

# UML 习题答案

## 第一章 面向对象设计与 UML

### 1. 填空题

- (1) 基本构造块 UML 规则 公共机制
- (2) 名字 属性 操作
- (3) 封装 继承 多态
- (4) 继承
- (5) 瀑布模型 喷泉模型 基于组件的开发模型 XP 开发模型

### 2. 选择题

- (1) C
- (2) A B C D
- (3) A B C D
- (4) A B C
- (5) A

### 3. 简答题

(1) 类是具有相同或相似结构、操作和约束规则的对象组成的集合，而对象是某一类的具体化实例，每一个类都是具有某些共同特征的对象抽象。类与对象的关系就如模具和铸件的关系，类的实例化结果就是对象，而对一类对象的抽象就是类。类描述了一组有相同特性和相同行为的对象。

(2) 1.UML 是一种语言。 2.UML 是用来建模的。 3.UML 是统一的标准。

(3) 建模能够帮助我们按照实际情况或按我们需要的形式对系统进行可视化；提供一种详细说明系统的结构或行为的方法；给出一个指导系统构造的模板；对我们所做出的决策进行文档化。

在建模时应遵循以下原则：选择要创建什么模型对如何动手解决问题和如何形成解决方案有着意义深远的影响；每一种模型可以在不同的精度级别上表示；最好的模型是与现实相联系的；单个模型不充分；对每个重要的系统最好用一组几乎独立的模型去处理。

(4) UML 和面向对象软件开发之间有紧密的关系，可以说是面向对象软件开发促使了 UML 的产生。但是由于在 UML 标准化的过程中，吸收了业务建模、工作流建模和数据库建模等领域的标准规范，形成了适应性很强的标准。

(5) 在软件设计过程中，使用 UML 建模是为了能够更好地理解正在开发的系统。通过 UML 建模，可以达到以下目的：有助于按照实际情况或按照所需要的样式对系统进行可视化；能够规约系统的结构或行为；给出了指导构造系统的模板；对做出的决策进行文档化。

## 第二章 UML 通用知识点综述

### 1. 填空题

- (1) 依赖 泛化 关联 实现
- (2) 视图 图 模型元素
- (3) 实现视图 部署视图
- (4) 构造型 标记值 约束
- (5) 规格说明 修饰 通用划分

## 2. 选择题

- (1) D
- (2) C
- (3) A
- (4) A B
- (5) D

## 3. 简答题

(1) 在 UML 中，定义了四种基本的面向对象的事物，分别是结构事物、行为事物、分组事物和注释事物等。

(2) 构件种类有：源代码构件、二进制构件和可执行构件。

(3) 在 UML 中主要包括的视图为静态视图、用例视图、交互视图、实现视图、状态机视图、活动视图、部署视图和模型管理视图。

(4) 视图和图是包含和被包含的关系。在每一种视图中都包含一种或多种图。

(5) UML 提供了一些通用的公共机制，使用这些通用的公共机制（通用机制）能够使 UML 在各种图中添加适当的描述信息，从而完善 UML 的语义表达。通常，使用模型元素的基本功能不能够完善的表达所要描述的实际信息，这些通用机制可以有效地帮助表达，帮助我们进行有效的 UML 建模。UML 提供的这些通用机制，贯穿于整个建模过程的方方面面。前面我们提到，UML 的通用机制包括规格说明、修饰和通用划分三个方面。

## 第三章 Rational 统一过程

### 1. 填空题

- (1) 角色 活动 产物 workflow
- (2) 逻辑视图 过程视图 物理视图 开发视图 用例视图
- (3) 设计 开发 验证
- (4) 二维
- (5) 周期 迭代过程 里程碑

### 2. 选择题

- (1) A B C D
- (2) A C D
- (3) A C D
- (4) A B C
- (5) A B C D

### 3. 简答题

(1) 初始阶段、细化阶段、构造阶段和移交阶段。

(2) 迭代式软件开发、需求管理、基于构件的架构应用、建立可视化的软件模型、软件质量验证和软件变更控制。

(3) 在 Rational 统一过程的开发流程中，分别使用角色、活动、产物和 workflow 四种建模元素来进行表达。

(4) 对于一个以架构为中心的开发组织，需要对架构的目的、架构的表示和架构的过程进行关注。

(5) Rational 统一过程是 Rational 软件开发公司的一款软件产品，并且和 Rational 软件开发公司开发的一系列软件开发工具进行了紧密的集成。Rational 统一过程拥有自己的一套架构，并且这套架构是以一种大多数项目和开发组织都能够接受的形式存在的。它提供了如何对软

件开发组织进行管理的方式，并且拥有自己的目标和方法。

(6) 实现 Rational 统一过程的步骤：评估当前状态、建立明确目标、识别过程风险、计划过程实现、执行过程实现和评价过程实现。

(7) Rational 统一过程作为一种软件工程过程为开发组织提供了如何在开发过程中如何对软件开发的任务进行严格分配、如何对参与开发的人员职责进行严格的划分等方法。按照预先制定的计划，这些计划包括项目时间计划和经费预算，开发出高质量的软件产品，并且能够满足最终用户的要求。Rational 统一过程提供了一种以可预测的循环方式进行软件开发的过程、一个用来确保生产高质量软件的系统产品、一套能够被灵活改造和扩展的过程框架和许多软件开发最佳实践，这些都使 Rational 统一过程对现代软件工程的发展产生了深远的影响。

## 第四章 Rational Rose 的安装和操作

### 1. 填空题

- (1) 新建 打开 最近使用的模型
- (2) 用例视图 逻辑视图 构件视图 部署视图
- (3) 编辑区
- (4) 模型内容 日志记录
- (5) mdl

### 2. 选择题

- (1) A B C D
- (2) A C D
- (3) A B C
- (4) A C D
- (5) C

### 3. 简答题

(1) Rational Rose 是一个完全的、具有能满足所有建模环境（包括 Web 开发、数据库建模以及各种开发工具和语言）需求能力和灵活性的一套解决方案。Rational Rose 允许系统开发人员、系统管理人员和系统分析人员在软件的各个开发周期内，建立系统地需求和系统的体系架构的可视化模型，并且能够将这些需求和系统的体系架构可视化模型转换成代码，帮助系统开发。Rational Rose 建模工具能够为 UML 提供很好的支持。

(2) 从略，参照第二节。

(3) 通过选择“File”（文件）菜单栏下的“Import”（导入）可以用来导入模型、包或类等，可供选择的文件类型包含.mdl、.ptl、.sub 或.cat 等，导入模型的对话框。导入模型，可以对利用现成的建模。

通过选择“File”（文件）菜单栏下的“Export Model ...”（导出模型）可以用来导出模型，导出的文件后缀名为.ptl，当选择一个具体的类的时候，比如选择一个类名称为“User”，然后我们可以通过选择“File”（文件）菜单栏下的“Export User”（导出 User 类）来导出 User 类，弹出导出的文件后缀名称为.ptl

(4) Rational Rose 2003 的主界面主要是由标题栏、菜单栏、工具栏、工作区和状态栏构成。默认的工作区域包含四个部分，分别是左侧的浏览器，文档编辑区和右侧的图形编辑区域，以及下方的是日志记录。

## 第五章 使用 Rose 设计 UML

### 1. 填空题

- (1) 用例视图 逻辑视图 构件视图 部署视图
- (2) 非一致性检查 审查功能 重用建模元素和图功能
- (3) 构件 构件图 包
- (4) 部署
- (5) 模型代码库 执行文件 运行库 其他构件的信息

### 2. 选择题

- (1) A B D
- (2) A C D
- (3) A C D
- (4) A B C
- (5) A B C D

### 3. 简答题

(1) 使用 Rational Rose 建立的 Rose 模型中分别包括用例视图、逻辑视图、构件视图和部署视图。用例视图是系统功能的高层抽象。逻辑视图是系统如何实现用例中所描述的功能。部署视图显示的是系统的实际部署情况，它是为了便于理解系统如何在的一组处理节点上的物理分布，而在分析和设计中使用的构架视图。

(2) 使用 Rational Rose 生成代码有通过以下四个步骤进行：选择待转换的目标模型、检查语言的语法错误、设置代码生成属性、最后生产代码。

(3) 在 Rational Rose 中，可以通过收集有关类、类的属性、类的操作、类与类之间的关系以及包和构件等静态信息，将这些信息转化成为对应的模型，在相应的图中显示出来。

(4) 用例视图关注的是系统功能的高层抽象，适合于对系统进行分析和获取需求，而不关注于系统的具体实现方法。逻辑视图关注系统如何实现用例中所描述的功能，主要是对系统功能性需求提供支持，即在为用户提供服务方面，系统所应该提供的功能。

## 第六章 用例图

### 1. 填空题

- (1) 用例图
- (2) 参与者（角色） 用例 系统边界 关联
- (3) 包含 扩展 泛化
- (4) 用例粒度
- (5) 组成部分 系统外部

### 2. 选择题

- (1) D
- (2) A C D
- (3) C
- (4) C
- (5) D

### 3. 简答题

(1) 识别用例的最好方法就是从分析系统参与者开始，在这个过程中往往会发现新的参与者。当找到参与者之后，我们就可以根据参与者来确定系统的用例，主要是看各参与者如何使用系统，需要系统提供什么样的服务。对于这个被选出的用例模型，不仅要做到易于理解，

还要做到不同的涉众对于它的理解是一致的

(2) 我们可以在用例之间抽象出包含、扩展和泛化这三种关系。多个用例用到同一段的行为, 则可以把这段共同的行为单独抽象成为一个用例, 然后让其他用例来包含这一用例。扩展关系往往被用来处理异常或者构建灵活的系统框架。使用扩展关系可以降低系统的复杂度, 有利于系统的扩展, 提高系统的性能。扩展关系还可以用于处理基础用例中的那些不易描述的问题, 使系统显得更加清晰易于理解。当您发现系统中有两个或者多个用例在行为、结构和目的方面存在共性时, 就可以使用泛化关系。这时, 可以用一个新的(通常也是抽象的)用例来描述这些共有部分, 这个新的用例就是父用例。

(3) 视系统的复杂度决定。对于比较简单的系统, 可以相对用的少些用例图, 对于比较复杂的系统, 为表示清楚系统功能必须多创建用例图。我们应该根据每个系统的具体情况, 具体问题具体分析, 在尽可能保证整个用例模型的易理解性前提下决定用例的大小和数目。

(4) 用例图是从软件需求分析到最终实现的第一步, 它显示了系统的用户和用户希望提供的功能, 有利于用户和软件开发人员之间的沟通。借助于用例图, 系统用户、系统分析人员、系统设计人员、领域专家能够以可视化的方式对问题进行探讨, 减少了大量交流上的障碍, 便于对问题达成共识。

(5) 使用 Rose 创建用例图的步骤: 识别参与者、创建用例, 最后创建用例之间的关系。

#### 4. 上机题

(1) 用例图位于光盘中学生管理系统.mdl. ->User Case View->系统管理员用例图

(2) 用例图位于光盘中学生管理系统.mdl. ->User Case View->教师用例图

(3) 用例图位于光盘中学生管理系统.mdl. ->User Case View->学生用例

## 第七章 类图与对象图

### 1. 填空题

- (1) 对象 链
- (2) 依赖 泛化 关联 实现
- (3) 类
- (4) 类 接口 数据类型 构件
- (5) 共有类型 私有类型 受保护类型

### 2. 选择题

- (1) A B
- (2) C
- (3) D
- (4) D
- (5) C

### 3. 简答题

- (1) 类的组成元素有类的名称、类的属性、类的操作、类的职责、类的约束和类的注释。
- (2) 对象图是由对象和链组成的。
- (3) 在面向对象分析方法中, 类和对象的图形表示法是关键建模技术之一。它们能够有效的对业务领域和软件系统建立可视化的对象模型, 使用强大的表达能力来表示出面向对象模型的主要概念。UML 中的类图和对象图显示了系统的静态结构, 其中的类、对象是图形元素的基础。
- (4) 在类中包含三个部分, 分别是类名、类的属性和类的操作。类的名称栏只包含类名。类的属性栏定义了所有属性的特征。类中列出了操作类中使用了关联连接, 关联中使用名称、

角色以及约束等特征定义。类是一类的对象的抽象，类不存在多重性。

对象包含两个部分：对象的名称和对象的属性。对象的名称栏包含“对象名：类名”。对象的属性栏定义了属性的当前值。对象图中不包含操作内容，因为对属于同一个类的对象，其操作是相同的。对象使用链进行连接，链中包含名称、角色。对象可以具有多重性。

#### 4. 上机题

- (1) 类图位于光盘中学生管理系统.mdl. -> User Case View->类图
- (2) 对象图位于光盘中学生管理系统.mdl. -> Logical View->类图
- (3) 对象图位于光盘中学生管理系统.mdl. -> Logical View->类图
- 类图位于光盘中学生管理系统.mdl. -> User Case View->类图

### 第八章 序列图

#### 1. 填空题

- (1) 序列图
- (2) 发送者 接收者 活动
- (3) 激活
- (4) 生命线
- (5) 矩形框 下划线

#### 2. 选择题

- (1) A B C D
- (2) A
- (3) A C
- (4) B
- (5) A B C


#### 3. 简答题

(1) 序列图的用途包括：确认和丰富一个使用语境的逻辑表达。细化用例的表达。有效地描述如何分配各个类的职责以及各类具有相应职责的原因。

(2) 确定序列对象。创建对象。创建生命线。创建消息。销毁对象。

(3) 序列图显示不同的业务对象如何交互，对于交流当前业务如何进行很有用。除记录组织的当前事件外，一个业务级的序列图能被当作一个需求文件使用，为实现一个未来系统传递需求。在项目的需求阶段，分析师能通过提供一个更加正式层次的表达，把用例带入下一层次。那种情况下，用例常常被细化为一个或者更多的序列图。组织的技术人员也能通过序列图在记录一个未来系统的行为应该如何表现。在设计阶段，架构师和开发者能使用该图，挖掘出系统对象间的交互，这样充实整个系统设计

(4) 销毁对象表示对象生命线的结束，在对象生命线中使用一个“X”来进行标识。给对

象生命线上添加销毁标记的步骤如下：在序列图的图形编辑工具栏中选择  按钮，此时的光标变为“+”符号。单击欲销毁对象的生命线，此时该标记在对象生命线上标识。该对象生命线自销毁标记以下的部分消失。

#### 4. 上机题

- (1) 序列图位于光盘中学生管理系统.mdl->Logical View->输入学生信息 1（序列图）
- (2) 序列图位于光盘中学生管理系统.mdl->Logical View->输入学生信息（序列图）
- (3) 序列图位于光盘中学生管理系统.mdl->Logical View->修改学生信息（序列图）

## 第九章 协作图

### 1.填空题

- (1) 对象 协作关系中的链
- (2) 对象 对象间 进行交互
- (3) 独立连接 关联
- (4) 协作图
- (5) 消息

### 2.选择题

- (1) B
- (2) A B D
- (3) D
- (4) A B C
- (5) A

### 3. 简答题

(1) 协作图的作用：通过描绘对象之间消息的传递情况来反映具体的使用语境的逻辑表达。显示对象及其交互关系的空间组织结构。协作图的另外一个作用是表现一个类操作的实现。

(2) 对象、消息和链，这三个元素构成了协作图。对象的角色表示一个或一组对象在完成目标的过程中所应起的那部分作用。通过一系列的消息来描述系统的动态行为。链是两个或多个对象之间的独立连接，是关联的实例。

(3) 协作图中消息的种类有同步消息和异步消息。同步消息用于系统中使用多线程的场合。异步消息用于使用有并行的活动如消息队列的场合。

(4) 协作图和序列图都是交互图，它们既是等价的，又有所区别。顺序图表示了时间消息序列，但没有表示静态对象关系。顺序图可以有效的帮助我们观察系统的顺序行为。而协作图用于表示一个协同中的对象之间的关系和消息以及描述一个操作或分类符的实现。在对系统进行行为建模时，通常做法是用顺序图按时间顺序对控制流建模，用协作图按对象组织对控制流建模。

### 4.上机题

- (1) 协作图位于光盘中学生管理系统.mdl->Loginal View->输入学生信息 1（协作图）
- (2) 协作图位于光盘中学生管理系统.mdl->Loginal View->输入学生信息（协作图）
- (3) 协作图位于光盘中学生管理系统.mdl->Loginal View->修改学生信息（协作图）

## 第十章 活动图

### 1.填空题

- (1) 活动图
- (2) 一个对象流
- (3) 动作状态
- (4) 入口动作和出口动作
- (5) 泳道

### 2.选择题

- (1) A B D
- (2) A B D
- (3) C

(4) B

(5) A

### 3. 简答题

(1) 活动图描述一个操作执行过程中所完成的工，作它对活动图对用例描述尤其有用，它可建模用例的工作流，显示用例内部和用例之间的路径。活动图可以说明用例的实例是如何执行动作以及如何改变对象状态。它显示如何执行一组相关的动作，以及这些动作如何影响它们周围的对象。活动图对理解业务处理过程十分有用。

(2) UML 活动图中包含的图形元素有：动作状态、活动状态、组合状态、分叉与结合、分支与合并、泳道、对象流。

(3) 分叉用来表示将一个控制流分成两个或者多个并发运行的分支，分叉具有一个输入转换，两个或者多个输出转换，每个转换都可以是独立的控制流。分支是转换的一部分，它将转换路径分成多个部分，每一部分都有单独的监护条件和不同的结果。当动作流遇到分支时，会根据监护条件（布尔值）的真假来判定动作的流向。分支的每个路径的监护条件应该是互斥的，这样可以保证只有一条路径的转换被激发。

(4) 动作状态是原子性的动作或操作的执行状态，它不能被外部事件的转换中断。动作状态的原子性决定了动作状态要么不执行，要么就完全执行，不能中断。动作状态不能有入口动作和出口动作，也不能有内部转移。动作状态是一种特殊的活动状态。可以把动作状态理解为一种原子的活动状态。

活动状态是非原子性的，用来表示一个具有子结构的纯粹计算的执行。活动状态可以分解成其他子活动或动作状态，可以被使转换离开状态的事件从外部中断。活动状态可以有内部转换，可以有入口动作和出口动作。活动状态具有至少一个输出完成转换，当状态中的活动完成时该转换激发。

### 4. 上机题

(1) 活动图位于光盘中学生管理系统.mdl->Loginal View->State/Activity Model->学生登录

(2) 活动图位于光盘中学生管理系统.mdl-> User Case View-> State/Activity Model->学生登录

(3) 活动图位于光盘中学生管理系统.mdl->Loginal View->State/Activity Model->删除学生信息

## 第十一章 包图

### 1. 填空题

(1) 包 子系统 依赖关系

(2) 公共的 私有的 受保护的

(3) 模型元素 图

(4) 包图

(5) 模型

### 2. 选择题

(1) A

(2) A B

(3) B C D

(4) B C D

(5) A B C

### 3. 简答题



(1) 包和包之间的关系总的来讲可以概括为依赖关系和泛化。

(2) 包图是一种维护和描述系统总体结构的模型的重要建模工具，通过对图中各个包以及包之间关系的描述，展现出系统的模块与模块之间的依赖关系。包图是由包和包之间的联系构成的，它是维护和控制系统总体结构的重要工具。

(3) 构成包图的基本元素有包、子系统和依赖关系。包是一个可以拥有任何种类的模型元素的通用的命名空间。在系统模型中，每个图必须被一个唯一确定的包所有，同样这个包可能被另一个包所包含。包构成进行配置控制、存储和访问控制的基础。若干个相对独立的子系统构成了一个大型的复杂系统，系统和子系统的关系基本上是组合关系。通过对包图中各个包以及包之间关系的描述，展现出系统的模块与模块之间的依赖关系。

(4) 包是包图中最重要的概念，是最重要的组成部分。包图是由包和包之间的联系构成的，没有包就没有包图。

#### 4.上机题

(1) 包图位于光盘中学生管理系统.mdl->Loginal View->包图

### 第十二章 构件图和部署图

#### 1.填空题

- (1) 代码特征 身份特征
- (2) 构件图
- (3) 构件 构件 类和接口
- (4) 虚包
- (5) 部署图

#### 2.选择题

- (1) A B D
- (2) B
- (3) A B D
- (4) A
- (5) B

#### 3. 简答题

(1) 构件图适用于下列建模需求：系统源代码、系统的发布版本、物理数据库、自适应系统、用于建立业务模型和用于系统的开发管理等。

(2) 二者都有名称，都可以实现一组接口，都可以参与依赖关系，都可以被嵌套，都可以有实例，都可以参与交互。类描述了软件设计的逻辑组织和意图，而构件这描述了软件设计的物理实现，类可以有属性和操作，构件只有操作，只有通过构件使得接口才能使用。

(3) 在一张基本构件图中，构件之间最常见的关系是依赖关系和实现关系。

(4) 在 UML 中，构件主要包括配置构件、工作产品构件和可执行构件。

#### 4.上机题

(1) 构件图位于光盘中学生管理系统.mdl->Component View->构件图

(2) 部署图位于光盘中学生管理系统.mdl-> Deployment View

### 第十三章 状态图

#### 1.填空题

- (1) 对象

- (2) 状态图
- (3) 简单状态和组成状态
- (4) 历史状态
- (5) 事件

## 2. 选择题

- (1) A B C D
- (2) A B
- (3) A
- (4) A C D
- (5) A B C

## 3. 简答题

(1) 状态图是由状态、初始状态、终止状态、转换和判定这几个要素构成的

(2) 状态机是一种记录下给定时刻状态的设备，它可以根据各种不同的输入对每个给定的变化而改变其状态或引发一个动作。一个状态图(Statechart Diagram)本质上就是一个状态机，或者是状态机的特殊情况，它基本上是一个状态机中的元素的一个投影，这也就意味着状态图包括状态机的所有特征。状态图描述了一个实体基于事件反应的动态行为，显示了该实体如何根据当前所处的状态对不同的时间做出反应的。

(3) 在软件开发中使用状态图建模的作用：状态图清晰的描述了状态之间的转换顺序，通过状态的转换顺序也就可以清晰的看出事件的执行顺序；清晰的事件顺序有利于程序员在开发程序时避免出现事件错序的情况；状态图清晰的描述了状态转换时所必须的触发事件、监护条件和动作等影响转换的因素，有利于程序员避免程序中非法事件的进入；状态图通过判定可以更好的描述工作流因为不同的条件发生的分支。

(4) 顺序组成状态的多个子状态之间是互斥的，不能同时存在的。一个顺序组成状态最多可以有一个初始状态和一个终态。并发组成状态可以有两个或者多个并发的子状态，每个并发子状态还可以进一步分解为顺序组成状态。一个并发组成状态可以没有初始状态，终态。

## 4. 上机题

(1) 状态图位于位于光盘中学生管理系统.mdl->Loginal View-> State/Activity Model->学生信息

(2) 状态图位于位于光盘中学生管理系统.mdl->Loginal View-> State/Activity Model->系统管理员