质量策划，并不是包罗万象的，而是针对那些影响组织业绩的项目进行的。一般来说，它包括：
　　（1）有关质量管理体系的策划
　　这是一种宏观的质量策划，应由最高管理者负责进行，根据质量方针确定的方向，设定质量目标，确定质量管理体系要素，分配质量职能等。在组织尚未建立质量管理体系而需要建立时，或虽已建立却需要进行重大改进时，就需要进行这种质量策划。
　　（2）有关质量目标的策划
　　组织已建立质量管理体系虽不需要进行曲重大改变，但却需要对某一时间段（例如中长期、年度、临时性）的业绩进行控制，或者需要对某一特殊的、重大的项目、产品、合同和临时的、阶段性的任务进行控制时，就需要进行这种质量策划，以便调动各部门和员工的积极性，确保策划的质量目标得以实现。例如每年进行的综合性质量策划（策划结果是形成年度质量计划）。这种质量策划的重点在确定具体的质量目标和强化质量管理体系的某些功能，而不是对质量管理体系本身进行改造。
　　（3）有关过程的策划
　　针对具体的项目、产品、合同进行的质量策划，同样需要设定质量目标，但重点在于规定必要的过程和相关的资源。这种策划包括对产品实现全过程的策划，也包括对某一过程（例如设计和开发、采购、过程运作）的策划，还包括对具体过程（例如某一次设计评审、某一项检验验收过程）的策划。也就是说，有关过程的策划，是根据过程本身的特征（大小、范围、性质等）来进行的。
　　（4）质量改进的策划
　　质量改进虽然也可视为一种过程，但却是一种特殊的、可能脱离了企业常规的过程。因此，更应当加强质量策划。如果说有关过程的策划一旦确定，这些过程就可以按策划规定重复进行的话，那么质量改进则不同，一次策划只可能针对一次质量改进课题（项目）。这样，质量改进策划就可以有是经常进行的，而且是分层次（组织及组织内的部门、班组或个人）进行的。质量改进策划越多，说明组织越充满生机和活力。

Ch5  
2021.3.13 quality improvement(质量改进)  
https://baike.so.com/doc/5722142-5934873.html  
Quality improvement is a continuous activity aiming at the highest benefit and efficiency  
 质量改进概述   
　　为向本组织及其顾客提供增值效益，在整个组织范围内所采取的提高活动和过程的效果与效率的措施。质量改进是消除系统性的问题，对现有的质量水平在控制的基础上加以提高，使质量达到一个新水平、新高度。
　　质量改进的步骤本身是一个PDCA循环,即计划(Plan),实施(Do), 检查(Check),处置(Action)四个阶段内容如下:
计划:包括制定方针,目标,计划书,管理项目等;
实施:按计划去做,落实具体对策;
检查:实施了具体对策后,验证其效果;
处置:总结成功的经验,实施标准化,以后可以按该标准进行。对于没有解决的问题,转入下一轮PDCA循环解决,为制定下一轮改进计划提供资料。
　　质量改进是企业跨部门人员参加的突破性改进；由企业各部门内部人员对现有过程进行渐进的持续改进活动。
　　质量改进是一个过程，要按照一定的规则进行，否则会影响改进的成效，甚至会徒劳无功。质量改进的组织是分为两个层次，一是从整体的角度为改进项目调动资源，这是管理层，即质量管理委员会。二是为了具体地开展工作项目，这是实施层，即质量改进团队或称质量改进小组。
  
 质量改进的含义   
　　ISO9000：2000的定义是：质量管理的一部分致力于满足质量要求的能力。
　　质量管理活动可划为两个类型。一类是维持现有的质量，其方法是“质量控制”。另一类是改进目前的质量，其方法是主动采取措施，使质量在原有的基础上有突破性的提高，即“质量改进”。
　　美国质量管理学家朱兰在欧洲质量管理组织第30届年会上发表《总体质量规划》论文中指出：质量改进是使效果达到前所未有的水平的突破过程。由此可见，质量改进的含义应包括以下内容：
　　1．质量改进的对象
　　它包括产品（或服务）质量以及与它有关的工作质量，也就是通常所说的产品质量和工作质量两个方面。前者如电视机厂生产的电视机实物的质量，饭店的输出服务质量等；后者如企业中供应部门的工作质量，车间计划调度部门的工作质量等。因此质量改进的对象是全面质量管理中所叙述的“广义质量”概念。
　　2．质量改进的效果在于“突破”
　　朱兰认为：质量改进的最终效果是按照比原计划目标高得多的质量水平进行工作。如此工作必然得到比原来目标高得多的产品质量。质量改进与质量控制效果不一样，但两者是紧密相关的，质量控制是质量改进的前提，质量改进是质量控制的发展方向，控制意味着维持其质量水平，改进的效果则是突破或提高。可见，质量控制是面对“今夭”的要求，而质量改进是为了“明天”的需要。
　　3．质量改进是一个变革的过程
　　质量改进是一个变革和突破的过程，该过程也必然遵循PDCA循环的规律。由于时代的发展是永无止境的，为立足于时代，质量改进也必然是“永无止境”的。国外质量专家认为：永不满足则兴、裹足不前则衰。
　　此外，还要深刻理解“变革”的含义，变革就是要改变现状。改变现状就必然会遇到强大的阻力。这个阻力来自技术和文化两个方面。因此，了解并消除这些阻力，是质量改进的先决条件。
　　4．偶发性缺陷与长期性缺陷
　　在质量管理过程中，既要及时排除产品的质量缺陷，又要保证产品质量的继续提高。缺陷是质量管理的主要对象，缺陷是指不满足预期的使用要求，即指一种或多种质量特性偏离了预期的使用要求。一般情况下，质量缺陷分为偶然性质量缺陷和长期性质量缺陷两种类型。
　　（1）偶然性质量缺陷是指产品质量突然恶化所造成的缺陷
　　它是由于生产过程中系统偏差所造成的。由于偶然性质量缺陷影响生产的进展，因此需要立即采取措施使生产恢复正常。它类似产品质量的“急性病”，采取对策的方式是“救火式”，其目的仅局限于“恢复常态”。
　　（2）长期性质量缺陷是指产品质量长期处于低水平状态所造成的缺陷　　
　　它是生产过程中随机偏差综合影响所造成的。人们虽然对它有所察觉，但习以为常，缺乏采取措施的紧迫感。例如，某车间不合格品率由15%下降到4%，并长期停滞在该水平上，人们认为4%的不合格品率是天经地义之事，从而不思改进。长期性质量缺陷不易引起人们的重视，所造成的经济损失远远高于偶发性质量缺陷。长期性质量缺陷类似产品质量的“慢性病”，对其采取的对策是“质量突破”方式，其目的是“层次提高”。

Ch6  
2021.3.13 QC(质量控制)  
https://xuewen.cnki.net/R2006062680002883.html  
Use mathematical statistics to study various problems of quality control in the production process of products  
 质量控制的步骤<sup id="\_ref-0" class="reference">[1]</sup>   
　　1．制订标准。确定产品所需要的质量成本、性能、安全性、可靠性等质量标准。
　　2．评价符合标准的程度。
　　3．必要时采取措施。对影响用户满意的营销、设计、工程、生产和维修等各个因素采取措施，解决问题。
　　4．制订改进计划。制订降低成本，提高性能、安全性和可靠性标准的计划。
  
什么是质量控制  
　　质量控制是指为达到质量要求所采取的作业技术和活动。这就是说，质量控制是为了通过监视质量形成过程，消除质量环上所有阶段引起不合格或不满意效果的因素。以达到质量要求，获取经济效益，而采用的各种质量作业技术和活动。在企业领域，质量控制活动主要是企业内部的生产现场管理，它与有否合同无关，是指为达到和保持质量而进行控制的技术措施和管理措施方面的活动。质量检验从属于质量控制，是质量控制的重要活动。

Ch7  
2021.3.13 Quality assurance(质量保证)  
https://baike.baidu.com/item/%E8%B4%A8%E9%87%8F%E4%BF%9D%E8%AF%81  
Quality assurance is a planned and systematic activity that provides sufficient confidence that a component or product meets certain technical requirements.  
质量保证  
　　为使人民确信某实体能满足质量要求，而在质量体系中实施并根据需要进行证实的全部有计划、有系统的活动，称为质量保证。显然，质量保证一般适用于有合同的场合，其主要目的是使用户确信产品或服务能满足规定的质量要求。如果给定的质量要求不能完全反映用户的需要，则质量保证也不可能完善。
　　质量保证分为内部质量保证和外部质量保证，内部质量保证是企业管理的一种手段，目的是为了取得企业领导的信任。外部质量保证是在合同环境中，供方取信于需方信任的一种手段。因次，质量保证的内容绝非是单纯的保证质量，而更重要的是要通过对那些影响质量的质量体系要素进行一系列有计划、有组织的
评价活动，为取得企业领导和需方的信任而提出充分可靠的证据。
　　当然，质量保证是以质量控制为其基础的，没有质量控制，就谈不上质量保证。有时，质量控制活动和质量保证活动又是相关的（见下图）。
　　　　
　 　上图中正方形表示全部质量管理工作。要开展质量管理，首先应制定质量方针，同时进行质量策划、设计并建立一个科学有效的质量体系。而要建立质量体系，则应设置质量管理组织机构，明确其职责权限，然后开展质量控制活动和内部质量保证活动。质量控制活动是作业技术和活动，而内部质量保证活动则是为了取得企业领导的信任而开展购活动。二者之间用虚S形分开，这是说明这两种活动是很难明显区分开来的，而大小虚圆则是表示方形内的活动和工作都是质量管理。如用实圆就是把它们与质量管理割开了。
　　弧形斜线部分表示外部质量保证活动，它是在合同上或法规中有质量保证要求时，才发生的。这种外部质量保证活动的开展，是为了取得需方的信任。而弧形部分覆盖在方形上，则形象地说明了外部质量保证只能建立在企业内部质量管理基础上，也就是说，质量保证体系应建立在质量管理体系基础上。离开质量管理和质量控制，就谈不上质量保证。离开质量管理体系，也就不可能建立质量保证体系。
　　通过质量控制和质量保证活动，发现质量工作中的薄弱环节和存在问题，再采取针对性的质量改进措施，进入新一轮的质量管理PDCA循环，以不断获得质量管理的成效。
  
质量保证要点<sup id="\_ref-0" class="reference">[1]</sup>  
　　(1)证实性
　　质量保证活动的关键在于能提供产品符合要求及质量管理过程符合要求的证据。无必要的证据，则谈不到信任。要把对具体产品的信任提高Nx,t组织的信任，只有使顾客对组织未来提供的新产品也同样寄予信任，才会取得更大的成功。
　　(2)预防性
　　质量保证要求对质量问题的发生应有充分的预防能力，这可以通过有效的质量策划来实现。要防患于未然，对一切可能影响产品质量的因素，预先做出周密的控制安排，确保其不失控。在实施中，还应针对发生的问题，采取相应的纠正和预防措施。
　　(3)系统性
　　不能把质量保证活动当作孤立的事件，而应从系统性的高度，从全局作出安排并加以协调控制，如诸文件间的相容性、各过程的界面和接口、过程的信息反馈、过程网络功能的发挥等。
　　(4)反应能力
　　以目前国内外各类企业的管理水平，要使问题根本不发生是难以做到的，然而质量保证的前提是满足要求，因此对任何偏离要求的现象，应能迅速作出反应，采取有效措施来加以纠正和预防。

Ch8  
2021.3.13 Software quality(软件质量)  
https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E8%B4%A8%E9%87%8F  
Software quality means that clearly stated functional and performance requirements, clearly documented development standards, and all the implicit features of software developed by professionals are met.  
 什么是软件质量   
　　软件质量是指：“对用户在功能和性能方面需求的满足、对规定的标准和规范的遵循以及正规软件某些公认的应该具有的本质”。该软件质量定义一共强调了三项：前二项是根据质量术语“明确需要而规定的；后一项是按术语“隐含需要”而提出的意即：用户的需求是软件质量评价的基础，不满足用户需求的软件是不能交付使用和走向市场的；其次为规定的标准和规范是软件开发的共同准则，不遵循这些标准和规范，就可能导致软件开发的无序和软件质量的低下；再次是对软件的某些要求虽未明确提出，但却是大家公认的，也应得到满足

Ch9  
2021.3.13 Formal technology review(正式技术复审)  
https://www.baike.com/wiki/ftr?view\_id=bf0u8p4i03k00  
Formal technical review is a software quality assurance activity carried out by software developers. Its purpose is to find functional, logical or implementation errors in any representation of the software, verify that the reviewed software does meet the requirements, and ensure that the software meets the requirements. Pre-defined standards enable software to be developed in a consistent manner, making projects easier to manage.

Ch10  
2021.3.26 Extreme Programming(极限编程)  
https://www.baike.com/wikiid/4029078207730642347?prd=home\_search&search\_id=1g9rsk3yoq9s00&view\_id=3e33j1ta15o000  
ExtremeProgramming, referred to as XP) By Kentbeck proposed in 1996, the limit becomes a software engineering methodology, which is one of the most effective ways in agile software development. Like other agile methods, the essence of extreme programming and traditional methodology is that it is more emphasizable rather than predictability. Extremely programming supporters believe that the continuous change in software demand is a natural phenomenon, which is inevitable in software project development, but also phenomena that should be accepted; they believe that all the needs of the project starting phase Compared with the method of controlling changes, it is possible to adapt changes in any stage of the project cycle, which will be more realistic and more effective.  
什么是极限编程  
　　极限编程是由Kent Beck在1996年提出的。Kent仔细地观察和分析了各种简化软件开发的前提条件、可能性以及面临的困难。1996年3月，Kent提出了新的软件开发观念——XP。
　　XP是一种轻量级的、灵巧的软件开发方法。同时，该方法具有严谨和周密的特征。XP的基础和价值观是交流、朴素、反馈和勇气，即任何一个软件项目都可以从四个方面入手进行改善：加强交流、从简单做起、寻求反馈和勇于实事求是。XP是一种近似螺旋式的开发方法，它将复杂的开发过程分解为一个个相对比较简单的小周期，通过积极的交流、反馈以及其他一系列的方法，开发人员和客户可以非常清楚开发进度、变化、待解决的问题和潜在的困难等，并根据实际情况及时地调整开发过程。
　　“Extreme(极限)”是指，对比传统的项目开发方式，XP强调把它列出的每个方法和思想做到极限、做到最好。XP所不提倡的，则一概忽略(如开发前期的整体设计等)。一个严格实施XP的项目，其开发过程应该是平稳的、高效的和快速的，能够做到一周40小时工作制而不拖延项目进度。
  
极限编程的要求  
　　极限编程要求有极限的工作环境、极限的需求、极限的设计、极限的编程和极限的测试。
　　1．极限的工作环境
　　为了在软件开发过程中最大程度地实现和满足客户和开发人员的基本权利和义务，XP要求把工作环境也做得最好。每个参加项目开发的人都将担任～个角色(项目经理、项目监督人等)并履行相应的权利和义务。所有的人都在同一个开放的开发环境中工作，每周40小时，不提倡加班。
　　2．极限的需求
　　客户应该是项目开发队伍中的一员，而不是和开发人员分开的。因为从项目的计划到最后验收整个过程客户一直起着很重要的作用。开发人员和客户一起，把各种需求分割为一个个小的需求模块，这些模块又会根据实际情况被组合在一起或者被再次分解成更小的模块。上述需求模块都被记录在一些小卡片(Story Card)上，之后将这些卡片分别分配给程序员们，并在一段时间内(通常不超过3个星期)实现。客户根据每个模块的商业价值进行排序，确定开发的优先级。开发人员要做的是确定每个需求模块的开发风险。风险高的(通常是因为缺乏类似的经验)需求模块将被优先研究、探索和开发。经过开发人员和客户分别从不同的角度评估每个模块后，它们被安排在不同的开发周期里，客户将得到一个尽可能准确的开发计划。
　　3．极限设计
　　从具体开发过程的角度来看，XP内部的过程是多个基于测试驱动的开发(Test Driven Development)周期。诸如计划和设计等外层的过程都是围绕这些测试展开的，每个开发周期都有很多相应的单元测试(Unit Test)。通过这种方式，客户和开发人员都很容易检验所开发的软件原型是否满足了用户的需求。XP提倡简单的设计(Simple Design)，即针对每个简单的需求用最简单的方式进行设计和后续的编程工作。这样写出来的程序可以通过所有相关的单元测试。
　　XP强调抛弃那种一揽子详细设计方式(Big Design Up Front)，因为在这种设计中有很多内容是现在或近期所不需要的。
　　XP还大力提倡设计复核(Review)、代码复核、重整和优化(Re{ectory)。所有的这些过程的目标，归根到底还是对设计的优化。在这些过程中不断运行单元测试和功能测试，可以保证经过优化后的系统仍然符合用户的需求。
　　4．极限编程
　　编程是程序员使用某种程序设计语言编写程序代码，并最终得到能够解决某个问题的程序的过程。XP极其重视编程，提倡配对编程(Pair Programming)，即两个人一起写同一段程序，而且代码所有权是归于整个开发队伍(Collective Code Ownership)。程序员在写程序和优化程序的时候，都要严格遵守编程规范。任何人都可以修改其他人写的程序，修改后要确定新程序能通过单元测试。
　　5．极限测试
　　测试在XP中是很重要的。XP提倡开发人员经常把开发好的模块整合到一起(Continuous Integration)，并且在每次整合后都进行单元测试。对代码进行的任何复核和修改，也都要进行单元测试。发现了错误，就要增加相应的测试，因此XP方法不需要错误数据库。

Ch11  
2021.3.26 Manifesto for Agile Software Development(敏捷软件开发宣言)  
https://www.baike.com/wikiid/8957355073873942155?prd=result\_list&view\_id=43tdktun0k8000  
AgilesoftWareDevelopmentasd established an agile Software Development Alliance in February 2001, the Snowbird City, USA, and issued a declaration of 17 people and proposed agile software development methods. Agility is dynamic to adapt to specific situations, causing changes in changing and self-improvement. Not only is to improve efficiency, reduce costs, but also to improve competitiveness. The focus exchanges agile software development methodology, identifies four perspectives: correctly handling the specifications of the document and achieves the relationship of the operating software design through good interaction; adopts full necessary principles to the document, pay attention to the development of all schedule of the development of the life period. Information; strengthen users and developers to communicate and cooperate, respond to demand variations, and achieve agile development; both to formulate plans, but also have a large number of response mechanisms.  
 什么是敏捷宣言   
　　敏捷宣言，也叫做敏捷软件开发宣言，正式宣布了对四种核心价值和十二条原则，可以指导迭代的以人为中心的软件开发方法。

Ch12  
2021.3.26 Data Dictionary(数据字典)  
https://baike.so.com/doc/2241827-2371998.html  
The data dictionary is a definition and description of data items, data structures, data streams, data storage, processing logic, etc. of data. The purpose is to make a detailed description of each element in the data flow diagram, using data dictionary as simple Modeling project. Briefly, the data dictionary is a collection of information describing data, which is a collection of definitions of all data elements used in the system.  
 什么是数据字典   
　　数据字典是指对数据的数据项、数据结构、数据流、数据存储、处理逻辑、外部实体等进行定义和描述，其目的是对数据流程图中的各个元素做出详细的说明。
  
 数据字典的内容  
　　数据字典是一种用户可以访问的记录数据库和应用程序源数据的目录。主动数据字典是指在对数据库或应用程序结构进行修改时，其内容可以由DBMS自动更新的数据字典。被动数据字典是指修改时必须手工更新其内容的数据字典。数据字典是用规范化的,无二义性的语言表达数据流程图的各组成部分,是对数据流程图各个组成部分的详细数据说明,也是表达新系统逻辑模型的主要工具之一。
　　数据字典最重要的作用是作为分析阶段的工具。任何字典最重要的用途都是供人查询对不了解的条目的解释，在结构化分析中，数据字典的作用是给数据流图上每个成分加以定义和说明。换句话说，数据流图上所有的成分的定义和解释的文字集合就是数据字典，而且在数据字典中建立的一组严密一致的定义很有助于改进分析员和用户的通信。数据库数据字典不仅是每个数据库的中心，而且对每个用户也是非常重要的信息。用户可以用SQL语句访问数据库数据字典，如数据项、数据结构、数据流、数据存储、处理过程。
数据字典是数据库的重要组成部分。它存放有数据库所用的有关信息，对用户来说是一组只读的表。数据字典内容包括：
数据库中所有模式对象的信息，如表、视图、簇、及索引等。
分配多少空间，当前使用了多少空间等。
列的缺省值。
约束信息的完整性。
用户的名字。
用户及角色被授予的权限。
用户访问或使用的审计信息。
其它产生的数据库信息。
　　数据库数据字典是一组表和视图结构。它们存放在SYSTEM表空间中。数据库数据字典不仅是每个数据库的中心。而且对每个用户也是非常重要的信息。用户可以用SQL语句访问数据库数据字典。关于数据的信息集合,是一种用户可以访问的记录数据库和应用程序元数据的目录,是对数据库内表信息的物理与逻辑的说明。

Ch13  
2021.3.26 data item(数据项)  
https://wiki.mbalib.com/wiki/%E6%95%B0%E6%8D%AE%E9%A1%B9  
The data item is the minimum unit discussed in the data structure, which is the most basic, unsatisfactory-name data unit in the data record. The data item can be a combination of letters, numbers, or both. The data type (logic, numerical, character, etc.) and data length are described. Data items are used to describe some of the properties of the entity. In a geographic information system, data item description object is a geographic entity, and its properties are divided into geometric properties (geographic locations and shapes) and non-geometric properties, including scalar properties (various measurements) and name properties (place names, The object name), etc. Data elements can be a collection of data items.  
 什么是数据项   
　　数据项是数据结构中讨论的最小单位，是数据记录中最基本的、不可分的有名数据单位。
　　数据项可以是字母、数字或两者的组合。通过数据类型（逻辑的、数值的、字符的等）及数据长度来描述。数据项用来描述实体的某种属性。在地理信息系统中，数据项描述对象是地理实体各要素及其属性，分为几何属性（地理位置和形状）和非几何属性，包括标量属性（各种量测值）和名称属性（地名及地物名）等。
　　数据元素可以是数据项的集合。

Ch14  
2021.3.26 data structure(数据机构)  
https://www.cnblogs.com/jingcaijueyan/p/9456072.html  
The data structure is the logical structure and physical structure of the research data and their interrelationships, and define this structure, and ensure that the new structure obtained after these operations is still the original structure type.

Ch15  
2021.3.26 data flow(数据流)  
https://baike.so.com/doc/6148990-6362177.html  
Data stream is initially the concept of communication domain, representing a digital encoded signal sequence of information used in the transfer. However, the data streams we mentioned are different from this. This concept was originally presented in 1998 by Henzinger in Document 87, and he defined data stream as "a sequence of data that can only be read in advance."  
数据流的分类<sup id="\_ref-1" class="reference">[2]</sup>  
　　数据的性质、格式不同，则对流的处理方法也不同，因此，在Java的输入／输出类库中，有不同的流类来对应不同性质的输入／输出流。在java．io包中，基本输入／输出流类可按其读写数据的类型之不同分为两种：字节流和字符流。
　　(1)输入流与输出流
　　数据流分为输入流(InputStream)和输出流(OutputStream)两类。输入流只能读不能写，而输出流只能写不能读。通常程序中使用输入流读出数据，输出流写入数据，就好像数据流入到程序并从程序中流出。采用数据流使程序的输入输出操作独立与相关设备。
　　输入流可从键盘或文件中获得数据，输出流可向显示器、打印机或文件中传输数据。
　　(2)缓冲流
　　为了提高数据的传输效率，通常使用缓冲流(Buffered Stream)，即为一个流配有一个缓冲区(buffer)，一个缓冲区就是专门用于传输数据的内存块。当向一个缓冲流写入数据时，系统不直接发送到外部设备，而是将数据发送到缓冲区。缓冲区自动记录数据，当缓冲区满时，系统将数据全部发送到相应的设备。
　　当从一个缓冲流中读取数据时，系统实际是从缓冲区中读取数据。当缓冲区空时，系统就会从相关设备自动读取数据，并读取尽可能多的数据充满缓冲区。
  
数据流的基本操作<sup id="\_ref-2" class="reference">[3]</sup>  
　　1．生成数据流的基本过程
　　数据流由一系列的节点组成，当数据通过每个节点时，节点对它进行定义好的操作。建立数据流通常遵循以下4个步骤：
　　(1)向数据流区域增添新的节点；
　　(2)将这些节点连接到数据流中；
　　(3)设定数据节点或数据流的功能：
　　(4)运行数据流。
　　2．向数据流区域添加，删除节点
　　当向数据流区域添加新的节点时，通过下面3种方法均可实现：
　　(1)双击选项板区中待添加的节点。
　　(2)单击待添加节点，按住鼠标不放，将其拖入数据流区域内。
　　(3)先选中选项板区中待添加的节点，然后将鼠标放入数据流区域，在鼠标变为十字形时单击数据流区域的任何空白处。
　　通过上面3种方法都将发现选中的节点出现在了数据流区域内。当不再需要数据流区域内的某个节点时，可以通过以下两种方法来删除：
　　(1)单击待删除的节点，按Delete键删除。
　　(2)右击待删除的节点，在快捷菜单中选择delete命令。
　　注意，删除一个节点后，与之相连的所有连接也将一并被删除。
　　3．将节点连接到数据流中
　　上面介绍了将节点添加到数据流区域的方法，然而要使节点真正发挥作用，需要把节点连接到数据流中。以下有3种方法可将节点连接到数据流中：
　　(1)双击节点。首先选中数据流中要连接新节点的节点(起始节点)，然后双击选项板区中要添加到数据流的节点(目标节点)，这样新节点会出现在数据流区域，并自动建立从起始节点到目标节点的连接。
　　(2)通过Alt键连接。首先在数据流中选中连接起始节点，按住Alt键不放，然后用鼠标将起始节点拖曳到目标节点后放开，连接便自动生成。
　　(3)手动连接。右击待连接的起始节点，从弹出的快捷菜单中选择“连接…”命令，然后用鼠标单击目标节点，连接便自动生成。
　　需要注意的是，并不是任何两个节点之间都可以建立连接。
　　4．绕过数据流中的节点
　　当暂时不需要数据流中的某个节点时可以绕过该节点。在绕过它时，如果该节点既有输入节点又有输出节点，那么它的输入节点和输出节点便直接相连；如果该节点没有输出结点，那么绕过该节点时与这个节点相连的所有连接便被取消。
　　方法：按住Alt键不放，然后双击数据流中需要绕过的节点，如图2所示。
　　5．将节点插入已存在的连接中
　　当需要在两个已连接的节点中再插入一个节点时，可以用鼠标将连接拖到要插入的节点上，即可将节点插入到连接中，如图3所示。同时原来两个节点间的连接被删除。
　　6．删除连接
　　当某个连接不再需要时，可以通过以下两种方法将其删除：
　　(1)选择待删除的连接，右击，从弹出的快捷菜单中选择“删除连接”命令。
　　(2)选择待删除连接的节点，按F3键，删除所有连接到该节点上的连接。
　　7．数据流的执行
　　数据流构建好后要通过执行数据流数据才能从读入开始流向各个数据节点。执行数据流的方法有以下3种。
　　(1)单击菜单栏中的▶按钮，数据流区域内的所有数据流将被执行。
　　(2)先选择要输出的数据流，再单击菜单栏中的▶按钮，被选的数据流将被执行。
　　(3)选择要执行的数据流中的输出节点，右击，在弹出的快捷菜单中选择“执行”命令，执行被选中的节点。

Ch16  
2021.3.26 data storage(数据存储)  
https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AD%98%E5%82%A8  
Data storage objects include information that requires a lookup during a temporary file or processing process generated during processing. Data is recorded on a computer or an external storage medium in some format. Data storage To name, this naming should reflect the composition meaning of the information feature. The data stream reflects the data flowing in the system, represents the feature of the dynamic data; the data stores reflects the statics of data in the system, exhibiting the characteristics of static data.  
 什么是数据存储  
　　数据存储是指数据流在加工过程中产生的临时文件或加工过程中需要查找的信息。
  
 数据存储的内容  
　　数据存储是数据流在加工过程中产生的临时文件或加工过程中需要查找的信息。数据以某种格式记录在计算机内部或外部存储介质上。常用的存储介质为磁盘和磁带。数据存储组织方式因存储介质而异。在磁带上数据仅按顺序文件方式存取；在磁盘上则可按使用要求采用顺序存取或直接存取方式。数据存储方式与数据文件组织密切相关，其关键在于建立记录的逻辑与物理顺序间对应关系，确定存储地址，以提高数据存取速度。数据存储要命名，这种命名要反映信息特征的组成含义。数据流反映了系统中流动的数据，表现出动态数据的特征；数据存储反映系统中静止的数据，表现出静态数据的特征。
　　广泛应用的硬盘驱动器可以存储大量的数据，但只能在室温下使用大约10年的时间。这是因为硬盘的磁能势垒较低，因此在一段时间后，其上面储存的信息就会逐渐丢失。CD和DVD光盘、纸张、磁带、陶瓷、泥版和石头等介质的寿命也是有限的。

Ch17  
2021.3.26 demand analysis(需求分析)  
https://www.baike.com/wikiid/6707020170390709387?prd=home\_search&search\_id=1ncfpp3wro1s00&view\_id=56kt95nw2hc000  
Demand analysis is also known as software demand analysis, system requirements analysis or demand analysis engineering, etc. It is the developer's in-depth investigation and analysis, accurate understanding of the function, performance, reliability of users and projects, and non-form Demand explicit into a complete demand definition, thereby determining what the system must do.  
内容  
需求分析的内容是针对待开发软件提供完整、清晰、具体的要求，确定软件必须实现哪些任务。具体分为功能性需求、非功能性需求与设计约束三个方面。  
目标  
为了促进软件研发工作的规范化、科学化，软件领域提出了许多软件开发与说明的方法，如结构化方法、原型化法、面向对象方法等。这些方法有的很相似。在实际需求分析工作中．每一种需求分析方法都有独特的思路和表示法，基本都适用下面的需求分析的基本原则。

Ch18  
2021.3.26 Entity contact(实体联系图)  
https://baike.so.com/doc/330693-350219.html  
E-R Picture is an entity - contact map, provides a method of representing physical, attributes, and contacts to describe the conceptual model of the real world.  
什么是E-R图  
　　E-R图即实体-联系图(Entity Relationship Diagram)，是指提供了表示实体型、属性和联系的方法，用来描述现实世界的概念模型。E-R方法:是“实体-联系方法”（Entity-Relationship Approach）的简称。它是描述现实世界概念结构模型的有效方法。
　　实体联系模型，实体关系模型或实体联系模式图(ERD)是由美籍华裔计算机科学家陈品山(Peter Chen)发明，是概念数据模型的高层描述所使用的数据模型或模式图，它为表述这种实体联系模式图形式的数据模型提供了图形符号。这种数据模型典型的用在信息系统设计的第一阶段；比如它们在需求分析阶段用来描述信息需求和/或要存储在数据库中的信息的类型。但是数据建模技术可以用来描述特定论域(就是感兴趣的区域)的任何本体(就是对使用的术语和它们的联系的概述和分类)。在基于数据库的信息系统设计的情况下，在后面的阶段(通常叫做逻辑设计)，概念模型要映射到逻辑模型如关系模型上；它依次要在物理设计期间映射到物理模型上。注意，有时这两个阶段被一起称为"物理设计"。

Ch19  
2021.3.26 state transition graph(状态转换图)  
https://www.w3cschool.cn/uml\_tutorial/uml\_tutorial-jnf328y2.html  
State conversion diagram (referred to as a state diagram) represents the behavior of the system by depicting the status of the system and the event that causes system state transition. In addition, the state diagram also indicates which movements will be done as a result system as a specific event.

Ch20  
2021.4.9 Consolidation(内聚)  
https://cloud.tencent.com/developer/article/1437781  
The stronger of the module, the stronger the correction, the higher the internal gather, and the module is stronger. A module should complete a feature as soon as possible. If there are various scenes that need to be introduced into the current module, the code quality will become very fragile, which is recommended to split into multiple modules. The low-poly module code, whether it is maintenance, expansion or reconstruction is quite troublesome, it is difficult to start.  
内聚强度类型  
内聚按强度从低到高有以下几种类型：  
内聚的概念  
内聚的概念是Constantine、Yourdon、Stevens等人提出的。按他们的观点，把内聚按紧密程度从低到高排列次序为偶然内聚、逻辑内聚、时间内聚、过程内聚、通信内聚、顺序内聚、功能内聚。但是紧密程度的增长是非线性的。偶然内聚和逻辑内聚的模块联系松散，后面几种内聚相差不多，功能内聚一个功能、独立性强、内部结构紧密，是最理想的内聚。

Ch21  
2021.4.9 coupling(耦合)  
https://www.cnblogs.com/xuxinstyle/p/9128872.html  
There is dependence between modules, resulting in changes in the change, the closer the relationship, the stronger the coupling, the worse the independence of the module. For example, the module A directly operates data in module B, it is considered strongly coupled, and if a interacts with the module B, it is considered weakly coupled. Independent modules are easy to expand, maintain, write unit testing, if the module is relied, it will greatly reduce the development efficiency.  
简介  
软件工程中  
耦合效率

Ch22  
2021.4.9 Module(模块)  
https://www.zhihu.com/question/21552857  
The module is logically decomposes the system into a thinner part, divided into and cured, and the complex problem dismantling is a number of simple problems, and solved one by one. Coupling mainly describes the relationship between modules, and consolidates mainly describes the inside of the module. The particle size of the module can be small, which can be functions, classes, function blocks, and the like.

Ch23  
2021.4.9 Abstraction(抽象)  
https://www.baike.com/wikiid/3831055210011202329?prd=result\_list&view\_id=3zdpbra6daw000  
Simplify complex reality

Ch24  
2021.4.9 hierarchy(层次结构)  
https://support.microsoft.com/zh-cn/office/%E5%88%9B%E5%BB%BA%E5%B1%82%E6%AC%A1%E7%BB%93%E6%9E%84-45b915d6-eef0-4722-a7ac-b42f1ffe7c3c#:~:text=%E5%B1%82%E6%AC%A1%E7%BB%93%E6%9E%84%E4%BB%A5%E5%9B%BE%E5%BD%A2%E6%96%B9%E5%BC%8F%E8%A1%A8%E7%A4%BA%E7%B3%BB%E7%BB%9F%E4%B8%AD%E4%B8%80%E7%B3%BB%E5%88%97%E7%94%A8%E6%88%B7%E6%88%96%E9%A1%B9%E7%9B%AE%E7%9A%84%E6%9C%89%E5%BA%8F%E5%88%86%E7%BB%84%E3%80%82%20%E9%80%9A%E8%BF%87%E5%9C%A8%20Excel%E3%80%81Outlook%E3%80%81PowerPoint%20%E6%88%96,Word%20%E4%B8%AD%E4%BD%BF%E7%94%A8%20SmartArt%20%E5%9B%BE%E5%BD%A2%EF%BC%8C%E4%BD%A0%E5%8F%AF%E4%BB%A5%E5%88%9B%E5%BB%BA%E5%B1%82%E6%AC%A1%E7%BB%93%E6%9E%84%EF%BC%8C%E5%B9%B6%E5%B0%86%E5%85%B6%E5%8C%85%E5%90%AB%E5%9C%A8%E5%B7%A5%E4%BD%9C%E8%A1%A8%E3%80%81%E7%94%B5%E5%AD%90%E9%82%AE%E4%BB%B6%E3%80%81%E6%BC%94%E7%A4%BA%E6%96%87%E7%A8%BF%E6%88%96%E6%96%87%E6%A1%A3%E4%B8%AD%E3%80%82  
It is an organizational structure of an operating system that provides a model for isolating each layer of the operating system. It includes attribute-attribut, hierarchical database, etc.  
简介  
它是根据信息的类型、级别、优先级等划分并按照特定的规则排列的一群硬件或软件项目。  
详情  
这种结构的最大特点就是将一个大型复杂的系统分解成若干单向依赖的层次，即每一层都提供一组功能且这些功能只依赖该层以内的的各层。其最内部的一层为系统核，具有初级中断处理、外部设备驱动、在进程之间切换处理机以及实施进程控制和通信的功能，其目的为提供一种进程可以存在和活动的环境。系统核以外依次为储存管理层、I/O处理层，文件存取层、作业调度层和资源分配层。他们具有各种资源管理功能并为用户提供各种服务。分层的组织结构在一些操作系统中只是作为一种指导性原则，因为如何划分操作系统的功能以及如何确定各层的内容和调用顺序都十分困难。分层操作系统的经典案例是Dijkstra的THE系统该系统的设计目标是实现一个可证明正确性的操作系统，其层次如表格所示。分层方法提供了一个隔离操作系统各层功能的模型。对现在操作系统而言，分层结构的限制过于严格，几乎没有一种操作系统是用这种方法构造的。然而，在设计操作系统时，分层的思想方法是值得借鉴和参考的。第五层 用户程序第四层 输入/输出管理第三层 操作员控制台第二层 储存管理第一层 CPU调度与信号第零层 硬件设施表：THE操作系统的分层体系

Ch25  
2021.4.9 Single job principle(单一职责原则)  
https://www.jianshu.com/p/526a70f24ac5  
A class is only responsible for the corresponding responsibilities in a functional area  
定义  
如果一个类承担的职责过多，就等于把这些职责耦合在一起了。一个职责的变化可能会削弱或者抑制这个类完成其他职责的能力。这种耦合会导致脆弱的设计，当发生变化时，设计会遭受到意想不到的破坏。而如果想要避免这种现象的发生，就要尽可能的遵守单一职责原则。此原则的核心就是解耦和增强内聚性。  
相关知识

Ch26  
2021.4.9 Open and closing principle(开闭原则)  
https://www.cnblogs.com/az4215/p/11489712.html  
A software entity should be open to the extension, close to modification  
梅耶开闭原则  
伯特兰·迈耶一般被认为是最早提出开闭原则这一术语的人，在他1988年发行的《面向对象软件构造》中给出。这一想法认为一旦完成，一个类的实现只应该因错误而修改，新的或者改变的特性应该通过新建不同的类实现。新建的类可以通过继承的方式来重用原类的代码。衍生的子类可以或不可以拥有和原类相同的接口。  
简介  
在面向对象编程领域中，开闭原则规定“软件中的对象（类，模块，函数等等）应该对于扩展是开放的，但是对于修改是封闭的”，这意味着一个实体是允许在不改变它的源代码的前提下变更它的行为。该特性在产品化的环境中是特别有价值的，在这种环境中，改变源代码需要代码审查，单元测试以及诸如此类的用以确保产品使用质量的过程。遵循这种原则的代码在扩展时并不发生改变，因此无需上述的过程。

Ch27  
2021.4.9 Rivien replacement principle(里氏替换原则)  
http://c.biancheng.net/view/1324.html  
All references to the base class (parent class) must be able to transparently use objects of their subclasses  
SOLID (面向对象设计)  
SOLID- “SOLID”中的 L 指代了里氏替换原则  
简介  
在面向对象的程序设计中，里氏替换原则（Liskov Substitution principle）是对子类型的特别定义。它由芭芭拉·利斯科夫（Barbara Liskov）在1987年在一次会议上名为“数据的抽象与层次”的演说中首先提出。

Ch28  
2021.4.9 Relying on the principle of reverse(依赖倒置原则)  
https://baike.so.com/doc/1623968-1716841.html  
Abstract should not depend on the details, the details should depend on abstraction  
代码实现  
这儿有一个“AutoSystem”类，它包含一个“ICar”接口。这个“AutoSystem”类根本不依赖于“FordCar”和“HondaCar”。所以，依赖关系被“倒置”了：“AutoSystem”模块依赖于抽象，那些具体的汽车操作也依赖于相同的抽象。于是可以添加ICar  
意图  
面向过程的开发，上层调用下层，上层依赖于下层，当下层剧烈变动时上层也要跟着变动，这就会导致模块的复用性降低而且大大提高了开发的成本。面向对象的开发很好的解决了这个问题，一般情况下抽象的变化概率很小，让用户程序依赖于抽象，实现的细节也依赖于抽象。即使实现细节不断变动，只要抽象不变，客户程序就不需要变化。这大大降低了客户程序与实现细节的耦合度。面向过程思想的结构图：

Ch29  
2021.4.9 Interface isolation principle(接口隔离原则)  
https://cloud.tencent.com/developer/article/1662926  
The client should not rely on the interface that it doesn't need.  
举例说明  
使用场合,提供调用者需要的方法,屏蔽不需要的方法.满足接口隔离原则.比如说电子商务的系统,有订单这个类,有三个地方会使用到,  
简介  
使用多个专门的接口比使用单一的总接口要好。

Ch30  
2021.4.9 Dimit Block(迪米特法则)  
https://blog.csdn.net/fanyun\_01/article/details/51956076  
A software entity should interact with other entities as little as possible, such as appearance mode, and uniform interfaces.  
来源历史  
1987年秋天由美国Northeastern University的Ian Holland提出，被UML的创始者之一Booch等普及。后来，因为在经典著作《 The Pragmatic Programmer》而广为人知。  
模式与意义  
迪米特法则可以简单说成：talk only to your immediate friends。 对于OOD来说，又被解释为下面几种方式：一个软件实体应当尽可能少的与其他实体发生相互作用。每一个软件单位对其他的单位都只有最少的知识，而且局限于那些与本单位密切相关的软件单位。迪米特法则的初衷在于降低类之间的耦合。由于每个类尽量减少对其他类的依赖，因此，很容易使得系统的功能模块功能独立，相互之间不存在（或很少有）依赖关系。迪米特法则不希望类之间建立直接的联系。如果真的有需要建立联系，也希望能通过它的友元类来转达。因此，应用迪米特法则有可能造成的一个后果就是：系统中存在大量的中介类，这些类之所以存在完全是为了传递类之间的相互调用关系——这在一定程度上增加了系统的复杂度。有兴趣可以研究一下设计模式的门面模式（Facade）和中介模式（Mediator），都是迪米特法则应用的例子。值得一提的是，虽然Ian Holland对计算机科学的贡献也仅限于这一条法则，其他方面的建树不多，但是，这一法则却不仅仅局限于计算机领域，在其他领域也同样适用。比如，美国人就在航天系统的设计中采用这一法则。

Ch31  
2021.4.9 SimpleFactoryPattern(简单工厂模式)  
https://www.cnblogs.com/shanzhiming/p/12616423.html  
Define a factory class, create and return different classes, depending on the parameters. These classes have a public parent class or an interface  
优缺点  
优点  
基本简介  
实现方式（附图）

Ch32  
2021.4.9 FactoryMethod(工厂方法模式)  
https://www.jianshu.com/p/d0c444275827  
Factory method model, also known as factory model, polymorphization plant mode and virtual constructor mode, by defining factory parent class, responsible for defining the public interface of the creation of objects, and subclasses are responsible for generating specific objects.  
模式简介  
工厂方法(Factory Method)模式的意义是定义一个创建产品对象的工厂接口，将实际创建工作推迟到子类当中。核心工厂类不再负责产品的创建，这样核心类成为一个抽象工厂角色，仅负责具体工厂子类必须实现的接口，这样进一步抽象化的好处是使得工厂方法模式可以使系统在不修改具体工厂角色的情况下引进新的产品。  
角色结构  
工厂方法经常用在以下两种情况中:

Ch33  
2021.4.9 AbstractFactory(抽象工厂模式)  
https://www.runoob.com/design-pattern/abstract-factory-pattern.html  
Abstract Factory Pattern is to create other factories around a super factory. The super factory is also known as a factory in other factories. This type of design pattern belongs to the creation mode, which provides an optimal way to create an object.In abstract plant mode, the interface is a factory responsible for creating a related object, and does not need to explicitly specify their classes. Each generated factory can provide an object in plant mode.  
实现原理  
我们的产品结构是这样的  
适用情况

Ch34  
2021.4.9 BuilderPattern(建造者模式)  
https://www.sohu.com/a/377577354\_120579833  
Hide the construction process & detail of the creation of the object, allows the user to create complex objects directly without knowing the construction of the object  
建造者模式  
Builderpublic interface PersonBuilder {void buildHead();void buildBody();void buildFoot();Person buildPerson();}//具体建造工具ConcreteBuilderpublic class ManBuilder implements PersonBuilder {Person person;public ManBuilder() {person = new Person();}public void buildbody() {person.setBody("建造男人的身体");}public void buildFoot() {person.setFoot("建造男人的脚");}public void buildHead() {person.setHead("建造男人的头");}public Person buildPerson() {return person;}}//建造者Directorpublic class PersonDirector {public Person constructPerson(PersonBuilder pb) {pb.buildHead();pb.buildBody();pb.buildFoot();return pb.buildPerson();}}Productpublic class Person {private String head;private String body;private String foot;public String getHead() {return head;}public void setHead(String head) {this.head = head;}public String getBody() {return body;}public void setBody(String body) {this.body = body;}public String getFoot() {return foot;}public void setFoot(String foot) {this.foot = foot;}}public class Man extends Person {}Testpublic class Test{public static void main(String[] args) {PersonDirector pd = new PersonDirector();Person person = pd.constructPerson(new ManBuilder());System.out.println(person.getBody());System.out.println(person.getFoot());System.out.println(person.getHead());}}result建造男人的身体建造男人的脚建造男人的头  
扩展

Ch35  
2021.4.9 Singleton(单例模式)  
https://blog.csdn.net/absolute\_chen/article/details/93380566  
As the name, it is only one example, and she is responsible for creating her own object, which provides a way to access its unique object, can access directly, do not need to instantiate the object of the class. Let's take a look at what kind of implementation.  
定义  
数学与逻辑学中，singleton定义为“有且仅有一个元素的集合”。单例模式最初的定义出现于《设计模式》（艾迪生维斯理, 1994）：“保证一个类仅有一个实例，并提供一个访问它的全局访问点。”Java中单例模式定义：“一个类有且仅有一个实例，并且自行实例化向整个系统提供。”Java单例模式例子  
简介  
通常单例模式在Java语言中，有两种构建方式：懒汉式—线程不安全：最基础的实现方式，线程上下文单例，不需要共享给所有线程，也不需要加synchronize之类的锁，以提高性能。懒汉式—线程安全：加上synchronize之类保证线程安全的基础上的懒汉模式，相对性能很低，大部分时间并不需要同步

Ch36  
2021.4.9 Prototype Pattern(原形模式)  
https://www.runoob.com/design-pattern/prototype-pattern.html  
This mode is to implement a prototype interface, which is used to create a cloning of the current object. This model is adopted when the cost of directly creating an object is relatively large.

Ch37  
2021.4.9 Decorator Pattern(装饰者模式)  
https://www.cnblogs.com/of-fanruice/p/11565679.html  
The function is attached to the object on the basis of the original object. Provide more elastic alternatives than inheritance (extended original object function)

Ch38  
2021.4.9 Adapter Pattren(适配器模式)  
http://c.biancheng.net/view/1361.html  
Convert a class interface into another interface that customers want, so that those classes that are originally working together cannot work together.  
基本信息  
共有两类适配器模式：  
解释  
将一个类的接口转换成客户希望的另外一个接口。Adapter模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

Ch39  
2021.4.9 Flyweight(享元模式)  
https://www.baike.com/wikiid/2119217174466196?prd=home\_search&search\_id=5qw6vy27tbg000&view\_id=5qd59ofri20000  
Use shared techniques to effectively support a large number of fine-grained objects. It greatly reduces the number of objects that need to be created, avoiding a large number of similar classes, and improving system resources utilization  
定义  
享元模式（FlyWeight），运用共享技术有效的支持大量细粒度的对象。  
结构  
内蕴状态存储在享元内部，不会随环境的改变而有所不同，是可以共享的。

Ch40  
2021.4.24 Bridge Pattern(桥接模式)  
https://refactoring.guru/design-patterns/bridge#:~:text=The%20Bridge%20pattern%20lets%20you%20split%20the%20monolithic,and%20minimizes%20the%20risk%20of%20breaking%20existing%20code.  
The intention of bridge mode is to delift abstraction and Implementation, so that both can vary independently.  
概述  
【GOF95】在提出桥梁模式的时候指出，桥梁模式的用意是"将抽象化(Abstraction)与实现化(Implementation)脱耦，使得二者可以独立地变化"。这句话有三个关键词，也就是抽象化、实现化和脱耦。  
结构  
下列各语言的代码都用于写出两个不同的圆的坐标和半径。API1.circle at 1:2 7.5API2.circle at 5:7 27.5

Ch41  
2021.4.24 Composite Pattern(组合模式)  
https://www.itdevtools.com/tech/4983.html  
This mode is used to create a structure such as a folder and a file that makes the container with the content consistency, creates a recursive mode, just like the file can be stored in the folder, and can be stored in the folder, and find it, You can get its names.  
 什么是组合模式  
　　组合模式是指将对象组合成树形结构以表示“部分-整体”的层次结构，组合模式使得用户对单个对象和组合对象的使用具有一致性。
  
 组合模式的作用  
　　组合模式的目的是：让客户端不再区分操作的是组合对象还是叶子对象，而是以一个统一的方式来操作。实现这个目标的关键之处，是设计一个抽象的组件类，让它可以代表组合对象和叶子对象。这样一来，客户端就不用区分到底是组合对象还是叶子对象了，只需要全部当成组件对象进行统一的操作就可以了。
统一元素与部分整体，简化处理代码。
将元素内部结构同处理程序解耦，从而一致的对待元素与部分整体。
实际上，组合模式在应用中其实非常广泛，像文件系统、企业结构等都可以看做是组合模式的典型应用。
Component:定义成接口或者抽象类，是元素与不部分整体的父类。
Leaf:叶子节点，定义了基本元素的特征，继承自Component父类，是整体最基本的组成部分。
Composite:部分整体或者容器，继承或实现自Component父类。内部包含Composite或者Leaf的聚合，用来保存内部的组织结构，并且针对该聚合有相应的操作方法。
[编辑] 组合模式的应用
　　1. 在对象与部分整体之间，想要通过统一的方式对其进行处理，模糊处理其差异的时候可以选用组合模式。
　　2. 当客户端忽视结构层次，无差异的看待元素与部分整体，不关心元素和部分整体之间的层次结构，想要实现对统一接口编程的时候。
　　3. 对象的变化是动态，而客户端想要一致的处理对象的时候。
　　4.组合模式意图是通过整体与局部之间的关系，通过树形结构的形式进行组织复杂对象，屏蔽对象内部的细节，对外展现统一的方式来操作对象，是我们处理更复杂对象的一个手段和方式。公司OA系统如何进行设计有关于此。
　　综上，在上述3种情况下可以考虑使用组合模式来设计系统程序。组合模式关键定义了一个抽象类或者接口，既可以代表元素又可以代表部分整体，而客户端是针对该接口进行编程。

Ch42  
2021.4.24 Façade Pattern(外观模式)  
https://www.runoob.com/design-pattern/facade-pattern.html  
Facade Pattern hides the complexity of the system and provides a client to access the interface of the system to the client. This type of design pattern belongs to the structural mode, which adds an interface to the existing system to hide the complexity of the system.  
优点  
（1）实现了子系统与客户端之间的松耦合关系。  
结构  
在以下情况下可以考虑使用外观模式：

Ch43  
2021.4.24 Proxy Pattern(代理模式)  
https://www.tutorialspoint.com/design\_pattern/proxy\_pattern.htm#:~:text=Next%20Page.%20In%20proxy%20pattern%2C%20a%20class%20represents,object%20to%20interface%20its%20functionality%20to%20outer%https://www.tutorialspoint.com/design\_pattern/proxy\_pattern.htm#:~:text=Next%20Page.%20In%20proxy%20pattern%2C%20a%20class%20represents,object%20to%20interface%20its%20functionality%20to%20outer%https://www.tutorialspoint.com/design\_pattern/proxy\_pattern.htm#:~:text=Next%20Page.%20In%20proxy%20pattern%2C%20a%20class%20represents,object%20to%20interface%20its%20functionality%20to%20outer%20world.  
In proxy pattern, a class represents functionality of another class. This type of design pattern comes under structural pattern.  
简介  
代理模式的定义：为其他对象提供一种代理以控制对这个对象的访问。在某些情况下，一个对象不适合或者不能直接引用另一个对象，而代理对象可以在客户端和目标对象之间起到中介的作用。  
组成  
抽象角色：通过接口或抽象类声明真实角色实现的业务方法。

Ch44  
2021.4.24 Chain of Responsilbility Pattern(责任链模式)  
https://www.geeksforgeeks.org/chain-responsibility-design-pattern/#:~:text=Chain%20of%20responsibility%20pattern%20is%20used%20to%20achieve,the%20next%20object%20in%20the%20chain%20or%20not.  
Chain of responsibility pattern is used to achieve loose coupling in software design where a request from the client is passed to a chain of objects to process them. Later, the object in the chain will decide themselves who will be processing the request and whether the request is required to be sent to the next object in the chain or not.  
Java代码示例  
以下的日志类(logging)例子演示了该模式。 每一个logging handler首先决定是否需要在该层做处理，然后将控制传递到下一个logging handler。程序的输出是:Writing to debug output: Entering function y. Writing to debug output: Step1 completed. Sending via e-mail: Step1 completed. Writing to debug output: An error has occurred. Sending via e-mail: An error has occurred. Writing to stderr: An error has occurred.注意：该例子不是日志类的推荐实现方式。同时，需要注意的是，通常在责任链模式的实现中，如果在某一层已经处理了这个logger，那么这个logger就不会传递下去。在我们这个例子中，消息会一直传递到最底层不管它是否已经被处理。  
简介  
责任链模式涉及到的角色如下所示：●　抽象处理者(Handler)角色：定义出一个处理请求的接口。如果需要，接口可以定义 出一个方法以设定和返回对下家的引用。这个角色通常由一个Java抽象类或者Java接口实现。Handler类的聚合关系给出了具体子类对下家的引用，抽象方法handleRequest()规范了子类处理请求的操作。

Ch45  
2021.4.24 Command Pattern(命令模式)  
https://zhuanlan.zhihu.com/p/86318110  
In object-oriented programming, the command pattern is a behavioral design pattern in which an object is used to encapsulate all information needed to perform an action or trigger an event at a later time. This information includes the method name, the object that owns the method and values for the method parameters.  
概述  
1.命令模式的本质是对命令进行封装，将发出命令的责任和执行命令的责任分割开。  
模式优点  
1.降低对象之间的耦合度。

Ch46  
2021.4.24 Interpreter Pattern(解释器模式)  
https://www.cnblogs.com/Bobby0322/p/4195555.html  
Interpreter Pattern: Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.  
定义  
《计算机科学技术名词 》第三版。

Ch47  
2021.4.24 Iterator Pattern(迭代器模式)  
https://refactoring.guru/design-patterns/iterator#:~:text=The%20main%20idea%20of%20the%20Iterator%20pattern%20is,traverse%20the%20same%20collection%20at%20the%20same%20time.  
In object-oriented software design, we often encounter a class of collection objects. The internal structure of such collection objects may have a variety of implementations, but they are attributed, nothing more than two is needed to care: one It is the data storage structure in the collection, and the other is data that traverses the interior of the collection. There is a single responsibility principle for object-oriented design principles, so we must decompose these duties as much as possible and use different classes to undertake different duties. The Iterator mode is to isolate the traversal behavior of the collection object, abstract an iterator class to be responsible, so that the internal structure that does not expose the set, but also allows external code to transparent access to data.  
意图  
提供一种方法顺序访问一个聚合对象中各个元素，而又不需暴露该对象的内部表示。  
简介  
在面向对象编程里，迭代器模式是一种设计模式，是一种最简单也最常见的设计模式。它可以让用户透过特定的接口巡访容器中的每一个元素而不用了解底层的实现。

Ch48  
2021.4.24 Mediator Pattern(中介模式)  
https://blog.csdn.net/sophiemantela/article/details/106571588  
Intermediar Mode: Encapsulate a series of object interactions, and the intermediaries make the objects need not be referenced to each other, making them rushing.

Ch49  
2021.4.24 Memento Pattern(备忘录模式)  
https://www.tutorialspoint.com/design\_pattern/memento\_pattern.htm  
Memento pattern uses three actor classes. Memento contains state of an object to be restored. Originator creates and stores states in Memento objects and Caretaker object is responsible to restore object state from Memento. We have created classes Memento, Originator and CareTaker.  
基本介绍

Ch50  
2021.4.24 Observer Pattern(观察者模式)  
https://www.baeldung.com/java-observer-pattern  
Observer is a behavioral design pattern. It specifies communication between objects: observable and observers. An observable is an object which notifies observers about the changes in its state.  
使用场景  
1、当一个抽象模型有两个方面，其中一个方面依赖于另一方面。将这二者封装在独立的对象中以使它们可以各自独立地改变和复用。2、当对一个对象的改变需要同时改变其他对象，而不知道具体有多少对象需要被改变。  
基本介绍  
观察者模式有很多实现方式，从根本上说，该模式必须包含两个角色：观察者和被观察对象。在刚才的例子中，业务数据是被观察对象，用户界面是观察者。观察者和被观察者之间存在“观察”的逻辑关联，当被观察者发生改变的时候，观察者就会观察到这样的变化，并且做出相应的响应。如果在用户界面、业务数据之间使用这样的观察过程，可以确保界面和数据之间划清界限，假定应用程序的需求发生变化，需要修改界面的表现，只需要重新构建一个用户界面，业务数据不需要发生变化。

Ch51  
2021.4.24 State Pattern(状态模式)  
https://zhuanlan.zhihu.com/p/86076272  
The state pattern is a behavioral software design pattern that allows an object to alter its behavior when its internal state changes. This pattern is close to the concept of finite-state machines. The state pattern can be interpreted as a strategy pattern, which is able to switch a strategy through invocations of methods defined in the pattern's interface.  
定义  
(源于Design Pattern)：当一个对象的内在状态改变时允许改变其行为，这个对象看起来像是改变了其类。  
状态模式  
(State Pattern)是设计模式的一种，属于行为模式。

Ch52  
2021.4.24 Strategy Pattern(策略模式)  
https://www.jianshu.com/p/0c62bf587b9c  
Define a series of algorithms to encapsulate each algorithm into a series of policy classes with common interfaces, allowing them to replace & let algorithms can change without affecting the client.  
简介  
策略模式作为一种软件设计模式，指对象有某个行为，但是在不同的场景中，该行为有不同的实现算法。比如每个人都要“交个人所得税”，但是“在美国交个人所得税”和“在中国交个人所得税”就有不同的算税方法。  
组成  
—抽象策略角色： 策略类，通常由一个接口或者抽象类实现。

Ch53  
2021.4.24 Template Method Pattern(模板方法模式)  
https://www.oodesign.com/template-method-pattern.html  
If we take a look at the dictionary definition of a template we can see that a template is a preset format, used as a starting point for a particular application so that the format does not have to be recreated each time it is used.  
用例（Java）  
  
简介  
模板方法模式多用在：

Ch54  
2021.4.24 Visitor Pattern(访问者模式)  
https://www.cnblogs.com/zhenyulu/articles/79719.html  
We are going to create a ComputerPart interface defining accept opearation.Keyboard, Mouse, Monitor and Computer are concrete classes implementing ComputerPart interface. We will define another interface ComputerPartVisitor which will define a visitor class operations. Computer uses concrete visitor to do corresponding action.  
涉及角色  
1.Visitor 抽象访问者角色，为该对象结构中具体元素角色声明一个访问操作接口。该操作接口的名字和参数标识了发送访问请求给具体访问者的具体元素角色，这样访问者就可以通过该元素角色的特定接口直接访问它。2.ConcreteVisitor.具体访问者角色，实现Visitor声明的接口。3.Element 定义一个接受访问操作(accept())，它以一个访问者(Visitor)作为参数。4.ConcreteElement 具体元素，实现了抽象元素(Element)所定义的接受操作接口。5.ObjectStructure 结构对象角色，这是使用访问者模式必备的角色。它具备以下特性：能枚举它的元素；可以提供一个高层接口以允许访问者访问它的元素；如有需要，可以设计成一个复合对象或者一个聚集（如一个列表或无序集合）。  
通用代码实现  
[java]

Ch55  
2021.4.24 Stub(桩)  
https://dictionary.cambridge.org/dictionary/english/stub  
Refers to a block for replacing a part of the function. Pile procedures can be used to simulate behavior of programs (such as a remote machine process) or a temporary replacement of the code to be developed

Ch56  
2021.4.24 Distributed computing(分布式计算)  
https://blog.csdn.net/OnlyQi/article/details/7975602#:~:text=Distributed%20computing%20is%20a%20field%20of%20computer%20science,other%20in%20order%20to%20achieve%20a%20common%20goal.  
Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of multiple autonomous computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal.  
 什么是分布式计算   
　　分布式计算是一种计算方法，和集中式计算是相对的。随着计算技术的发展，有些应用需要非常巨大的计算能力才能完成，如果采用集中式计算，需要耗费相当长的时间来完成。分布式计算将该应用分解成许多小的部分，分配给多台计算机进行处理。这样可以节约整体计算时间，大大提高计算效率。
  
 分布式计算意义格局   
　　（一）分布式计算与人类
　　由于现代人类各个课题学科繁多，涉及面广，而分类又细。而当今的每个学科似乎都需要进行大量的计算。天文学研究组织需要计算机来分析太空脉冲（pulse），星位移动；生物学家需要计算机来模拟蛋白质的折叠（protein folding）过程；药物学家想要研制克服艾滋病（AIDS）或非典（SARS）的药物；数学家想计算最大的质数和圆周率的更精确值；经济学家要用计算机分析计算在几万种因素考虑下某个企业/城市/国家的发展方向从而宏观调控。由此可见，人类未来的科学，时时刻刻离不开计算。而分布式计算（Distributed Computing），以其独特的优点——便宜、高效而越来越受到社会的关注。
　　（二）分布式计算格局
　　就目前来看，全球的各种分布式计算已有约百种，这些计算大多互无联系、独立管理、独立使用自己的一套软件。这种分布式计算互相割据的格局很不利于发展的需要。比如，某个生物学研究机构需要利用世界各地志愿者的计算机来模拟蛋白质折叠的过程，那个生物学研究机构没有分布式计算方面的专业人才，而但是社会上也并没有任何公司可以提供这样的服务，他们就不得不自己花费大量精力用于开发分布式计算的服务器、客户端。这样一来，原来可以用于研究生物的时间用在了别的地方。刚才提到的生物学研究机构就是美国斯坦福大学的PANDE小组。
　（三）　BOINC一统大局
　　为了改变这种杂乱无章的割据，美国加州大学伯克利分校（UC Berkeley）首先提出了建立BOINC的想法。BOINC的中文全称是伯克利开放式网络计算平台（Berkeley Open Infrastructure for Network Computing），他能够把许多不同的分布式计算项目联系起来统一管理。并对计算机资源进行统一分配（比方您对研究艾滋病药物和探索地外文明同时感兴趣，您就可以同时选择两个运行，并设置优先级）。对统计评分系统进行统一管理（无论你在为哪个项目工作，只要你奉献CPU时间长，就积分高）。有了这样的统一管理，的确给PANDE小组这样的科学研究机构提供了便利！
　　BOINC已经成熟，多个项目已经成功运行于BOINC平台之上，如SETI@home，LHC@home等。

Ch57  
2021.4.24 Black box test(黑河测试)  
https://www.w3cschool.cn/software\_testing/software\_testing-qna537uy.html  
The black box test is to see the test object as a black box, completely based on input and output data to determine the correctness of the test object.Test the specification of the test object to design input and output data

Ch58  
2021.4.24 Decision table(决策表)  
https://zhuanlan.zhihu.com/p/61723055  
Test objects used to test with complex logic judgment as specifications  
 什么是决策矩阵   
　　决策矩阵是风险型决策常用的分析手段之一，又称“决策表”、“益损矩阵”、“益损表”、“风险矩阵”。
  
决策矩阵的应用  
　　决策矩阵由备选方案、自然状态(及其发生的概率)益损值所组成。对决策问题的描述就集中地表现在决策矩阵上，决策分析就是以决策矩阵为基础，运用不同的分析标准与方法，从若干个可行方案中选出最优方案。

Ch59  
2021.4.24 Module independence(模块独立性)  
https://www.baike.com/wiki/%e6%a8%a1%e5%9d%97%e7%8b%ac%e7%ab%8b%e6%80%a7?view\_id=2n21dtzzxhq000  
Module independence refers to a measure of the relationship between each part and modules within the module, with a metric and coupling metric.  
度量标准  
模块的独立程度可以由两个定性标准度量，这两个标准分别称为内聚和耦合。耦合衡量不同模块彼此间互相依赖（连接）的紧密程度；内聚衡量一个模块内部各个元素彼此结合的紧密程度。  
模块独立性的重要性  
具有独立的模块的软件比较容易开发出来。这是由于能够分割功能而且接口可以简化，当许多人分工合作开发同一个软件时，这个优点尤其重要。

Ch60  
2021.4.24 White box test(白盒测试)  
https://www.w3cschool.cn/software\_testing/software\_testing-i75b3889.html  
White box test, also known as glass box testing, structural testing, unpacking test and transparent box test. It tests the internal coding and infrastructure of the software, focusing on checking predefined inputs based on the expected and desired output.  
 什么是白盒测试   
　　白盒测试是按照程序内部的结构测试程序，通过测试来检测产品内部动作是否按照设计规格说明书的规定正常进行，检验程序中的每条通路是否都能按预定要求正确工作。白盒测试一般用来分析程序的内部结构，对测试者而言是透明的，测试者可以看到被测程序源代码，并分析其内部结构。因此，白盒测试也叫结构测试或逻辑驱动测试。[1]
  
 白盒测试的功能<sup id="\_ref-a\_1" class="reference">[1]</sup>   
　　白盒测试主要是对程序模块进行如下检查：对程序模块的所有独立的执行路径至少测试一遍：对所有的逻辑判定，取“真”与取“假”的两种情况都能至少测一遍：在循环的边界和运行的界限内执行循环体；测试内部数据结构的有效性等等。

Ch61  
2021.5.8 quality(质量)  
https://corporatefinanceinstitute.com/resources/knowledge/strategy/quality-management/#:~:text=Quality%20management%20consists%20of%20four%20key%20components%2C%20which,and%20reliability%20in%20achieving%20an%20outcome.%20%E6%9B%B4%E5%A4%9A%E7%BB%93%E6%9E%9C...%20  
Quality is a collection of products or services to meet the characteristics and characteristics of explicit or implicit demand capabilities. In the contract environment, demand is clear; in other environments, implicit needs need to be identified and defined.

Ch62  
2021.5.8 Control domain(控制域)  
https://docs.oracle.com/cd/E23120\_01/html/821-2854/initialconfigofcontroldomain.html  
The module itself and all its module collections directly or indirectly from it  
域控制器  
“域”的真正含义指的是服务器控制网络上的计算机能否加入的计算机组合。一提到组合，势必需要严格的控制。所以实行严格的管理对网络安全是非常必要的。在对等网模式下，任何一台电脑只要接入网络，其他机器就都可以访问共享资源，如共享上网等。尽管对等网络上的共享文件可以加访问密码，但是非常容易被破解。在由Windows 9x构成的对等网中，数据的传输是非常不安全的。不过在“域”模式下，至少有一台服务器负责每一台联入网络的电脑和用户的验证工作，相当于一个单位的门卫一样，称为“域控制器（Domain Controller，简写为DC）”。域控制器中包含了由这个域的账户、密码、属于这个域的计算机等信息构成的数据库。当电脑联入网络时，域控制器首先要鉴别这台电脑是否是属于这个域的，用户使用的登录账号是否存在、密码是否正确。如果以上信息有一样不正确，那么域控制器就会拒绝这个用户从这台电脑登录。不能登录，用户就不能访问服务器上有权限保护的资源，他只能以对等网用户的方式访问Windows共享出来的资源，这样就在一定程度上保护了网络上的资源。要把一台电脑加入域，仅仅使它和服务器在网上邻居中能够相互“看”到是远远不够的，必须要由网络管理员进行相应的设置，把这台电脑加入到域中。这样才能实现文件的共享。  
简介  
控制器（controller）是指按照预定顺序改变主电路或控制电路的接线和改变电路中电阻值来控制电动机的启动、调速、制动和反向的主令装置。由程序计数器、指令寄存器、指令译码器、时序产生器和操作控制器组成，它是发布命令的“决策机构”，即完成协调和指挥整个计算机系统的操作。控制器分组合逻辑控制器和微程序控制器，两种控制器各有长处和短处。组合逻辑控制器设计麻烦，结构复杂，一旦设计完成，就不能再修改或扩充，但它的速度快。微程序控制器设计方便，结构简单，修改或扩充都方便，修改一条机器指令的功能，只需重编所对应的微程序；要增加一条机器指令，只需在控制存储器中增加一段微程序，但是，它是通过执行一段微程。具体对比如下：组合逻辑控制器又称硬布线控制器，由逻辑电路构成，完全靠硬件来实现指令的功能。电磁吸盘控制器：交流电压380V经变压器降压后，经过整流器整流变成110V直流后经控制装置进入吸盘此时吸盘被充磁，退磁时通入反向电压线路，控制器达到退磁功能。门禁控制器：门禁控制器工作在两种模式之下。一种是巡检模式，另一种是识别模式。在巡检模式下，控制器不断向读卡器发送查询代码，并接收读卡器的回复命令。这种模式会一直保持下去，直至读卡器感应到卡片。当读卡器感应到卡片后，读卡器对控制器的巡检命令产生不同的回复，在这个回复命令中，读卡器将读到的感应卡内码数据传送到门禁控制器，使门禁控制器进入到识别模式。在门禁控制器的识别模式下，门禁控制器分析感应卡内码，同设备内存储的卡片数据进行比对，并实施后续动作。门禁控制器完成接收数据的动作后，会发送命令回复读卡器，使读卡器恢复状态，同时，门禁控制器重新回到巡检模式。

Ch63  
2021.5.8 Fan-out(扇出)  
https://www.jianshu.com/p/a0cd74930656  
Refers to a module number of direct calling modules

Ch64  
2021.5.8 Fan in(扇入)  
https://www.jianshu.com/p/a0cd74930656  
How many superior modules do not call it directly

Ch65  
2021.5.8 Module independent(模块独立化)  
https://www.mesacc.edu/community-civic-engagement/service-learning/independent-module  
Each module completes a relatively independent specific sub function independently, and the relationship between other modules is simple.

Ch66  
2021.5.8 Structured programming design(结构化程序设计)  
https://searchsoftwarequality.techtarget.com/definition/structured-programming-modular-programming#:~:text=Structured%20programming%20encourages%20dividing%20an%20application%20program%20into,related%20logic%20designed%20to%20improve%20readability%20and%20maintainability.  
If a program code block is connected only by order, select, the three basic control structures are connected, and each code block has only one entry and exit, the program is called structured programming.  
 什么是结构化程序设计  
　　结构化程序设计是指程序的设计、编写和测试都采用一种规定的组织形式进行，这样，可使编制的程序结构清晰，易于读懂，易于调试和修改，充分显示出模块化程序设计的优点。
  
 结构化程序设计的内容  
　　结构化程序设计（structured programming）是进行以模块功能和处理过程设计为主的详细设计的基本原则。其概念最早由E.W.Dijikstra在1965年提出的，是软件发展的一个重要的里程碑。它的主要观点是采用自顶向下、逐步求精及模块化的程序设计方法；使用三种基本控制结构构造程序，任何程序都可由顺序、选择、循环三种基本控制结构构造。结构化程序设计主要强调的是程序的易读性。
　　结构化程序设计在1960年代开始发展，科拉多·伯姆及朱塞佩·贾可皮尼伯姆于1966年5月在《Communications of the ACM》期刊发表论文，说明任何一个有goto指令的程序，可以改为完全不使用goto指令的程序，后来艾兹赫尔·戴克斯特拉在1968年也提出著名的论文《GOTO陈述有害论》（Go To Statement Considered Harmful），因此结构化程序设计开始盛行，此概念理论上可以由结构化程序理论所证明，而在实务上，当时也有像ALGOL一样，有丰富控制结构的编程语言来实现结构化程序设计。在20世纪70年代初，由Boehm和Jacobi提出并证明的结构定理：即任何程序都可以由3种基本结构程序构成结构化程序，这3种结构是：顺序结构、分支（条件选择）结构和循环结构。每一个结构只有一个入口和一个出口，3种结构的任意组合和嵌套就构成了结构化的程序。程序的基本结构有四种：顺序结构，分支程序结构，循环程序结构，子程序结构。

Ch67  
2021.5.8 unit test(单元测试)  
https://www.cnblogs.com/yangcaogui/archive/2012/02/06/2337483.html  
Test each module as an independent entity, use a detailed design description to make a guide, test the important execution path to find errors of the module part, discover the encoding and detailed designs  
什么是单元测试  
　　单元测试是指对软件中的最小可测试单元进行检查和验证。对于单元测试中单元的含义，一般来说，要根据实际情况去判定其具体含义，如C语言中单元指一个函数，Java里单元指一个类，图形化的软件中可以指一个窗口或一个菜单等。总的来说，单元就是人为规定的最小的被测功能模块。单元测试是在软件开发过程中要进行的最低级别的测试活动，软件的独立单元将在与程序的其他部分相隔离的情况下进行测试。
　　在一种传统的结构化编程语言中，比如C，要进行测试的单元一般是函数或子过程。在像C++这样的面向对象的语言中， 要进行测试的基本单元是类。对Ada语言来说，开发人员可以选择是在独立的过程和函数，还是在Ada包的级别上进行单元测试。单元测试的原则同样被扩展到第四代语言（4GL)的开发中，在这里基本单元被典型地划分为一个菜单或显示界面。
　　经常与单元测试联系起来的另外一些开发活动包括代码走读（Code review)，静态分析（Static analysis)和动态分析（Dynamic analysis)。静态分析就是对软件的源代码进行研读，查找错误或收集一些度量数据，并不需要对代码进行编译和执行。动态分析就是通过观察软件运行时的动作，来提供执行跟踪，时间分析，以及测试覆盖度方面的信息。
  
单元测试的使用  
　　编写代码时，一定会反复调试保证它能够编译通过。代码通过编译，只是说明了它的语法正确，无法保证它的语义也一定正确，编写单元测试就是用来验证这段代码的行为。
　　单元测试越早越好。先编写产品函数的框架，然后编写测试函数，针对产品函数的功能编写测试用例，然后编写产品函数的代码，每写一个功能点都运行测试，随时补充测试用例。所谓先编写产品函数的框架，是指先编写函数空的实现，有返回值的直接返回一个合适值，编译通过后再编写测试代码，这时，函数名、参数表、返回类型都应该确定下来了，所编写的测试代码以后需修改的可能性比较小。
　　单元测试与其他测试不同，单元测试可看作是编码工作的一部分，应该由程序员完成，经过了单元测试的代码才是已完成的代码，提交产品代码时也要同时提交测试代码。测试部门可以作一定程度的审核。
　　在实践工作中，进行了完整计划的单元测试和编写实际的代码所花费的精力大致上是相同的。一旦完成了这些单元测试工作，很多Bug将被纠正，开发人员能够进行更高效的系统集成工作。使用AdaTEST和Cantata这样的支持工具可以使单元测试更加简单和有效。
　　相比后阶段的测试，单元测试的创建更简单，维护更容易，并且可以更方便的进行重复。从全程的费用来考虑， 相比起那些复杂且旷日持久的集成测试，或是不稳定的软件系统来说，单元测试所需的费用是很低的。

Ch68  
2021.5.8 Integration Testing(集成测试)  
https://www.edureka.co/blog/what-is-integration-testing-a-simple-guide-on-how-to-perform-integration-testing/  
As a result, assemble independent module is assembled into a subsystem or system, and a systematic technology that has been tested to discover an excuse.  
什么是集成测试  
　　集成测试是指在单元测试的基础上，将所有模块按照设计要求组装成为子系统或系统，进行测试。
　　集成测试最简单的形式是：把两个已经测试过的单元组合成一个组件，测试它们之间的接口。从这一层意义上讲，组件是指多个单元的集成聚合。在现实方案中，许多单元组合成组件，而这些组件又聚合为程序的更大部分。方法是测试片段的组合，并最终扩展成进程，将模块与其他组的模块一起测试。最后，将构成进程的所有模块一起测试。此外，如果程序由多个进程组成，应该成对测试它们，而不是同时测试所有进程。
　　集成测试测试组合单元时出现的问题。通过使用要求在组合单元前测试每个单元并确保每个单元的生存能力的测试计划，可以知道在组合单元时所发现的任何错误很可能与单元之间的接口有关。这种方法将可能发生的情况数量减少到更简单的分析级别。一个有效的集成测试有助于解决相关的软件与其它系统的兼容性和可操作性的问题。
　　集成测试是在单元测试的基础上，测试在将所有的软件单元按照概要设计规格说明的要求组装成模块、子系统或系统的过程中各部分工作是否达到或实现相应技术指标及要求的活动。也就是说，在集成测试之前，单元测试应该已经完成，集成测试中所使用的对象应该是已经经过单元测试的软件单元。这一点很重要，因为如果不经过单元测试，那么集成测试的效果将会受到很大影响，并且会大幅增加软件单元代码纠错的代价。
　　集成测试是单元测试的逻辑扩展。在现实方案中，集成是指多个单元的聚合，许多单元组合成模块，而这些模块又聚合成程序的更大部分，如分系统或系统。集成测试采用的方法是测试软件单元的组合能否正常工作，以及与其他组的模块能否集成起来工作。最后，还要测试构成系统的所有模块组合能否正常工作。集成测试所持的主要标准是《软件概要设计规格说明》，任何不符合该说明的程序模块行为都应该加以记载并上报。
　　所有的软件项目都不能摆脱系统集成这个阶段。不管采用什么开发模式，具体的开发工作总得从一个一个的软件单元做起，软件单元只有经过集成才能形成一个有机的整体。具体的集成过程可能是显性的也可能是隐性的。只要有集成，总是会出现一些常见问题，工程实践中，几乎不存在软件单元组装过程中不出任何问题的情况。从表中可以看出，集成测试需要花费的时间远远超过单元测试，直接从单元测试过渡到系统测试是极不妥当的做法。
活动输入输出参与角色和职责
制定集成测试计划设计模型设计模型集成测试用例测试过程测试设计员负责设计集成测试用例和测试过程
实施集成测试集成测试用例测试过程工作版本测试脚本（可选）测试过程（更新）测试设计员负责编制测试脚本（可选），更新测试过程。
驱动程序或稳定桩设计员负责设计驱动程序和装，实施员负责实施驱动程序和桩
执行集成测试测试脚本（可选）工作版本测试结果测试员负责执行测试并记录测试结果
评估集成测试集成测试计划测试结果测试评估摘要测试员负责会同及成员、编码员、设计员等有关人员（具体化）评估此次测试，并生成测试评估摘要。
[编辑]集成测试的目标
　　集成测试的目标是按照设计要求使用那些通过单元测试的构件来构造程序结构。单个模块具有高质量但不足以保证整个系统的质量。有许多隐蔽的失效是高质量模块间发生非预期交互而产生的。以下两种测试技术是用于集成测试：
　　1、功能性测试。使用黑盒测试技术针对被测模块的接口规格说明进行测试。
　　2、非功能性测试。对模块的性能或可靠性进行测试。
　　另外，集成测试的必要性还在于一些模块虽然能够单独地工作，但并不能保证连接起来也能正常工作。程序在某些局部反映不出来的问题，有可能在全局上会暴露出来，影响功能的实现。此外，在某些开发模式中，如迭代式开发，设计和实现是迭代进行的。在这种情况下，集成测试的意义还在于它能间接地验证概要设计是否具有可行性。
  
集成测试应考虑问题  
　　1、在把各个模块连接起来的时候，穿越模块接口的数据是否会丢失；
　　2、各个子功能组合起来，能否达到预期要求的父功能；
　　3、一个模块的功能是否会对另一个模块的功能产生不利的影响；
　　4、全局数据结构是否有问题；
　　5、是采用何种系统组装方法来进行组装测试；
　　6、组装测试过程中连接各个模块的顺序；
　　7、模块代码编制和测试进度是否与组装测试的顺序一致；
　　8、测试过程中是否需要专门的硬件设备；
　　9、单个模块的误差积累起来，是否会放大，从而达到不可接受的程度。
　　因此，单元测试后，有必要进行集成测试，发现并排除在模块连接中可能发生的上述问题，最终构成要求的软件子系统或系统。对子系统，集成测试也叫部件测试。
　　任何合理地组织集成测试，即选择什么方式把模块组装起来形成一个可运行的系统，直接影响到模块测试用例的形式、所用测试工具的类型、模块编号和测试的次序、生成测试用例和调试的费用。通常，有两种不同的组装方式：一次性组装方式和增值式组装方式。

Ch69  
2021.5.8 System test(系统测试)  
https://softwaretestingfundamentals.com/system-testing/  
It means that the software that will be integrated as the entire computer system, combined with computer hardware, peripherals, support software, data, and personnel, etc., a series of assembly tests and confirmation tests for computer systems.  
内容  
系统测试是将经过集成测试的软件，作为计算机系统的一个部分，与系统中其他部分结合起来，在实际运行环境下对计算机系统进行的一系列严格有效地测试，以发现软件潜在的问题，保证系统的正常运行。  
分类  
比较常见的、典型的系统测试包括恢复测试、安全测试、压力测试。下面对这几种测试进行一一介绍：1）恢复测试恢复测试作为一种系统测试，主要关注导致软件运行失败的各种条件，并验证其恢复过程能否正确执行。在特定情况下，系统需具备容错能力。另外，系统失效必须在规定时间段内被更正，否则将会导致严重的经济损失。2）安全测试安全测试用来验证系统内部的保护机制，以防止非法侵入。在安全测试中，测试人员扮演试图侵入系统的角色，采用各种办法试图突破防线。因此系统安全设计的准则是要想方设法使侵入系统所需的代价更加昂贵。3）压力测试压力测试是指在正常资源下使用异常的访问量、频率或数据量来执行系统。在压力测试中可执行以下测试：①如果平均中断数量是每秒一到两次，那么设计特殊的测试用例产生每秒十次中断。②输入数据量增加一个量级，确定输入功能将如何响应。③在虚拟操作系统下，产生需要最大内存量或其它资源的测试用例，或产生需要过量磁盘存储的数据。

Ch70  
2021.5.8 Alpha test(Alpha测试)  
https://zhuanlan.zhihu.com/p/361444661  
The user is performed by the user, and the developer is tested under the "guidance" of the user, the developer is responsible for recording the problem found to have problems encountered in use.  
 Alpha测试的目的   
　　Alpha测试的目的是评价软件产品的FLURPS(即功能、局域化、可使用性、可靠性、性能和支持)。尤其注重产品的界面和特色。Alpha测试可以从软件产品编码结束之时开始，或在模块(子系统)测试完成之后开始，也可以在确认测试过程中产品达到一定的稳定和可靠程度之后再开始。Alpha测试是指软件开发公司组织内部人员模拟各类用户行对即将面市软件产品（称为Alpha版本）进行测试，试图发现错误并修正。Alpha测试的关键在于尽可能逼真地模拟实际运行环境和用户对软件产品的操作并尽最大努力涵盖所有可能的用户操作方式。经过Alpha测试调整的软件产品称为Beta版本。
  
 什么是Alpha测试   
　　Alpha测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的测试。

Ch71  
2021.5.8 Beta test(Beta测试)  
https://www.softwaretestinghelp.com/beta-testing/#:~:text=What%20is%20Beta%20Testing%20%E2%80%93%20Definition.%20Beta%20Testing,use%20it%2C%20for%20over%20a%20period%20of%20time.  
The end users of the software are performed in one or more clients. Developers are usually not on the spot. Test the "real" application in the environment that the software cannot be controlled by the developer. The user records all the problems encountered during the test process and reports these questions to developers.  
 什么是Beta测试   
　　Beta测试是一种验收测试。所谓验收测试是软件产品完成了功能测试和系统测试之后，在产品发布之前所进行的软件测试活动，它是技术测试的最后一个阶段，通过了验收测试，产品就会进入发布阶段。验收测试一般根据产品规格说明书严格检查产品，逐行逐字地对照说明书上对软件产品所做出的各方面要求，确保所开发的软件产品符合用户的各项要求。
　　Beta测试由软件的最终用户们在一个或多个客房场所进行。与Alpha测试不同，开发者通常不在Beta测试的现场，因Beta测试是软件在开发者不能控制的环境中的“真实”应用。用户Beta测试过程中遇到的一切问题（真实在或想像的），并且定期把这些问题报告给开发者。接收到在Beta测试期间报告的问题之后，开发者对软件产品进行必要的修改，并准备向全体客户发布最终的软件产品。

Ch72  
2021.5.8 Regression Testing(回归测试)  
https://www.testingxperts.com/blog/regression-testing  
Refers to re-executing the already tested tests to ensure that the new changes in the software have not brought non-expected side effects.  
什么是回归测试  
　　回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。自动回归测试将大幅降低系统测试、维护升级等阶段的成本。
　　回归测试作为软件生命周期的一个组成部分，在整个软件测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。在渐进和快速迭代开发中，新版本的连续发布使回归测试进行的更加频繁，而在极端编程方法中，更是要求每天都进行若干次回归测试。因此，通过选择正确的回归测试策略来改进回归测试的效率和有效性是非常有意义的。
  
常用的选择回归测试方式  
　　在软件生命周期中，一个完全的回归测试包括每个基线测试用例，时间和成本约束可能阻碍运行这样一个测试，有时测试组不得不选择一个缩减的回归测试包来完成回归测试。
　　回归测试的价值在于它是一个能够检测到回归错误的受控实验。当测试组选择缩减的回归测试时，有可能删除了将揭示回归错误的测试用例，消除了发现回归错误的机会。然而，如果采用了代码相依性分析等安全的缩减技术，就可以决定哪些测试用例可以被删除而不会让回归测试的意图遭到破坏。
　　选择回归测试策略应该兼顾效率和有效性两个方面。常用的选择回归测试的方式包括：
　　1、再测试全部用例
　　选择基线测试用例库中的全部测试用例组成回归测试包，这是一种比较安全的方法，再测试全部用例具有最低的遗漏回归错误的风险，但测试成本最高。全部再测试几乎可以应用到任何情况下，基本上不需要进行分析和重新开发，但是，随着开发工作的进展，测试用例不断增多，重复原先所有的测试将带来很大的工作量，往往超出了我们的预算和进度。
　　2、基于风险选择测试
　　可以基于一定的风险标准来从基线测试用例库中选择回归测试包。首先运行最重要的、关键的和可疑的测试，而跳过那些非关键的、优先级别低的或者高稳定的测试用例，这些用例即便可能测试到缺陷，这些缺陷的严重性也仅有三级或四级。一般而言，测试从主要特征到次要特征。
　　3、基于操作剖面选择测试
　　如果基线测试用例库的测试用例是基于软件操作剖面开发的，测试用例的分布情况反映了系统的实际使用情况。回归测试所使用的测试用例个数可以由测试预算确定，回归测试可以优先选择那些针对最重要或最频繁使用功能的测试用例，释放和缓解最高级别的风险，有助于尽早发现那些对可靠性有最大影响的故障。这种方法可以在一个给定的预算下最有效的提高系统可靠性，但实施起来有一定的难度。
　　4、再测试修改的部分
　　当测试者对修改的局部化有足够的信心时，可以通过相依性分析识别软件的修改情况并分析修改的影响，将回归测试局限于被改变的模块和它的接口上。通常，一个回归错误一定涉及一个新的、修改的或删除的代码段。在允许的条件下，回归测试尽可能覆盖受到影响的部分。
　　再测试全部用例的策略是最安全的策略，但已经运行过许多次的回归测试不太可能揭示新的错误，而且很多时候，由于时间、人员、设备和经费的原因，不允许选择再测试全部用例的回归测试策略，此时，可以选择适当的策略进行缩减的回归测试。

Ch73  
2021.5.8 Software maintenance(软件维护)  
https://www.sciencedirect.com/topics/computer-science/software-maintenance  
After the software has been delivered, the process of modifying software in order to correct software running errors or meet new needs  
 什么是软件维护  
　　软件维护是指软件系统交付使用以后,为了改正软件运行错误,或者为了满足用户新的需求而加入新功能的修改软件的过程。
  
软件维护  
　　软件维护主要是指根据需求变化或硬件环境的变化对应用程序进行部分或全部的修改，修改时应充分利用源程序。修改后要填写《程序修改登记表》，并在《程序变更通知书》上写明新旧程序的不同之处。一般认为软件维护只和修正错误有关。不过有研究指出80%的软件维护工作是用在非纠正性的行动。
　　软件维护同时包括管理层面及技术层面。管理层面的问题包括：配合客户的优先级、人员配置及费用估计。技术层面的问题包括：对需求、系统或问题有限的理解、影响分析、测试以及可维护性的量测。软件可维护性：
　　根据国家标准《软件工程 产品质量 第1部分：质量模型》（GB/T 16260.1-2006），软件可维护性是指软件产品被修改的能力，修改包括纠正、改进或软件对环境、需求和功能规格说明变化的适应。GB/T 16260.1-2006标准还规定了可维护性的五个子特性：
　　（1）易分析性。软件产品诊断软件中的缺陷或失效原因或识别待修改部分的能力。
　　（2）易改变性。软件产品使指定的修改可以被实现的能力，实现包括编码、设计和文档的更改。如果软件由最终用户修改，那么易改变性可能会影响易操作性。
　　（3）稳定性。软件产品避免由于软件修改而造成意外结果的能力。
　　（4）易测试性。软件产品使已修改软件能被确认的能力。
　　（5）维护性的依从性。软件产品遵循与维护性相关的标准或约定的能力。

Ch74  
2021.5.8 Corrective maintenance(改正性维护)  
https://www.fiixsoftware.com/corrective-maintenance/  
The wrong test, diagnosis, positioning, error correction, and verification, modified regression test process, gradually exposed in the test phase, gradually exposed after the software is put into use, and modified the regression test process, called correction repair  
概述  
改正性维护作业的目的是改正软件中原有的错误、缺陷和不足，以完善原有的软件。它从接到错误申报开始。首先，要收集和占有与错误申报有关的信息或数据，然后据此判断错误申报是否应被接受。由于可能会有不正确的错误申报，因此改正性作业中还包括最终不导致修改软件的作业。根据资料统计，操作系统和数据库系统等基本软件的错误申报中，只有1/3~1/5与软件中的错误有关。用户提出的错误申报大多数是因文档（如操作使用手册）质量不高引起的。因此也属于软件错误范畴。判断出软件确有错误后，接着要分析错误原因，确定修改方案。在确定修改方案时，应尽可能地使修改对程序的其它部分影响最小。修改后，不仅要确认申报的错误已经改正，还要检查所做的修改是否向软件引进了新的错误。最后，还要相应地修改有关文档。  
软件改正性维护度量  
软件改正性维护度量涉及维护服务质量的许多方面。需要将维护组处理的软件系统失效和维护服务失效区别开来，后者指的是维护不能提供满足指定标准或合同需求的修复的情况。因此，软件维护度量的分类如下：

Ch75  
2021.5.8 Perfect maintenance(完善性维护)  
http://www.perfectmaintenancecleaning.com/  
In order to meet the requirements of the new function and performance of the software in the use process, the function of the original software needs to be modified or expanded.

Ch76  
2021.5.8 Adaptive maintenance(适应性维护)  
https://www.corrosionpedia.com/definition/6211/adaptive-maintenance  
It is software to adapt to external new hardware and software environments, or new changes in the data environment, and the process of modifying software

Ch77  
2021.5.8 Preventive maintenance(预防性维护)  
https://www.ibm.com/topics/what-is-preventive-maintenance  
In order to improve the future of maintainability, reliability, or to lay a better basis for future improvements, the process is modified  
 PM的发展过程   
　　PM一词最初引进我国是在1951年（昭和26年），即所谓战后。那时，从废墟上重建了军需工业，随之以此力图转向现代化工业，逐渐走向活跃的开端。
在日本通过化学工业和其它，以所谓安装工业为中心的日本能率协会的顾问活动，用报告会、出版刊物等极力宣传；与其在发生故障之后才进行维修，不如采取美国那种更为经济的在发生故障之前就进行维修的PM方式，从而开始出现预防维修的热潮。
　　其后，在1954年（昭和29年），如上所述，根据GE社的经验又开始提倡生产维修。表－1PM发展过程:
　　
　　这个生产维修实际也包含着预防维修，也就是为了提高生产的经济效果的维修。
　　防止由于过分的预防维修（定期点检，提早维修等）而有损于原先重视的经济效果，这就是特别强调提出以“赚钱PM”为目的的理由。所以不仅预防维修，就要从设备的各种不同条件来考虑，按其设备、部位不同，有时也会有必须等待发生故障以后进行修理才合乎经济效果的情况。在这种情况下，当然，就不如采取在发生故障以后进行修理的办法，也就是所谓事后维修（BreakdownMaintenance）较为妥善。
　　大约从1957年（昭和32年）起，特别强调了要采取比预防维修更进一步的措施，力求从设备本身少出故障，缩短修理时间、延长使用寿命，开展了依靠设备改造、更新、改善质量来减少设备损坏和降低维修所需费用的运动。在这个意义上的设备质量改善措施就被称为改善维修（Corrective Maintenance）。
　　此外在设备设计阶段也强调了必须考虑维修预防（MaintenancePrevention），以期减少设备损坏和降低维修费用。
　　其后，因受美国航天工业的影响，在设备设计上追求了无故障，易维修的先进办法，这就是发展成为可靠性工程（Reliability Engineering）的由来。
　　经历了上述过程，回顾我国自从引进PM后的二十年间，在与日本工业发展的同时，由于对PM的重视，无论在技术上、技能上均有了显著的进步。
　　在这个过程中，由于大量成批生产，工人深受机械化、自动化的影响，以致情绪萎靡，同时环境、公害等一系列问题也逐渐突出，因此，对于人身的尊重如何，车间里士气如何，以及干劲如何等各方面开始被重视起来，于是群众便踊跃地自发地组织起ZD（安全无事故）小组和QC小组（质量管理小组）。
　　在这种背景之下以前的PM运动就发展成了全员参加的生产维修，即TPM运动。
  
 什么是PM   
　　所谓PM是预防性维修（Preventive Maintenance）和生产维修（Productive Maintenance）的英文首字母的略语，在日本设备工程协会的PM用语集上有如下定义。
　　预防维修－是指从预防医学的立场出发，对设备的异状进行早期发现和早期治疗。在预防维修由国外引进我国的最初阶段：即算作PM。
　　生产维修－是提高设备生产效能的最经济的维修方法。其目的在于从设备的设计、制造、使用以致维修的全过程，换言之，也就是在设备的一生之中，设法减低设备本身价值和维修等有关设备使用的一切维修费用，以及由于设备劣化而带来的损失等全部费用，用以提高企业的生产效能。作为它的手段，有预防维修，事后维修，改善维修和维修预防。
　　我国最初引进PM的时候（约在1951年），对于事先防止设备故障的活动，叫做“预防维修”，其后，约在 1954年，自从美国GE社特别强调提高生产效能以来，也就是提倡意味着赚钱维修的生产维修以来，PM就形成含有两种意义的词汇，而今天说PM就只是意味着“生产维修”。关于“预防维修”在日本语中多用预防保全一词。

Ch78  
2021.5.8 Software maintainability(软件可维护性)  
https://www.software.ac.uk/resources/guides/developing-maintainable-software#:~:text=Maintainable%20software%20allows%20you%20to%20quickly%20and%20easily%3A,7%20Bring%20new%20developers%20on%20board%20your%20project  
It means that the maintenance personnel understand, correct, and the change will improve this software, it is one of the main features of software quality.  
什么是软件可维护性  
　　软件可维护性是指软件产品被修改的能力，修改包括纠正、改进或软件对环境、需求和功能规格说明变化的适应。
　　在软件维护中往往会出现一系列的问题，例如，是否可能开发出易于维护的软件系统；在进行软件维护时，能否仍然保持软件的完整性；如何才能够提高软件维护的效率等。这些问题实际上涉及软件的可维护性方面。
  
软件可维护性的特征  
　　软件可维护性的五个子特性：
　　（1）易分析性。软件产品诊断软件中的缺陷或失效原因或识别待修改部分的能力。
　　（2）易改变性。软件产品使指定的修改可以被实现的能力，实现包括编码、设计和文档的更改。如果软件由最终用户修改，那么易改变性可能会影响易操作性。
　　（3）稳定性。软件产品避免由于软件修改而造成意外结果的能力。
　　（4）易测试性。软件产品使已修改软件能被确认的能力。
　　（5）维护性的依从性。软件产品遵循与维护性相关的标准或约定的能力。

Ch79  
2021.5.8 Side effects of software maintenance(软件维护的副作用)  
https://users.monash.edu/~jonmc/CSE2305/Topics/13.25.SWEng4/html/text.html  
It means that new errors or new additives due to modification procedures

Ch80  
2021.5.8 inherit(继承)  
https://www.merriam-webster.com/dictionary/inherit  
Subclasses can directly get the priority and characteristics of the parent class, do not need to be redefined

Ch81  
2021.5.22 Senior Manager(高级经理)  
https://www.kanzhun.com/jobs/1451/  
Specific participation in the project or product is not much time, but the success or failure of the project has grown vital. Usually advanced managers participate in the activities of various key links in the project, pay attention to the progress of product development, and make decisions on risk control and resource

Ch82  
2021.5.22 Product manager(产品经理)  
https://www.careerexplorer.com/careers/product-manager/#:~:text=The%20following%20are%20some%20responsibilities%20and%20duties%20of,the%20release%20aspect%20of%20the%20product%20%E6%9B%B4%E5%A4%9A%E7%BB%93%E6%9E%9C...%20  
As the link between the customer and the company's internal communication, monitor the project process, the progress, quality of the project, and the quality. The product manager should be an expert in the field of software engineering, but it is not necessarily an expert in the business. The basic activities of the product manager include: formulating plan, coordinating resources, concerns, and control planning progress, controlling customer expectations. Among them, the control of the customer's expectation value is especially important in the project's project.  
 品牌经理的概述   
　　品牌经理又称产品经理（Product Manager），品牌经理的概念诞生于1931年，创始者是美国宝洁公司负责佳美香皂销售的麦克爱尔洛埃。1926年宝洁公司开始销售一种与象牙香皂相竞争的佳美香皂，尽管使出了浑身解数，也投入了大量的广告费用，但销路一直不畅。负责销售工作的麦克爱尔洛埃通过研究发现，由几个人负责同类产品的广告和销售，不仅造成人力与广告费用的浪费，更重要的是对顾客容易造成顾此失彼。于是，他向公司的最高领导提出一种牌号一个经理的建议，就是一个品牌经理必须把产品的全部销售承担起来。
　　这一建议，一举拓开了宝洁公司的多种产品的销售市场，而且拉长了各种产品的生命周期，如汰渍(Tide)洗涤剂已行销40多年，佳洁士(Crest)牙膏已行销30多年，卡玫尔（Camay）香皂已行销60多年，而象牙(Ivory)香皂已行销110年以上，宝洁也由此成为拥有38个消费品大类的大企业。“品牌经理”制度，为市场营销带来一股清新之风，美国庄臣公司、美国家用品公司等世界范围内的众多大公司都先后采用了这一制度，对产品销售进行全方位的计划、控制与管理，减少人力重叠、广告浪费和顾客遗漏，有效地提高一个或几个品牌在整个公司利润中的比率，提升品牌的竞争力和生命力。
  
 品牌经理的类别与扮演的角色   
　　品牌经理在企业的职能因企业的发展所处的阶段不一样，而不同。一般来说，品牌经理可以分为4种：创立品牌型、维护品牌型、链接品牌型、组合品牌型。品牌经理也分为以下几类。一类是全球品牌经理，它管理着全球性的品牌，其任务是确保各个国家的分公司配合企业整体品牌战略的实施，并且促使跨国界的品牌管理的先进办法在各分公司内得以交流。还有一类叫做品牌类别经理，是为提高分销和物流的效率而设置的。例如，史密斯克兰公司(生产抗酸剂、感冒药、口腔卫生产品等)已经建立起一个全球品牌分类管理结构。每个品牌管理团队拥有自己的研发、市场和品牌管理队伍，决定现有品牌的延伸并为全球品牌扩充新的内涵。　
　　品牌经理在企业中扮演三个角色：一是制定和实施品牌的战略规划；二是向高层管理者汇报品牌战略思想和如何执行；三是培训员工为了公司的盈利自觉执行品牌管理的具体措施。品牌经理与其它部门协同作战，在产品开发、销售、物流、财务和管理各个方面把握品牌发展的大方向。　

Ch83  
2021.5.22 Development manager(开发经理)  
https://www.glassdoor.com/Job-Descriptions/Development-Manager.htm#:~:text=What%20is%20a%20Development%20Manager%3F%20A%20development%20manager,tasks%20required%20to%20successfully%20complete%20the%20company%27s%20initiatives.  
The leader of the specific development process must be served by experts who are familiar with business and development technology. The duties of the development manager are defining demand, determine the appropriate technical framework and system, and ensure that software products are developed according to the design standards.

Ch84  
2021.5.22 Designer(设计师)  
https://www.merriam-webster.com/dictionary/designer  
Software blueprint designer. Usually designers can be divided into three kinds of demand analysts, architect designers, business designers. In small-scale development teams, these three roles are usually borne by one person. Designers must be recognized by experts in business areas and technical fields, with extensive project experience, accurately grasp customer needs and provide feasible implementation. The basic activities of the designer include: performing demand analysis, performing architectural design and function design, writing corresponding documents in accordance with specifications, and spreading design ideas to developers, testers

Ch85  
2021.5.22 Test manager(测试经理)  
https://www.guru99.com/introduction-to-test-management-for-curious-manager.html  
The leader of the test activity is the product quality responsible person within the company (the project manager is the external software quality responsible person). The responsibility of the test manager is to plan and organize testers to test the target product, discover BUG, 窶銀逆rack bugs until the bug is solved; planned and organize user training. Product manager, development manager, designer, test manager as a project's high-level, key role in the success or failure of the project  
工作内容  
制定测试工作计划，包括人员安排、进度、使用的软硬件环境、测试的流程等；  
职业要求  
教育培训：

Ch86  
2021.5.22 Developer(开发人员)  
https://cn.bing.com/dict/search?q=developer&FORM=BDVSP2&qpvt=Developer  
The specific encoding work is performed according to the designer's design results, and the basic unit test of the own code is performed. Typically 3 to 4 developers form a development group that leads to development activities by a Team Leader. The development group Team Leader is a member of the team's technology and business. Team Leader is often duties with a detailed design and walking group member code. Considering that Team Leader needs to perform detailed design, writing documents, and team members communicating, so a Team Leader's development task cannot exceed the average task of developers. For developers, basic technologies, skills, such as programming speech, database application development experience, etc. If developers do not fully have these skills, development managers and project managers should provide the necessary internal or external, training to enable developers to have these necessary skills

Ch87  
2021.5.22 Testers(测试人员)  
https://www.thefreedictionary.com/tester  
According to the plan and test of the test manager, the target product is tested, prepare the test case and test code, discovery, and track bugs; write user manuals; user training and education. Test staff intervene items The sooner theoretically, the sooner, but considering testing human resources, it is usually more appropriate to intervene in demand analysis. For testers, in addition to requiring the same technical skills as required and developers, you should also be familiar with test theory and test methods, and do it as much as possible to observe and think about the perspective of users.

Ch88  
2021.5.22 Project implement personnel(项目实施人员)  
https://www.coursehero.com/file/38574358/2-Implement-project-personnel-training-and-development-rev2pdf/  
The necessary personnel for project nature projects. Project implementers are responsible for software system installation configuration, system cutting, maintenance during operation

Ch89  
2021.5.22 Demand(需求调研员)  
https://dictionary.cambridge.org/dictionary/english/demand  
Exchange with customers to understand customer needs

Ch90  
2021.5.22 Demand analyst(需求分析师)  
https://careers.skyworksinc.com/job/Irvine-Demand-Management-Business-Analyst-CA-92602/685953700/#:~:text=The%20Sr.%20Demand%20Analyst%20will%20be%20part%20of,as%20well%20as%20identifies%20and%20resolves%20forecast%20variances.  
Popularity, demand analysts are exchanged with customers, accurately obtain customer needs. Demand analysts are the people who have deal with the client in the previous period. For the customer, he can represent the entire project group, and his opinions on the project group can represent the customer's opinion, all of the project groups in the project group. Related things must be recognized

Ch91  
2021.5.22 System analyst(系统分析师)  
https://www.techopedia.com/definition/4816/systems-analyst#:~:text=-%20Definition%20from%20Techopedia%20What%20Does%20Systems%20Analyst,are%20functioning%20as%20effectively%20and%20efficiently%20as%20possible.  
It is mainly responsible for designing system overall architectures, regulatory system development schedules, determining software processes and frameworks in software according to system analysis results. System analysts are the CEO of the project team, and all aspects of the project are the project's progress, and it is also the key to the success of the project.  
 什么是系统分析员   
　　系统分析员又称系统分析师，是指具有从事计算机应用系统的分析和设计工作能力及业务水平，能指导系统设计师和高级程序员的工作的一族。在软件开发流程中主要从事需求分析、信息系统项目架构设计（包括概要设计和详细设计）、开发阶段的主要模块的规划、设计和测试，同时也涉及可行性分析的工作。英文system analyst，简称SA。 系统分析师(SA)是负责设计与开发应用软件系统．使其正确的反应出有效的信息，协助企业经营者管理、营运公司的运作者。系统分析师是抽象模型的建立者，他们需要专业的conceptionmodel（概念模型）知识和基础编程技巧。富有经验的系统分析师往往是优秀技术专家和项目管理者的结合体，他们精通系统论和控制论，擅长将杂乱无章的复杂性问题整理调顺，并将其模块化，从而使项目的实施走向成功。

Ch92  
2021.5.22 Module designer(模板设计师)  
https://docs.devexpress.com/eXpressAppFramework/112828/design-time-features/module-designer  
Further refinement of modules divided by system analysts and architects, ensuring that each module is completed according to established standards and requirements. Database designer: Also known as DBA, is the only person in the project group to perform direct operations directly

Ch93  
2021.5.22 Quality assurance personnel(质量保障人员)  
https://kta.com/kta-university/quality-assurance-quality-control-roles/#:~:text=QC%20personnel%20perform%20the%20testing%20and%20compare%20the,requirements%20and%20testing%20procedures%2C%20they%20cannot%20contol%20quality.  
Supervise and control the quality of software products during the production process. Usually a team has a quality assurance team, consisting of one or more people.

Ch94  
2021.5.22 Software configuration management(软件配置管理人员)  
https://www.sciencedirect.com/topics/computer-science/software-configuration-management  
Manage the various products generated in the management software development, the specific work is to identify, change control, status report, etc. for important articles. Usually a team has only one software configuration management personnel

Ch95  
2021.5.22 Dependency(依赖)  
https://dictionary.cambridge.org/dictionary/english/dependency  
A "use" relationship between the entity suggests that after the specification of an entity changes, it may affect other instances depend on it (Figure D). More specifically, it converts to any type of reference to a class or object not in the instance scope

Ch96  
2021.5.22 Association(关联)  
https://www.merriam-webster.com/dictionary/association  
A structured relationship between the entity indicates that the object is interconnected. The arrow is optional and it is used to specify the navigation capabilities. If there is no arrow, hint is a two-way navigation capability

Ch97  
2021.5.22 Aggregation(聚合)  
https://blog.csdn.net/qq\_31868149/article/details/82912139  
Represents the overall / partial relationship between two classes. The aggregation suggests that the whole is conceptually at a higher level than the local, and the association suggests that both classes are conceptual at the same level.

Ch98  
2021.5.22 Composition(合成)  
https://dictionary.cambridge.org/dictionary/english/composition  
Instead of "local" in the "overall" survival duties. Synthesis is also non-shared. Therefore, although local does not have to be destroyed with the overall destruction, the whole is either responsible for maintaining local survival status or is responsible for destroying it. Local cannot share with other overall

Ch99  
2021.5.22 Generalization(泛化)  
https://www.britannica.com/topic/generalization  
Uml element is used to model inheritance  
概念  
泛化关系是类元的一般描述和具体描述之间的关系，具体描述建立在一般描述的基础之上，并对其进行了扩展。具体描述完全拥有一般描述的特性、成员和关系， 并且包含补充的信息。例如，抵押是借贷中具体的一种，抵押保持了借贷的基本特性并且加入了附加的特性，如房子可以作为借贷的一种抵押品。一般描述被称作 父，具体描述被称作子，如借贷是父而抵押则是子。泛化在类元（类、接口、数据类型、用例、参与者、信号等等）、包、状态机和其他元素中使用。在类中，术语超 类和子类分别代表父和子。  
简介  
细分无需多言，强调的是目标人群的聚焦和集中。细分要求的是准确集中。而泛化则相反，无限扩大目标人群，虽针对某一种疾病，却试图将搭点边的人群都一网打尽。 泛化则要求分散、要求“广”和“多”。

Ch100  
2021.5.22 Realization(实现)  
About Realization Technologies | Realization  
An entity defines a contract, and another entity guarantees to fulfill the contract