一基本概念

1 软件定义

计算机程序、方法、规则、相关的文档资料以及在计算机上运行程序时所必需的数据。

2 简述软件危机的表现有哪些？以及解决软件危机的途径有哪些？

软件危机的主要表现包括：

(1)软件开发进度难以预测，开发成本难以控制，导致超预算、超时；

(2)产品功能难以满足用户需求；

(3)软件产品质量无法保证；

(4)软件缺少适当的文档资料，维护困难；

(5)软件成本超过硬件成本；

(6)软件开发生产率的提高速度跟不上计算机应用普及深入的趋势；

解决软件危机的途径有：

1 管理措施：项目管理、配置管理、过程管理、质量控制

2 技术措施：开发过程、开发技术与方法和开发工具

3 软件工程的定义

软件工程是指导计算机软件开发和维护的一门工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验和当前能够得到的最好的技术方法结合起来，以经济地开发出高质量的软件并有效地维护它。这就是软件工程。

4 软件工程的 10 个知识领域

软件工程过程(SoftwareEngineeringProcess)

软件工程工具和方法(SoftwareEngineeringToolsandMethods)

软件需求(SoftwareRequirement)l

软件设计(SoftwareDesign)

软件构造(SoftwareConstruction)

软件测试(SoftwareTesting)

软件维护(SoftwareMaintenance)

软件配置管理(SoftwareConfigurationmanagement)

软件工程管理(SoftwareEngineeringmanagement)

软件质量(SoftwareQuality)

5 软件工程的目标是什么，软件工程的三要素都是什么？

软件工程的目标：软件工程必须以有组织的质量保证为基础，进行全面质量管理，不断地过程改进使软件工程方法走向成熟。

软件工程的三要素包括：过程、方法和工具

过程为及时合理地开发出满足用户需求的计算机软件而进行一系列有组织的活动。过程定义了技术方法的采用、工程产品(包括模型、文档、数据、报告、表格等)的产生、里程碑的建立、质量的保证和变更的管理。

方法为软件开发提供“如何做”的技术，它涵盖了项目计划、需求分析、系统设计、程序实现、测试与维护等一系列的开发活动如何来做。开发方法经历了从面向结构、面向对象、面向组件到面向服务的发展工程。

工具为过程和方法提供自动的或半自动的支持。这些软件工具被集成起来，建立起一个支持软件开发的系统，称之为计算机辅助软件工程(CASE，ComputerAidedSoftwareEngineering)。

6 软件工程的七条基本原理

美国著名的软件工程专家巴利•玻姆(BarryBoehm)提出了软件工程的七条基本原理：

1．用分阶段的生命周期计划严格管理；

2．坚持进行阶段评审；

3．实行严格的产品控制；

4．采纳现代程序设计技术；

5．结果应能清楚地审查；

6．开发小组的人员应少而精；

7．承认不断改进软件工程实践的必要性。

7 简述软件过程的定义，软件过程又可以分为那几个类型？

软件过程(SoftwareProcedure)是为获得软件产品，在软件工具支持下由软件工程师完成的一系列软件工程活动。

软件过程可概括为基本过程、支持过程和组织过程等三种类型。

其中，基本过程包括：获取过程、供应过程、开发过程、运作过程和维护过程。支持过程包括：文档编制过程、配置管理过程、质量保证过程、验证过程、确认过程、联合评审过程和问题解决过程等过程。组织过程包括：管理过程、基础设施过程、改进过程和培训过程。

8 里程碑(MileStone)思想：

阶段工作的目标进行总结、评审、调整和部署下一个里程碑。目的：合理分配，细化管理“粒度”，降低项目风险。

9 基线思想

基线是指一个(或一组)配置项在项目生命周期的不同时间点的一种状态，各阶段有各阶段的基线：需求基线、设计基线、测试基线等。基线一旦建立后变化需要受控制。

10 简述软件生存周期的概念，说明软件生存周期划分为那几个主要时期？每个时期有包括哪些主要阶段？

软件生存周期是指软件产品从定义到开发、使用和维护，直到最终被弃用的时期，称为生存周期。

生存周期的可划分为计划时期、开发时期和运行时期等三个主要时期。

其中计划时期包括问题定义和可行性研究两个阶段。开发时期包括需求分

析、总体设计、详细设计和实现等四个阶段。运行时期的又称为运行和维护阶段。

10 简述教材中介绍了那些软件开发模型？这些模型各有什么特点？

教材中重点介绍了瀑布、原型、增量、螺旋四个传统模型和 RUP、XP 两个现代模型。

其中，瀑布模型严格按照生存周期开发软件，每个阶段必须完成规定的、完整、准确的合格文档，前一阶段的输出文档就是后一阶段的输入文档。

其主要特点包括：①活动间具有顺序性和依赖性；②推迟实现的观点；③质量保证的观点；

快速原型模型法是开发人员在一个基本的需求的基础上快速开发出一个软件原型，然后由用户使用和评价原型、开发人员根据用户意见再修改原型，然后再使用评价再修改、直至将原型进化为最终产品。

快速原型模型的特点包括：①做出系统原型，及早向用户展示系统要实现的界面及功能，增强用户的合作信心；②直观化的表达，容易交流，消除理解上的歧义；③修改集中在前期的原型确认上，较大程度减少后期实施中的返工。④入手快，加快开发进度；

增量模型是一种演化模型，先完成一个系统子集的开发，再按同样的开发步骤增加子集,如此递增下去直至满足全部系统需求。每个增量可按快速原型法进行。

增量模型的特点包括：①无须等待获取完整需求就可入手，尽快见到成果，增强双方信心；②分步开发，降低复杂性和难度，减少技术风险，并可并行开发；③边开发边投入，可及早发现问题，减少投资风险；④各个子集是逐渐并入已有的系统中，加入子集不能破坏已构造好的部分，这需要软件具备开放式的体系结构；⑤适用于需求不完整的软件开发，指的是需求逐渐摸清、逐步完善，并非随意改变，需求改变过大会导致整体性失控。后面要介绍的 XP（极限编程）属于该模型。

螺旋模型(Spiralmodel)是一种融合了瀑布模型、快速原型模型和增量模型的演进模型，并引入风险分析机制。适合大型复杂的系统开发。

螺旋模型特点包括：①多种模型结合的一种演进模型，融合了瀑布模型、快速原型和增量模型的所有特点，融进了循环往复、迭代演进的思想；②增加风险分析，一旦风险成立，原方案应终止、修订，力求风险可控③客户始终参与每个阶段的开发,每个阶段的成果需客户确认，避免错误的积累。

统一过程 RUP(RationalUnifiedProcess)是由 Rational 公司在推出统一建模语言 UML 后，推出的一个软件开发框架 RUP，称为软件统一开发过程。

11 统一过程 RUP 定义了那几个主要阶段？

初始阶段(Inception):主要完成商业需求，确定项目边界。里程碑是生命周期目标(LifecycleObjective)，评价项目基本的生存能力。

细化阶段(Elaboration):主要完成领域问题分析和软件设计。获取用户需求（功能和非功能需求），建立需求模型；进一步确立体系结构和设计软件结构等工作。里程碑是生命周期结构(LifecycleArchitecture)。

构造阶段(Construction):主要完成系统实现、测试，里程碑是初始功能(InitialOperational)，产品版本常被称为“beta”版。

交付阶段(Transition):重点是确保软件对最终用户是可用的。里程碑：产品发布(ProductRelease)

12 统一过程 RUP 工作流

6 个核心过程工作流(CoreProcessWorkflows)

- 商业建模(BusinessModeling)：弄清项目边界和约束，做出计划。
- 需求(Requirements)：描述系统应做什么，开发人员和用户达成需求基线。
- 分析和设计(Analysis&Design)：将需求转化成计算机可以实现的模型。
- 实现(Implementation)：用程序设计语言将设计模型组织成可执行的文件、数据。
- 测试(Test)：是发现软件中的错误，在实验环境下验证所有的需求是否被正确的实现。

- 部署(Deployment)：将软件分发给最终用户，安装在真实的环境下，由用户操作运行。

3 个核心支持工作流(CoreSupportingWorkflows)是对核心过程工作流的配套支持和管理，保障核心过程工作流顺畅、高效运行。

- 配置和变更管理：工作文档的管理，在版本更新、需求变更中做到各类文档及时、同步跟踪，保证各文档内容完整、一致。
- 项目管理(ProjectManagement)：资源配置、评估监控、风险控制、计划调整等管理工作，目的效益最大化。
- 环境(Environment)：软件开发环境，包括人员、设备、过程和工具，以及各种规范、指导手册和保障措施。

13 简述 rup 模型中基线与里程碑的概念，二者之间的关系。

基线，是软件文档或源码(或其它产出物)的一个稳定版本,它是进一步开发的基础，也可以理解成为一个阶段的起点并已经制定了相应的工作标准，并且只有经过授权后才能变更这个标准。

里程碑，是计划中确定的阶段性工作完成目标，要求提交阶段交付物，作为阶段评估的标准。

基线和里程牌的关系：基线是为了建立参照点，是阶段的起点；里程牌是建立阶段性目标，是阶段终点，最后的里程牌可能是一次迭代的终结。

14 简述软件计划的目标和主要工作.

软件计划的目标：研究项目的可行性，研究合理地运用软件项目开发所需的资源、经费，掌握开发进度，控制项目开发过程按此计划进行。

主要工作包括:确定项目实施范围、定义递交的工作成果、评估实施过程中主要的风险、制定项目实施的时间计划、成本和预算计划、人力资源计划等。

15 软件计划的活动有哪些？这些活动的内容是什么？

软件计划主要活动包括：问题定义，可行性研究，项目计划。

这些活动的内容是：

问题定义：确定项目实施范围，回答项目“做什么？”的问题。

可行性研究：项目的必要性和可能性。

制定项目计划：编制项目开发计划。

16 简述问题定义的目的和主要任务。

问题定义的目的：弄清要计算机解决的根本问题所在(要解决的问题是什么？)，确定新系统的作用域，以及项目所需的资源、工期和经费。

问题定义的主要任务：编写项目报告提交审查，作为可行性分析的依据。

17 简述可行性分析的目的、任务和内容。

可行性分析的目的：确定项目的必要性和可能性。

可行性分析的任务包括：可行性分析；写可行性研究报告；编制开发计划。

可行性分析的内容包括：技术、经济和社会三个方面的可行性：

18 简述需求分析的目的、必要性和参与角色

需求分析的目的是：弄清用户对系统的细节要求，完整、准确、清晰、具体地回答目标系统“做什么”。准确地理解用户提出的软件功能、性能及其环境的要求。

需求分析的必要性：用户与开发者的知识领域不同，产生歧义；软件开发失败 50%是需求不合理，早期错误易放大。

参与角色：开发方包括分析师、设计师和架构师。用户方包括领域专家、用户和部门负责人。

19 需求开发的任务有哪些？

需求开发的任务包括

需求获取：收集用户对目标软件系统在功能、性能、行为、设计约束等方面的期望。

需求分析：通过符号和文字说明描述系统模型，使用户和开发者间建立共同语言基础，消除理解上的歧义的过程。

需求说明：既编写需求文档，也称编写需求规格说明书。需求说明书是需求分析阶段的最终成果，也是需求分析阶段复审的依据；是用户领域专家、软件分析师、软件设计师共同交流的途径和媒介；是交付给用户文档的一部份；

需求验证:即需求评审。根据需求说明书，分析师、设计师、客户会审文档，对需求的正确性、一致性、完整性、无二义行进行评审、确认。

20 需求的层次

软件需求包括三个不同的层次：业务需求、用户需求、功能需求，也包括非功能需求。

1．业务需求(businessrequirement)

业务需求是反映企业/组织对软件系统的高层次目标要求,即软件系统的建设目标。

业务需求通常是“问题定义”或“可行性研究”阶段获取的内容；在需求规格说明书中反映在项目背景、系统目标或任务概述的描述中。

获取的主要对象是客户方的高管、专家、部门负责人。

2．用户需求(userrequirement)

用户需求：用来描述用户使用产品必须要完成的任务；使用业务领域的术语描述，采用开发者与用户都能理解的语言和图形表达。用户需求是经过调查、归纳后双方认同的结果。

获取的主要对象是部门负责人、软件的操作者或称终端用户。

3.功能需求(functionalrequirement)

功能需求定义了开发人员必须实现的软件功能，结果在需求规格说明书中;功能需求用软件行业术语表达：通常是需求建模的结果即目标系统的逻辑模型，如结构化的功能模型、数据模型、行为模型，面向对象的类模型等。

4.非功能需求

特性是指一些非功能需求，是满足业务需求的性能要求。如界面的交互性、数据的安全性、数据的事务性、用户的并发性、响应的快速性、操作的实时性、错误与异常的恢复性、软件的容错性等等。项目的失败或拖延一般不是在功能上，而恰恰倒是在性能要求上，因为这些性能与软件的体系结构有关，与构成系统的网路与硬件环境等底层技术有关，往往超越一般开发人员的技术能力。

21 需求获取的一般方法

① 需求获取方法以采访、观察、座谈、对先前的系统版本的测试等。必要时采用快速原型法。

② 先集中在使用者对系统的观点上，以收集用户原始资料，数据、工作方式、工作流程、使用要求等为工作起点，深入到部门、车间、班组，做好原始纪录；

③ 然后根据对问题及环境的理解与开发经验，改正用户需求的模糊、歧义和不一致性要求，排除用户的不合理要求，挖掘用户尚未提出但具有价值的潜在需求，使用户需求逐步精确化、一致化和完全化；

④ 需求获取非一次完成：需要往复进行、逐步深化。

⑤需求获取的内容：写进“需求规格说明书”，确认。

22 需求获取的策略

①循序渐进的策略；

②确定优先级：先进行重点的需求调研，有助于识别出重大的风险，并为制定迭代计划提供指导；

③不要陷入技术：需求未明确，应回避对技术问题讨论。

④挖掘用户需求:“诱导式”就是挖掘用户需求。客户并非 IT 专业人士，需求的概念是模糊的、笼统的，而且尺度难以把握，预测潜在需求。

⑤区分不必要的需求：客户对有些需求提不出来，自然也会提出一些不必要的需求。

23 简述概要设计和详细设计的内容。

软件设计包括概要设计和详细设计。概要设计又分为体系结构设计和领域问题结构设计。

体系结构设计：是支撑和管理软件运行的环境设计。由于现代的软件是处在操作系统、网络、各种服务器共同搭建的环境下运行，并且具有并发、安全、事务等多方面的管理，是软件设计优先考虑的问题。

领域问题结构设计：满足需求的软件功能设计，核心所在。将领域问题的分析模型细化成软件结构模型，也就是划分软件的模块结构及确定模块之间的关系。

②详细设计又分为如下三个部分：

对模块内部的过程和数据结构进行设计。也就是对模块内进行算法分析和程序设计。

人机交互界面的具体设计，还有与其它外部系统接口设计。

完成对数据库的物理设计

概要设计是根据需求确定软件和数据的总体框架；详细设计是进一步精化成软件的算法和数据结构。

24 简述衡量软件模块独立性的度量标准有哪些？

模块独立性是指模块能够完成独立的功能；模块符合信息隐藏和信息局部化原则；模块间关联和依赖程度尽量小。

衡量软件模块独立性的度量标准的指标有取决于模块的内部特征的指标内聚度和取决于模块的外部特征的指标耦合度。

内聚度：一个模块内部各个元素间(语句和程序段)彼此的紧密程度的度量。

耦合度：指软件结构中各模块间相互联系紧密程度的一种度量。

25 简述内聚度的七个等级？

内聚度表示一个模块内部各成分之间彼此结合的紧密程度。内聚度按其高低程度可分为七级，高内聚度模块独立性强，设计尽可能提高模块内聚度。

1. 偶然性内聚：是指一个模块内各成分为完成一组功能而组合在一起，它们相互之间即使有关系，也很松散。

2. 逻辑性内聚：模块内完成的诸任务逻辑上相关。该类内聚的缺点是执行中要从模块外引入用作判断的开关量,从而增加了块间偶合(控制偶合)。

3. 时间性内聚：如果一个模块包含的诸任务必须在同一时间段内执行，则称之为时间性内聚

4. 过程性内聚：模块的过程性内聚度是指模块内成份彼此相关，并且必须按特定的次序在本模块内执行；

5. 通讯性内聚：是指模块中各组成成分都将对某个数据结构的同一区域进行操作，以达到通信的目的。

6. 顺序性内聚：一个模块内的各处理成分均与同一功能相关，且这些处理必须顺序执行，通常，一个处理成分的输出是另一个处理成分的输入。

7. 功能性内聚：模块内所有成分形成一个整体，完成单个功能，则称功能内聚，功能内聚是最高程度的内聚形式。

26 耦合度的七个等级

耦合度是模块独立性最显著特征。耦合度按其高低程度可分为七级，松耦合是软件设计一直追求的目标。

① 非直接耦合:模块不依赖另一个模块能独立工作，这是最松的耦合。

② 数据耦合:两模块间通过参数交换数据信息，则称这两模块为数据耦合。

③ 特征耦合:模块之间除传递关键数据外还附加公共数据。

④ 控制耦合:如果两模块间通过参数交换信息，此时若传递的信息中含有控制信息，则为控制耦合。

⑤ 外部耦合:当若干模块均与同一个外部环境关联，它们之间便存在外部耦合。

⑥ 公共耦合:当若干模块通过全局的数据环境相互作用时，它们之间存在公共耦合。

⑦ 内容耦合:当一个模块使用另一个模块内部的数据或控制信息；一个模块直接转移到另一个模块内部等,模块间的耦合就是内容耦合。

27 简述模块的作用域与控制域的概念及其相关设计原则。

模块的作用域：从功能方面考虑，受模块内一个判定影响的所有模块的集合；

模块的控制域：从结构方面考虑，包括它自己及其所有下属模块的集合。

相关设计原则是：模块的作用域应在控制域之内。

28 详细设计的表达方式有哪些？

1.伪代码(Pseudocode)：是一种算法描述语言，也称 PDL 语言(ProgramDesignLanguage)。伪代码介于自然语言与编程语言之间，用伪代码描述的算法可以容易用任何一种编程语言实现。伪代码表达算法必须结构清晰、代码简单、可读性好。

2.程序流程图：用图形符号表达算法，直观表达循环、分支等复杂结构，是喜闻乐见的表现形式。

3.盒图(N-S)(Nassi 和 Shneiderman)：也是一种图形符号表达方式，同样可以表达各种流向控制，但比程序流程图紧凑、功能域明确。

4.PAD 图(PAD-ProblemAnalysisDiagram)[28]：同样是用图形符号表达算法，但它具有结构化的表达方式，因此结构十分清晰，很容易翻译成程序代码。PAD 支持自顶向下，逐步求精方法的使用。判定表与判定树：对于规则较多，判定条件较复杂的情况，宜采用这两种方法表达。

5.判定表与判定树：对于规则较多，判定条件较复杂的情况，宜采用这两种方法表达。

29 简述面向数据流的结构化分析 SA 方法中，有哪些建模方法？

功能建模:数据流图 DFD(DataFlowDiagram)+数据字典 DD(DataDictionary)；

数据建模:实体关系图 ERD(EntityRelationDiagram)；

行为建模:状态转换图 STD(StateTransformDiagram)；

加工说明 PESPEC(ProcessSPECification)和判定表等辅助工具。

30 简述数据流图（DFD）的图形符号有哪些？

①数据流：表示数据流的名称和数据的流向(从加工出发或流向加工)；

②外部实体：系统外与系统交互的人或实体；

③数据加工：数据处理；

④数据存储：数据进行持久保存的环节；

31 简述数据字典的作用、内容和组成元素有哪些？

数据流图描述了数据加工，但没有描述数据的内容。数据流图必须与描述并组织数据条目的数据字典 DD(DataDictionary)配套使用。

数据字典描述的对象：描述数据流图中出现的所有数据和加工。这里的数据描述是概念性的，属数据结构的抽象描述；加工采用加工小说明进行概念性的描述。

数据字典的组成元素包括：数据流条目、数据存储条目、数据项条目；加工条目(也称为小说明)；

32 简述面向对象的基本概念有哪些？

1．对象与面向对象

对象（Object）：即表示客观世界中的某个具体的事物。面向对象（ObjectOriented）：是人类的活动，是人类认知、观察客观事物的方法论。

2．面向对象的抽象与分类

3．类的封装与对象的整体性

4．关联性与交互性

客观事物都不是孤立存在的，万物之间相互依存、相互交流。关联性表达客观事物的社会性、共存性、组织性，是静态的结构描述。消息机制是对象的交互性，表示对象生存环境的依赖性。

5．继承性

对事物的分类本身就体现继承性。软件开发利用继承性可对 Object 更好

地分类，软件结构更严谨，代码的复用性更强。

6．多态性

对象在不同的条件下，同样的行为会表现不同的效果，这就是 Object 的多态(polymorphism)。

面向对象编程语言提供抽象类、接口、重载等技术支持多态的实现。

33 面向对象的五大特性有哪些？

面向对象的特性有抽象性、封装性（用于表示对象整体特征）、继承性、多态性和消息机制等五大特性。

34 简述面向对象开发过程的内容有哪些？

①需求获取：开发者以 OO 的观点(OOV)来观察客观世界的目标即获取需求，然后用自然语言写到需求规格说明(OOS)中，也就是对客观世界的最高层抽象。

②面向对象分析 OOA(ObjectOrientedAnalysis)与面向对象设计 OOD(ObjectOrientedDesign)。

③面向对象编程 ObjectOrientedProgram(OOP)与面向对象测试 ObjectOrientedTesting(OOT)是代码实现过程，它依赖于编程语言和工具。

④软件维护 ObjectOrientedSoftwareMaintenance(OOSM)。

35 与传统的软件开发方法相比较，面向对象开发的主要优点有哪些？

1．自然性即客观性

2．操作数据对象而非数据实体

3．阶段衔接平滑

4．结构性好、复用性强

5．提高扩展性和维护性

36 简述 UML 中定义了那几种图？并简述这些图的作用。

用例图(UseCase)	描述系统参与者与领域问题的功能
类图(Class)	描述系统的逻辑结构，类、接口及它们的协作关系
包图(Package)	描述类的复用组织—分组
对象图(Object)	描述类的实例在某时刻的关系
构件图(Component)	描述系统按构件组成上的关系
配置图(Deployment)	描述系统运行环境的配置情况
时序图(Sequence)	描述某些对象共同合作完成某项功能而按时间顺序进行的消息传递
协作图(Collaboration)	描述某些对象共同合作完成某项功能的依赖关系 比顺序图多链
活动图(Activity)	描述某个用例按事件流转所经历的的活动，即业务流程 泳道的主要作用是按职责 组织模型中的各项活动
状态图(Statechart)	描述某个业务流程按事件流转所经历的状态，即状态机

37 简述用例图中有哪些模型元素？并简述其含义？

①参与者：指存在于系统外部并与该系统发生交互的人或其他系统，代表系统的使用者或使用环境。

②用例(UseCase)，用例用于表示系统提供的服务，它定义了系统是如何与参与者交互，描述了参与者与系统之间的交互过程。

③角色与用例间的关系关联，它表示参与者与系统中的哪些用例交互。

用例之间的关系：包含<<include>>和扩充<<extend>>关系以及泛化关系。

参与者之间的泛化关系。

38 简述类图中有哪些模型元素？并简述其含义？

类：类名、属性、方法（可见性、作用域）

特殊类：接口

类之间的关系的定义、表示和属性：

关联、依赖、聚合、组合、泛化、实现。

各种类关系之间的关系。

各种关系的强弱顺序：

泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖

组合是一种特殊的聚合
包含是一种特殊的扩充

39 简述包图中有哪些模型元素？并简述其含义？

40 简述活动图中有哪些构成元素？并简述这些元素的含义？

41 简述 OOA 模型的结构

OOA 的核心任务是搞清用户需求，最终要建立起 OOA 模型。UML 的 OOA 模型由“用例模型”和“概念模型”两大部分组成。

①用例模型，是将用自然语言描述的领域问题，转换成 UML 语言表达的模型，主要面向用户，反映用户需求。

完整的用例模型由用例图和业务场景描述两个部分组成，用例图表示功能的划分；

业务场景描述则对每个用例的事件流进行描述；

②概念模型（类模型/结构模型/静态模型）。

将用例模型映射成类模型：从用例模型中找出类，面向设计人员。

主要工作是：根据用例图进行类的划分与封装；描述类间的静态关系与结构；用交互图表达类对象间的消息传递。

42 对象间的可访问性

①属性可见性：B 是 A 的一个属性（关联、聚合）；

②参数可见性：B 的对象是 A 的一个方法的参数；

③局部声明可见性：B 的对象是在 A 的一个方法中声明的一个局部变量；

④全局可见性：B 的对象在某种程度上全局可见；

43 对象持久化对象持久化常用技术

实体类的实例称为数据对象，对象持久化主要用于数据对象的持久化，简称数据持久化。

一个数据对象的持久化就是保存到实体表中的一条记录，对实体对象的访问就是操作属性的值。

对象持久化常用技术

①对象的序列化

指将对象的相关信息（对象序列号、属性名、属性值等）转换为字节流，然后再把字节流写入数据流。可以把对象这些信息存储在本地的文件里，也可以把它通过网络传输到远程。通过对象反序列化，得到原对象完全相同的副本。

②对象持久化到数据库中

数据库可以是对象数据库或关系数据库。

③用 XML（eXtensibleMarkupLanguage）存储。

44“实体模型”到“关系模型”的 OR 映射

(1) 一个对象类可以映射为一个以上的库表，当类间有一对多的关系时,一个表也可以对应多个类。

(2) 对象关系(一对一、一对多、多对多)的映射可能有多种情况，但一般映

射为一个表或多个表，在表间定义相应的主键 PK（Primarykey）和外键 FK（Foreignkey）建立实体间的关系。

(3) 单一继承的泛化关系可以对超类、子类分别映射表,也可以不定义父类表而让子类表拥有父类属性；反之，也可以不定义子类表而让父类表拥有全部子类属性。

(4)对多重继承的超类和子类分别映射表，对多次多重继承的泛化关系也映射一个表。

(5)对映射后的库表进行冗余控制调整,使其达到合理的关系范式。

45 软件测试的定义

软件测试是为了发现错误而运行程序的过程；软件测试的目的是发现程序中的错误，是为了证明程序有错,而不是证明程序无错；测试对象不仅是程序，还应该包括开发过程中产生的所有产品，包括文档，其目的是为了尽早地、尽可能多的发现并排除软件中潜在的错误。

46 软件测试的基本原则

①Who 来测试？

测试工作应该由独立的、专业的软件测试机构来完成，设计人员和程序员要参与测试；对测试结果一定要有一个确认的过程，一般由角色 A 测试出来的错误，一定要有一个角色 B 来确认，严重的错误可以召开评审会进行讨论和分析；

②测试 What？

程序员交付的模块、系统和文档；

③测试 Extent？

设计测试用例，充分覆盖所有条件或所有语句即可；

④When 测试？

尽早和不断的测试，即将这种“测试”贯穿于软件开发的各个阶段，坚持各个阶段的技术评审，以便尽早地发现和预防错误；

⑤How 测试？

设计测试用例时不仅要考虑到合法的输入，还要考虑到不合法的输入以及各种边界条件；对发现错误较多的程序模块，应进行重点测试。

47 软件缺陷，软件缺陷的属性：

从产品内部看，缺陷是软件产品开发或维护过程中存在的错误、毛病等各种问题；从产品外部看，缺陷是系统所需要实现的某种功能的失效或违背。

软件缺陷的属性：

缺陷标识、缺陷类型、缺陷严重、程度缺陷、优先级、缺陷状态、缺陷起源、缺陷来源、缺陷根源等。

48 简述测试用例的概念

测试用例（TestCase）是关于具体测试步骤的文档，以判断被测软件的工作是否正常。

内容包括：测试目标、测试环境、输入数据、测试步骤、预期结果等。从表现形式上看，测试用例可以是纯文本的文档，也可以是用程序设计语言编写的一段代码。

49 简述基本测试方法的分类情况

测试方法分类				内容
静态测试			走查	
			评审	
动态测试	白盒测试	逻辑覆盖	语句覆盖	语句覆盖是最简单、最弱覆盖。它只覆盖可执行语句至少执行一次。
			判定覆盖	判定覆盖又叫分支覆盖，是对每个判定式取真、假各一次，使每个判定的每个分支都至少执行一次，同时满足

		法		语句覆盖。
			条件覆盖	条件覆盖是把程序中每个判断的每个条件为真和假各取值一次。条件覆盖深入到判定中的每个条件，但不一定满足判定覆盖的要求。
			判定/条件覆盖	判定/条件覆盖能同时满足判定、条件两种覆盖标准的取值。就是使得判定中每个条件的所有可能取值至少执行一次，同时每个判定本身所有取值至少执行一次。
			条件组合覆盖	条件组合覆盖是按每个判断的所有条件取值进行组合。这是 5 种覆盖中最强的覆盖。它不但可覆盖所有条件，还可覆盖所有判断的可取分支。
		基本路径覆盖法		基本路径测试步骤： ①导出程序流程图的拓扑结构-流图(控制流程图); ②计算流图 G 的环路复杂性 V(G); ③确定只包含独立路径的基本路径集; ④设计测试用例;
	黑盒测试	等价类划分法		对测试数据进行区间划分，从这些区间中选取典型值作为用例代表，认为测试等价类中的一个代表值的结果就等于对该类其它值的测试。
		边界值分析法		边界值分析法就是对输入或输出的边界值进行测试的一种方法。
		错误推测法		
		因果图法		

50 简述软件测试过程的主要内容。

1．需求与设计评审

2．单元测试(UnitTesting)

3．集成测试

4.功能测试

5.系统测试

6.验收测试

7.α 与 β 测试

51 简述单元测试(UnitTesting)的主要内容。

单元测试又称模块测试，是针对软件设计的最小单位程序模块（函数、类等）进行正确性检验的测试工作。 以系统详细设计为基础的测试

单元测试采用黑盒+白盒混合方式，采用黑盒测试为主为先，白盒测试为辅为后的策略。

1)用黑盒进行模块接口测试

2）用黑盒进行模块边界条件的测试

3）用白盒进行模块局部数据结构和算法的测试

4）用白盒进行模块中独立路径的测试

5）模块中各条错误处理路径的测试

52 简述集成测试的主要内容

集成测试也叫组装测试或联合测试。

集成测试是在单元测试基础上，再将单元按照概要设计规格说明的要求组装成更大的模块、子系统或系统。

1）非渐增式集成测试

非渐增式集成测试又叫一次性集成测试，就是把所有经过单元测试的模块按照设计规格说明书一次性组装成系统，然后进行统一的测试。

2）渐增式集成测试

渐增式集成测试即把下一个要测试的模块同已经测试好的模块结合起来进行测试，测完后，再把下一个应该测试的模块结合进来测试。

53.简述验收测试的概念

验收测试是软件交付之前的最后一个测试操作，验收测试的目的是确保软

件准备就绪，并且可以让最终用户将其用于执行软件的既定功能和任务。

54.α 与 β 测试

这两种测试是针对商用软件的系统测试。

商用软件与合同定制式软件不同，它面向的使用群体数量大、不确定，没用针对性的验收用户。因此在软件正式面市之前免费供用户试用，由用户在试用中发现问题，这就是 β 测试。

提供给用户的 β 版如果 BUG 太多，客户将无法试用和承受，因此首先软件开发组织内部人员模拟各类用户对即将面市软件产品进行测试，此时称为 α 测试。

55 简述软件维护的概念

软件维护是在软件交付使用之后，为了改正错误或满足新的需求而修改软件的过程。

56 软件维护的分类

纠错性维护（CorrectiveMaintenance）

纠错性维护是在软件交付后，纠正哪些在运行中发现的残留错误，也称改正性维护。

适应性维护（AdaptiveMaintenance）

为适应软件运行环境（软件生态环境）的变化而修改软件的活动称为适应性维护。

完善性维护（PerfectiveMaintenance）

根据用户在软件使用过程中提出的建设性意见（需求变化）而进行的维护活动称为改善性维护。

预防性维护（PreventiveMaintenance）

为了进一步改善软件的可靠性和易维护性，或者为将来的维护奠定更好的基础而对软件进行修改。

57 简述软件维护的副作用及困难有哪些？

1．维护的副作用

维护的副作用就是指由于维护或在维护过程中其他一些不期望的行为引入的错误。

引起副作用的维护修改可分三类：代码副作用；数据副作用；文档副作用。

2．维护工作面临的困难

周期长、难度大、费用高。维护费用高达开发费用的 55%—70%，而且逐年上涨。

维护中还可能引入新的潜在错误。分析设计的缺欠、非维护者开发、现场追踪等等。

58 简述软件配置项的概念和内容。

软件配置项 SCI（softwareConfigurationItem），软件生存周期各个阶段活动的产物经审批后即可称之为软件配置项。软件配置项包括：

与合同、过程、计划和产品有关的文档和资料；

源代码、目标代码和可执行代码；

相关产品，包括软件工具、库内的可重用软件、外购软件及顾客提供的软件等。

59 简述结构化维护和非结构化软件维护的概念。

1．非结构化维护是指满足如下条件的维护

● 软件的配置中只有源代码。

● 由于没有分析和设计文档，无法对程序的功能进行反向追踪，理解别人的代码是很痛苦的事情。

● 由于配置中没有测试文档，所以维护后的代码无法进行回归测试。因而导致程序的结构化被不断的破坏，维护的质量无法得到保证。

2．结构化维护是指满足如下条件的维护

● 待维护的软件的配置是完整的。

● 用户提出的维护申请用正向追踪很容易从分析设计文档追踪直至代码中，从而使维护人员很容易定位代码的维护点。所以这种维护不会破坏软件的结构。

● 结构化维护不仅能减少维护的工作量，还能提高维护的质量。

软件配置文档的重要性。

二应用题

1 判定表

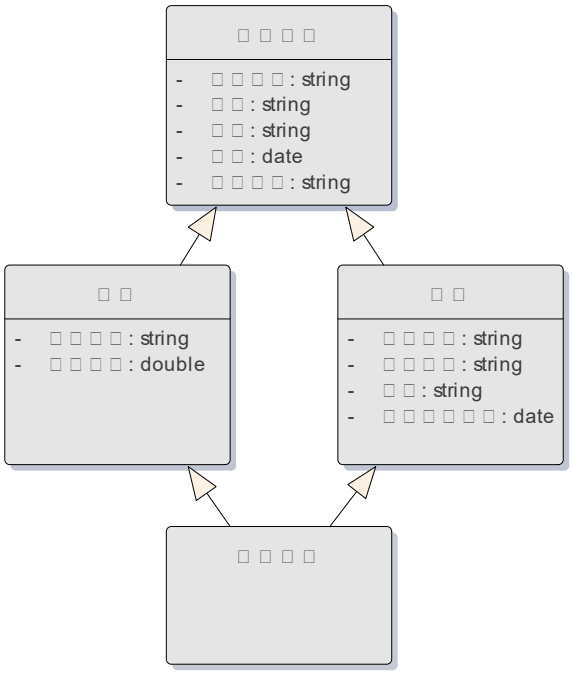
旅游价格折扣分类如下表，请用判定表和判定树写出表达该逻辑问题的算法；

旅游时间	7-9，12 月		1-6，10，11 月	
订票量	≤20	＞20	≤20	＞20
折扣量	5%	15%	20%	30%

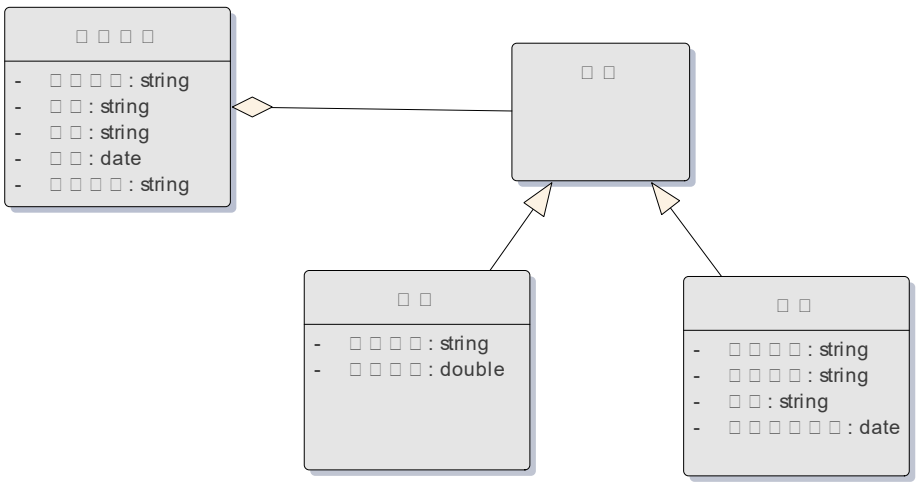
判定表

决策编号			1	2	3	4
条件	旅游时间=7-9，12 月		Y	N	Y	N
	订票量≤20		Y	Y	N	N
决策	折扣量	5%	X			
		15%			X	
		20%		X		
		30%				X

2 请将如下类图中的类调整为单继承的设计



调整结果：

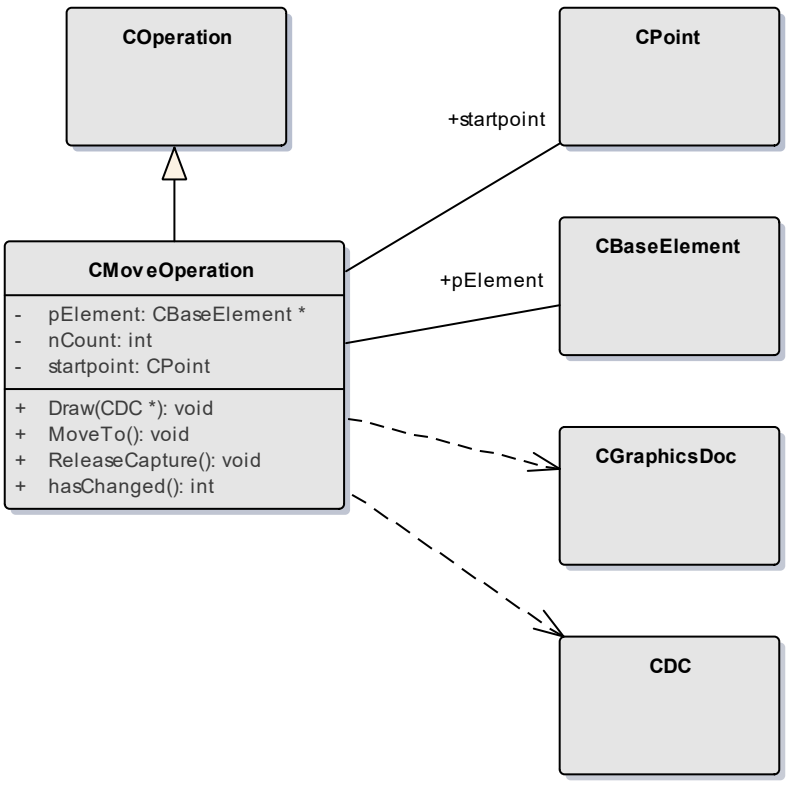


3 已知如下一个 C++类的定义，请分析这个类定义中引用了哪些类，并指出这些类之间的关系，用类图绘制出来。

已知如下一个 C++类的定义，请分析这个类定义中引用了那些类，并指出这些类之间的关系，用类图绘制出来。

```
Class CMoveOperation: public COperation
{
    CBaseElement*pElement;
    int nCount；
public:
    CMoveOperation(CGraphicsDoc*pDoc,HWNDhWnd);
    ~CMoveOperation(void);
private:
    CPoint StartPosition;
public:
    void Draw(CDC*pDC);
    void MoveTo(CPoint point);
    void ReleaseCapture();
    int hasChanged();
};
```

2 类图：



4 用例建模

1 假设一个商品销售网站系统的设计目标是通过网站来进行商品销售。该网站的用户分为有普通用户和注册用户和管理员三种。普通用户可以随时浏览网站并浏览网页查询全部商品信息，普通用户也可以随时注册为注册用户；注册用户可以登录浏览网页，也可以登录后进行购物。购物过程包括建立购物车、提交订单和在线支付等功能。管理员负责管理网站的商品信息、价格信息、订单管理、财务管理和销售统计等功能。

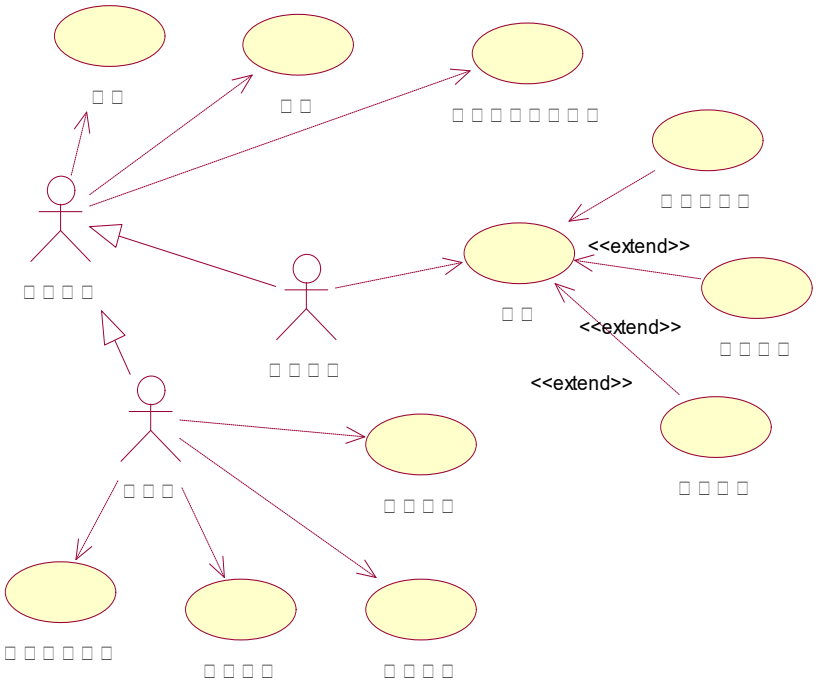
该网站系统对普通用户来讲主要具有浏览商品和购物两种功能。进入购物网页时要求用户必须进行登录，否则不能进入。

1）请根据上述陈述建立该系统的用例模型，并画出用例图。

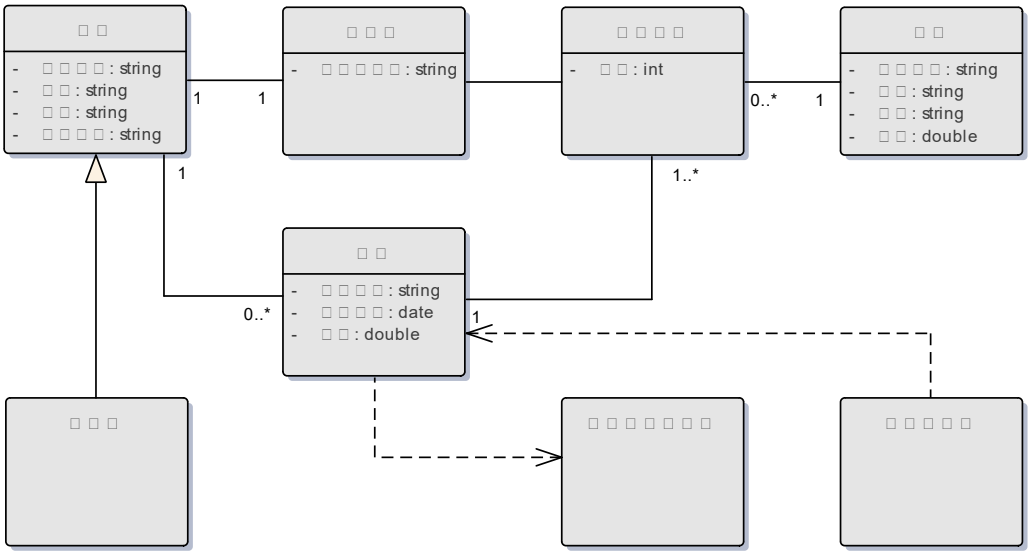
2）请分析上述系统可能需要那些类，并建立该系统的类图模型，即类和类之间的关系。

答案：

1 系统的用例模型



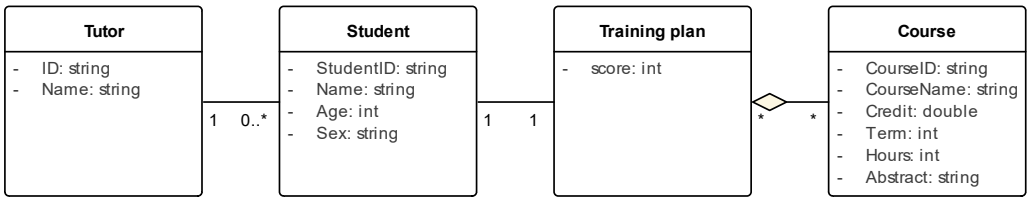
2 类图模型



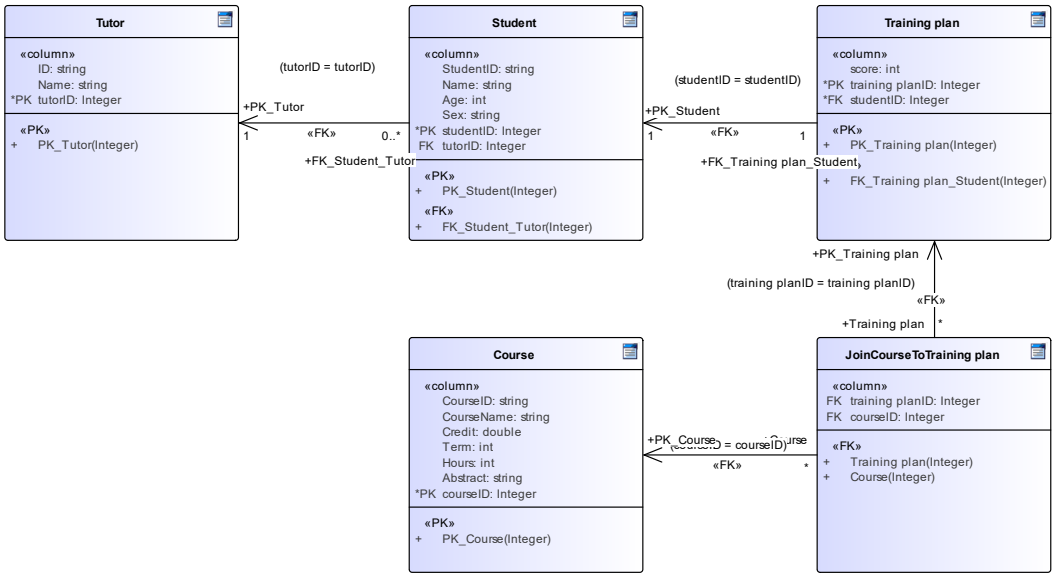
5 数据库设计

如下类图包含了导师(Tutor)、学生(student)、培养计划(TrainingPlan)和课程等四个类，以及它们之间的关系。

请将这个类图转换成相应的关系数据库逻辑模型。

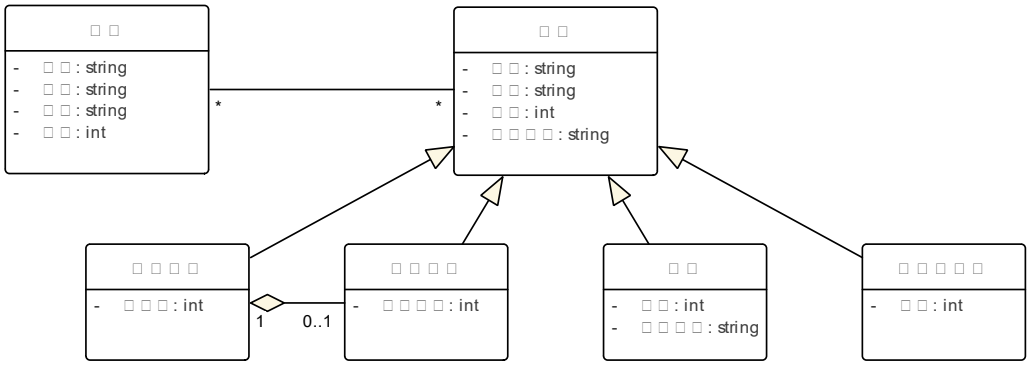


对应的数据关系图如下，请分析图中的各个图形元素以及转换方法。



6 数据库设计

某系统的类图模型如下图所示。假设学生与课程之间的关联还具有一个成绩属性。请将其转换成响应的关系数据库模型，并讨论你的设计方案的优缺点。



解：此题目中的关键问题是如何存储“学生与课程之间的关联、和各类课程的泛化问题”。

这可以根据对泛化的不同处理方式分成两大类方法。一种是将基类“课程“映射成一张表，同时将其每个派生类均映射成一张表。

另一种方式是，不将基类“课程“映射成表，而将每个派生类均映射成一张表，但这需要在每个派生类中维护课程与学生之间的关联关系，而使得系统变得更复杂。

下列答案给出了这种方式的映射结果。

1 • 学生表

序号	属性名	数据类型	是否主键	是否外键
1	学号	String（10）	是	
2	姓名	String（10）		
3	性别	String（10）		
4	年龄	Integer		

2 课程

序号	属性名	数据类型	是否主键	是否外键
1	课号	String（10）	是	否
2	名称	String（10）	否	否
3	学分	Decimal（10，2）	否	否
4	开课学期	Integer	否	否

3 学生-课程表

序号	属性名	数据类型	是否主键	是否外键
1	学号	String（10）	是	是，学生的主键
2	课号	String（10）	是	是，课程的主键
3	成绩	Double		

4 理论课程表

序号	属性名	数据类型	是否主键	是否外键
1	课号	String（10）	是	是，课程的主键
2	学时数	Integer		

5 课程设计表

序号	属性名	数据类型	是否主键	是否外键
1	课号	String（10）	是	是，课程的主键
2	课程编号	String（10）		是，参照理论课程表的主键
3	周数	Integer		

6 实训表

序号	属性名	数据类型	是否主键	是否外键
1	课号	String（10）	是	是，课程的主键
2	周数	Integer		
3	实训地点	String（10）		

7 新技术专题表

序号	属性名	数据类型	是否主键	是否外键
1	课号	String（10）	是	是，课程的主键
2	周数	Integer		

下图是自动转换得到的数据关系图模型，比较一下两者的联系和不同。