

考点突破



根据考试大纲，本章要求考生掌握以下几个方面的知识点。

- (1) UML的基本概念与作用
- (2) 用例图的表示与应用
- (3) 类图与对象图的表示与应用
- (4) 序列图的表示与应用
- (5) 活动图的表示与应用
- (6) 通信图的表示与应用
- (7) 组件图的表示与应用
- (8) 部署图的表示与应用
- (9) 状态图的表示与应用

从历年的考试情况来看，本章的考点主要集中在这几种图的应用：用例图、类图与对象图、顺序图、活动图、状态图。

[版权方授权希赛网发布，侵权必究](#)

[上一节](#) [本书简介](#) [下一节](#)

考点精讲—UML基础知识

UML是一种与开发方法无关的建模语言，其应用十分广泛。本节将从它的起源、概念、组成部分等方面展开论述，最后将对各种常用的UML图进行详细解读。

1. UML的起源

前面的章节已经提到过软件开发方法有三种：结构化方法、面向对象方法、原型法。其中能应用于软件全生命周期的是：结构化方法与面向对象方法，原型法一般只用于需求分析阶段。

面向对象方法是在结构化设计方法出现很多问题的情况下应运而生的。从结构化设计的方法中，我们不难发现，结构化设计方法求解问题的基本策略是从功能的角度审视问题域。它将应用程序看成实现某些特定任务的功能模块，其中子过程是实现某项具体操作的底层功能模块。在每个功能模块中，用数据结构描述待处理数据的组织形式，用算法描述具体的操作过程。面对日趋复杂的应用系统，这种开发思路在以下几个方面逐渐暴露了一些弱点：审视问题域的视角、抽象级别、封装体、可重用性。这样就催生了一批面向对象方法，形成百家争鸣的局面，后来由Booch方法、OOSE、OMT三大主流OOA技术的创始人通过融合与整理，形成了新的标准——UML（统一建模语言）。目前，UML已经纳为国际标准，是软件系统建模的主要规范之一。

2.UML的组成

关于UML的组成，有很多人存在误解，误认为：“UML由一系列的UML图组成”，这种观点是错误的。UML由构造块、公共机制、规则三个部分组成，如图13-1所示。

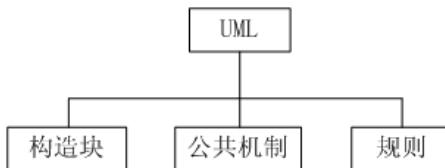


图13-1 UML结构示意图

(1) 构造块

构造块犹如建房子时的砖瓦，包括事物构造块、关系和图。

事物构造块：包括结构构造块（类、接口、协作、用例、活动类、构件、节点等）、行为构造块（交互、状态机）、分组构造块（包）、注释构造块。

关系：包括关联关系（包括表示整体-部分关系的聚合、组合关系）、依赖关系、泛化关系（表示一般/特别关系）、实现关系。

图：在UML 2.x中包括14种不同的图，分为表示系统静态结构的静态模型（包括对象图、类图、构件图、部署图、复合结构图、包图、制品图）；以及表示系统动态结构的动态模型（包括用例图、顺序图、协作图、状态图、活动图、定时图、交互概观图）。

(2) 规则

规则是支配基本构造块如何放在一起的规则。这些规则可以用于：

命名：也就是为事物、关系和图起名字；从语义的有效性而言，只要求是由字符、数字、下划线组成的唯一串；另外也要求是唯一的。

范围：使名字具有特定含义的语境。

可见性：用来表示编程语言中的public、private、protected等修饰符。

完整性：事物如何正确、一致地相互联系。

执行：运行或模拟动态模型的含义是什么。

UML对系统体系结构的定义是系统的组织结构，包括系统分解的组成部分、它们的关联性、交互、机制和指导原则，这些提供系统设计的信息。而具体来说，就是指5个系统视图，分别是逻辑视图、进程视图、实现视图、部署视图和用例视图。

逻辑视图：以问题域的语汇组成的类和对象集合。

进程视图：可执行线程和进程作为活动类的建模，它是逻辑视图的一次执行实例，描绘了所设计的并发与同步结构。

实现视图：对组成基于系统的物理代码的文件和构件进行建模。

部署视图：把构件部署到一组物理的、可计算的节点上，表示软件到硬件的映射及分布结构。

用例视图：最基本的需求分析模型。

(3) 公共机制

公共机制是指达到特定目标的公共UML方法，主要包括规格说明、修饰、公共分类和扩展机制四种。

规格说明：规格说明是元素语义的文本描述，它是模型真正的“肉”。

修饰：UML为每一个模型元素设置了一个简单的记号，还可以通过修饰来表达更多的信息。

公共分类：包括类元与实体（类元表示概念，而实体表示具体的实体）、接口和实现（接口用来定义契约，而实现就是具体的内容）两组公共分类。

扩展机制：包括约束（添加新规则来扩展元素的语义）、构造型（用于定义新的UML建模元素）、标记值（添加新的特殊信息来扩展模型元素的规格说明）。

3. UML中的图

UML中的图是需要掌握的重点，考试主要就是在考图的相关知识。下面是各种图的分类，以及基本功能，后面将继续深入探讨其中的一部分图。

表13-1 UML2.0的正式图形

类型	图名	功能
结构图 (静态图)	类图	描述类、类的特性以及类之间的关系
	对象图	描述一个时间点上系统中各个对象的一个快照
	复合结构图	描述类的运行时刻的分解
	构件图	描述构件的结构与连接
	部署图	描述在各个节点上的部署
	包图	描述编译时的层次结构
	制品图	描述了一组制品及它们之间的关系
行为图 (动态图)	用例图	描述用户与系统如何交互
	活动图	描述过程行为与并行行为
	状态机图	描述事件如何改变对象生命周期
	顺序图（序列图）	描述对象之间的交互，重点在强调顺序
	通信图（协作图）	描述对象之间的交互，重点在于连接
	定时图	描述对象之间的交互，重点在于定时
	交互概观图	是一种顺序图与活动图的混合

版权方授权希赛网发布，侵权必究

上一节 本书简介 下一节

第 13 章：UML建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月05日

考点精讲—用例图

首先值得强调的是：用例图在所有UML图中，重要程度是最高的。几乎在每次考试中都会考用例图，所以本节将对用例图进行详细论述。

在了解用例图的基本情况前，我们有必要先理解什么是用例。Ivar Jacobson是这样描述的：“用例实例是在系统中执行的一系列动作，这些动作将生成特定参与者可见的价值结果。一个用例定义一组用例实例。”

首先，从定义中得知用例是由一组用例实例组成的，用例实例也就是常说的“使用场景”，就是用户使用系统的一个实际的、特定的场景。其次，我们可以知道，用例应该给参与者带来可见的价值，这一点很关键。最后，我们得知，用例是在系统中的。通俗一点来理解，用例可以看成系统的功能，例如在结构化方法中，有登录这个功能，在使用UML建模时，“登录”将被构造为一个用例。

1. 用例图的概念

用例图（也可称用例建模）描述的是外部执行者（Actor）所理解的系统功能。用例图用于需求分析阶段，它的建立是系统开发者和用户反复讨论的结果，表明了开发者和用户对需求规格达成的共识。

在UML中，用例表示为一个椭圆。图13-2显示了一个图书管理系统的用例图。其中，“新增书

籍信息”、“查询书籍信息”、“修改书籍信息”、“登记外借情况”、“查询外借情况”、“统计金额与册数”等都是用例的实例。

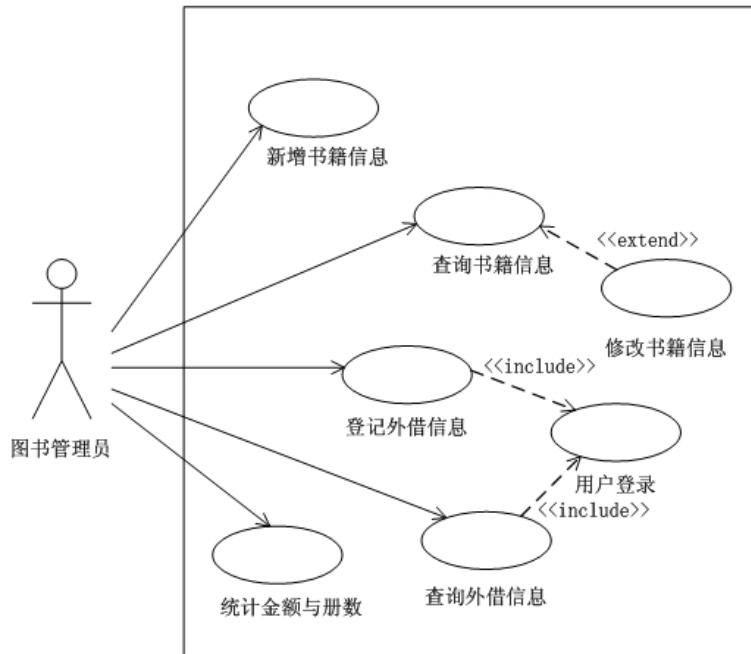


图13-2用例图示例

用例分析技术为软件需求规格化提供了一个基本的元素，而且该元素是可验证、可度量的。用例可以作为项目计划、进度控制、测试等环节的基础。而且用例还可以使开发团队与客户之间的交流更加顺畅。

2. 用例图的建立

用例图的建立通常要经历三个阶段：识别参与者、合并需求获得用例、细化用例描述。

(1) 识别参与者

参与者（Actor）是同系统交互的所有事物，它是指代表某一种特定功能的角色，因此参与者都是虚拟的概念。在UML中，用一个小人表示参与者。该角色不仅可以由人承担，还可以是其他系统、硬件设备、甚至是时钟。

其他系统：当你的系统需要与其他系统交互时，如在开发ATM柜员机系统时，银行后台系统就是一个参与者。

硬件设备：如果你的系统需要与硬件设备交互时，如在开发IC卡门禁系统时，IC卡读写器就是一个参与者。

时钟：当你的系统需要定时触发时，时钟就是一个参与者，如在开发Foxmail这样的电子邮件应用软件中的“定时自动接收”功能时，就需要引入时钟作为参与者。

要注意的是，参与者一定在系统之外，不是系统的一部分。通常可以通过以下问题来整理思路：谁使用这个系统？谁安装这个系统？谁启动这个系统？谁维护这个系统？谁关闭这个系统？哪些其他的系统使用这个系统？谁从这个系统获取信息？谁为这个系统提供信息？是否有事情自动在预计的时间发生？

(2) 合并需求获得用例

用例是对系统行为的动态描述，它可以促进设计人员、开发人员与用户的沟通，理解正确的需求，还可以划分系统与外部实体的界限，是系统设计的起点。在识别出参与者之后，你可以使用下列问题帮助你识别用例：每个参与者的任务是什么？有参与者将要创建、存储、修改、删除或读取系统中的信息吗？什么用例会创建、存储、修改、删除或读取这个信息吗？参与者需要通知系统外

部的突然变化吗？需要通知参与者系统中正在发生的事情吗？什么用例将支持和维护系统？所有的功能需求都对应到用例了吗？系统需要何种输入输出？输入从何处来？输出到何处？当前运行系统的主要问题是什么？

通常情况下，我们将在参与者都找到之后通过合并需求来获得用例。也就是仔细地检查参与者，为每一个参与者确定用例。而其中的依据主要可以来源于已经获取得到的“特征表”。

将特征分配给相应的参与者：首先，要将这些捕获到的特征，分配给与其相关的参与者，以便可以针对每一个参与者进行工作，而无遗漏。

进行合并操作：在合并之前，我们首先还要明确为什么要合并，知道了合并的目的，也就会让我们选择正确的合并操作。一个用例就是一个对参与者来说可见的价值结果，因此合并的根据就是使得其能够组合成为一个可见的价值结果。合并后，将产生用例，而用例的命名应该注意采用“动词（短语）+名词（短语）”的形式，而且最好能够对其进行编号，这也是实现跟踪管理的重要技巧，通过编号可以将用户的需求落实到特定的用例中去。

绘制成用例图：最后我们就将识别到的参与者，以及合并生成的用例通过用例图（的形式整理出来。在此，经常需要使用包含和扩展关系来描述用例之间的关系，该内容在后面会详细叙述。

（3）细化用例描述

经历了前两步，用例图已经完成，大家或许认为用例建模已“大功告成”，但事实并非如此。仅有用例图是不够的，因为这个图只能勾勒出一个大致的系统功能轮廓，系统很多细节信息都没有明确地表示出来。所以对于每个用例而言，我们还需要编写它的事件流。

就像类对应于对象一样，一个用例的实例就是一个使用场景，用例就是对使用场景进行抽象的总结，形成一组事件流。整个事件流的描述主线如图13-3所示。

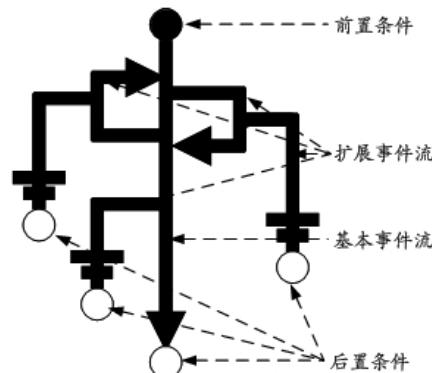


图13-3事件流模型

前置条件：指在用例启动时，参与者（Actor）与系统应置于什么状态，这个状态应该是系统能够检测到的、可观测的；

后置条件：用例结束时，系统应置于什么状态，这个状态也应该是系统能够检测得到的、可观测的；

基本事件流：基本事件流是对用例中常规、预期路径的描述，也被称为Happy day场景，这时大部分时间所遇到的场景；它将体现系统的核心价值；

扩展事件流：主要是对一些异常情况、选择分支进行描述。

进行细化用例描述时，通常有两个步骤，首先将事件流的基本框架完成（类似于写书时，先写目录）。然后再对每个阶段要完成的具体工作描述出来。以图13-2所述的图书管理系统为例，若要完成“新增书籍信息”用例的事件流，可以先完成基本框架，结果如下所示。

<p>1.用例名称： 新增书籍信息 (uc01)</p> <p>2.简要说明： 录入新购书籍信息，并自动存储建档。</p> <p>3.事件流：</p> <ul style="list-style-type: none"> 3.1 基本事件流 3.2 扩展事件流 <p>4.非功能需求</p> <p>5.前置条件 用户进入图书管理系统。</p> <p>6.后置条件 完成新书信息的存储建档。</p> <p>7.扩展点 无</p> <p>8.优先级 最高 (满意度 5, 不满意度 5)</p>

然后对这基本框架进行细化，结果如下所示。

<p>.....</p> <p>3.事件流：</p> <p>3.1 基本事件流</p> <ol style="list-style-type: none"> 1) 图书管理员向系统发出“新增书籍信息”请求； 2) 系统要求图书管理员选择要新增的书籍是计算机类还是非计算机类； 3) 图书管理员做出选择后，显示相应界面，让图书管理员输入信息，并自动根据书号规则生成书号； 4) 图书管理员输入书籍的相关信息，包括：书名、作者、出版社、ISBN号、开本、页数、定价、是否有CDROM； 5) 系统确认输入的信息中书名未有重名； 6) 系统将所输入的信息存储建档。 <p>3.2 扩展事件流</p> <ul style="list-style-type: none"> a) 如果输入的书名有重名现象，则显示出重名的书籍，并要求图书管理员选择修改书名或取消输入； a1) 图书管理员选择取消输入，则结束用例，不做存储建档工作； a2) 图书管理员选择修改书名后，转到 5)。 <p>4.非功能需求 无特殊要求</p> <p>.....</p>

3. 包含、扩展与泛化

用例图中常见的关系有：包含、扩展与泛化，其中包含与扩展是用例图中特有的关系，而泛化关系不仅用于用例图，同时也适用于其它图，如类图。

用例之间的包含和扩展关系是分解和组织用例的有效工具，表面上看它们有许多相似之处，因此很多初学者经常容易混淆。下面将详细介绍这些关系的特点以及区别。

(1) 包含关系

当可以从两个或两个以上的用例中提取公共行为时，应该使用包含关系来表示它们。其中这个提取出来的公用例称为抽象用例，而把原始用例称为基本用例或基础用例。例如，图13-2中的“登记外借信息”和“查询外借信息”两个用例都需要登录，为此，可以定义一个抽象用例“用户登录”。用例“登记外借信息”和“查询外借信息”与用例“用户登录”之间的关系就是包含关系，如图13-4所示。其中“<<include>>”是包含关系的构造型，箭头指向抽象用例。

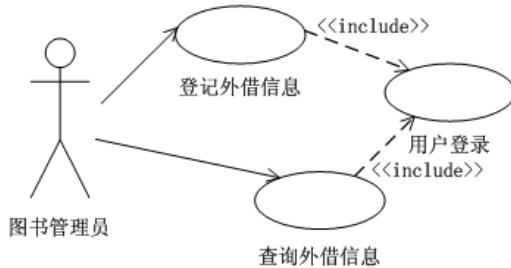


图13-4包含关系的例子

当多个用例需要使用同一段事件流时，抽象成为公共用例，可以避免在多个用例中重复地描述这段事件流，也可以防止这段事件流在不同用例中的描述出现不一致。当需要修改这段公共的需求时，也只要修改一个用例，避免同时修改多个用例而产生的不一致性和重复性工作。另外，当某个用例的事件流过于复杂时，为了简化用例的描述，也可以将某一段事件流抽象成为一个被包含的用例。

(2) 扩展关系。如果一个用例明显地混合了两种或两种以上不同的场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样使描述可能更加清晰。例如，图13-5中的图书管理员进行“查询书籍信息”操作时，如果发现书籍信息有误，他可以使用“修改书籍信息”用例来完成错误的修改。所以用例“查询书籍信息”和“修改书籍信息”之间的关系就是扩展关系，如图13-5所示。其中“<<extend>>”是扩展关系的构造型，箭头指向基本用例。

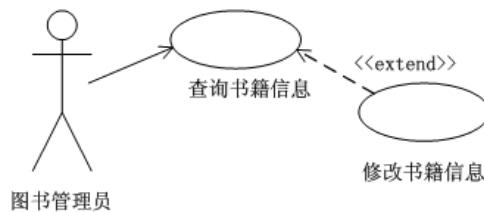


图13-5扩展关系的例子

(3) 泛化关系。当多个用例共同拥有一种类似的结构和行为的时候，可以将它们的共性抽象成为父用例，其他的用例作为泛化关系中的子用例。在用例的泛化关系中，子用例是父用例的一种特殊形式，子用例继承了父用例所有的结构、行为和关系。例如，图书管理系统中，用户注册时有多种方式，可以是“现场注册”，也可以是“网上注册”。所以“用户注册”用例就是“现场注册”用例和“网上注册”用例的泛化，如图13-6所示。其中三角箭头指向父用例。

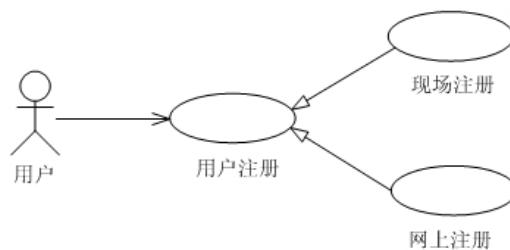


图13-6泛化关系的例子

从UML事物关系的本质上来看，包含关系和扩展关系都属于依赖关系。对包含关系而言，抽象用例中的事件流是一定插入到基本用例中去的，并且插入点只有一个。扩展用例的事件流往往可以抽象为基本用例的备选事件流，在扩展关系中，可以根据一定的条件来决定是否将扩展用例的事件流插入到基本用例的事件流中，并且插入点可以有多个。在实际应用中，很少使用泛化关系，子用例的特殊行为都可以作为父用例中的备选事件流而存在。

在实际工作中，要谨慎选用这些关系。从上面的介绍可以看出，包含、扩展和泛化关系都会增

加用例的个数，从而增加用例模型的复杂度。另外，一般都是在用例模型完成之后才对它进行调整，在用例模型建立之初不必急于抽象用例之间的关系。

版权方授权希赛网发布，侵权必究

上一节 本书简介 下一节

第 13 章：UML 建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月05日

考点精讲—类图与对象图

在面向对象建模技术中，我们将客观世界的实体映射为对象，并归纳成类。类（Class）、对象（Object）和它们之间的关联是面向对象技术中最基本的元素。对于一个想要描述的系统，其类模型和对象模型揭示了系统的结构。在UML中，类和对象模型分别由类图和对象图表示。类图技术是OO方法的核心。下面我们来总结一下类模型中的常见考点。

1. 发现/识别类

发现类的方法有很多种，其中最广泛应用的莫过于“名词动词法”。它的主要规则是从名词与名词短语中提取对象与属性；从动词与动词短语中提取操作与关联；而所有格短语通常表明名词应该是属性而不是对象。不过在使用“名词动词法”寻找类的时候，容易耗费大量的时间，并且容易迷失方向。要谨记，在此我们的主要目的是对问题域建立概要的了解，而非咬文嚼字。

找到备选类：首先，我们可以逐字逐句地阅读需求描述，列出所有名词及名词短语，就可以得到备选类列表。

决定候选类：很显然，并不是每一个备选类都是合适候选类，有些名词对于要开发的系统来说无关紧要，甚至是系统之外的；而有些名词表述的概念则相对较小，适合于某个候选类的属性。因此，我们需要对备选类进行一番筛选，将那些不合适的排除掉。

假设小王希望你帮助他开发一个简单的图书管理系统，经过你们一些简单的沟通后，得出了一些简要的需求描述。

小王是一个爱书之人，家里各类书籍已过千册，而平时又时常有朋友外借，因此需要一个图书管理系统。该系统应该能够将书籍的基本信息按计算机类、非计算机类分别建档，实现按书名、作者、类别、出版社等关键字的组合查询功能。在使用该系统录入新书籍时系统会自动按规则生成书号，可以修改信息，但一经创建就不允许删除。该系统还应该能够对书籍的外借情况进行记录，可对外借情况列表打印。另外，还希望能够对书籍的购买金额、册数按特定时间周期进行统计。

首先，我们可以逐字逐句地阅读上面那段需求描述，并将其中的所有名词及名词短语列出来，我们可以得到如下的备选类列表。

笔者人家里书籍朋友
图书管理系统 基本信息（书籍的）计算机类非计算机
书名作者类别出版社关键字功能
新书籍系统规则书号信息记录
外借情况外借情况列表购买金额册数特定时限

很显然，并不是每一个备选类都是合适候选类，有些名词对于要开发的系统来说并无紧要，甚至是系统之外的；而有些名词表述的概念则相对较小，适合于某个候选类的属性。因此，我们需要对备选类进行一番筛选，将这些不合适的排除掉。

“笔者”、“人”、“家里”很明显是系统外的概念，无须对其建模；而“图书管理系统”、“系统”指的就是将要开发的系统，即系统本身，也无须对其进行建模；

很明显“书籍”是一个很重要的类，而“书名”、“作者”、“类别”、“出版社”、“书号”则都是用来描述书籍的基本信息的，因此应该作为“书籍”类的属性处理，而“规则”是指书号的生成规则，而书号则是书籍的一个属性，因此“规则”可以作为编写“书籍”类构造函数的指南；

“基本信息”则是书名、作者、类别等描述书籍的基本信息统称，“关键字”则是代表其中之一，因此无须对其进行建模；

“功能”、“新书籍”、“信息”、“记录”都是在描述需求时使用到的一些相关词语，并不是问题域的本质，因此先可以将其淘汰掉；

“计算机类”、“非计算机类”是该系统中图书的两大分类，因此应该对其进行建模，并改名为“计算机类书籍”和“非计算机类书籍”，以减少歧义；

“外借情况”则是用来表示一次借阅行为，应该成为一个候选类，多个外借情况将组成“外借情况列表”，而外借情况中一个很重要的角色是“朋友”——借阅主体。虽然到本系统中并不需要建立“朋友”的资料库，但考虑到可能会需要列出某个朋友的借阅情况，因此还是将其列为候选类。为了能够更好地表述，将“外借情况”改名为“借阅记录”，而将“外借情况列表”改名为“借阅记录列表”；

“购买金额”、“册数”都是统计的结果，都是一个数字，因此不用将其建模，而“特定时限”则是统计的范围，也无须将其建模；不过从这里的分析中，我们可以发现，在该需求描述中隐藏着一个关键类——书籍列表，也就是执行统计的主体。

通过以上的分析，我们就可以得到一个候选类列表：

书籍
计算机类书籍
非计算机类书籍
借阅记录
借阅记录列表
书籍列表

当然，这样的过程通常是在脑子中完成的，在此详细地分析只是帮助大家更好地理解这一概念。

2. 确定类之间的关系

在建立抽象模型时，我们会发现很少有类会单独存在，大多数都将会以某种方式彼此协作，因此我们还需要描述这些类之间的关系。关系是事物间的连接，在面向对象建模中，有4种很重要的关系。

依赖关系

有两个元素X、Y，如果修改元素X的定义可能会引起对另一个元素Y的定义的修改，则称元素Y依赖（Dependency）于元素X。在UML中，使用带箭头的虚线表示依赖关系，如图13-7所示。

图13-7 依赖关系的图示

在类中，依赖由各种原因引起，例如：一个类向另一个类发消息；一个类是另一个类的数据成员；一个类是另一个类的某个操作参数。如果一个类的接口改变，它发出的任何消息可能不再合法。

依赖关系是一种很泛的关系，很多关系都属于依赖的范畴，将依赖关系进一步细化可得到表13-2。

表13-2 UML模型的各种依赖关系的含义

依赖关系	含义
访问(access)	引入另一外包的内容
绑定(bind)	为模板参数指定值，以生成一个新的模型元素
调用(call)	表明一个类的方法调用其他类的操作
创建(create)	表明一个类创造另一个类的实例
派生(derive)	表明一个实例可以从另一个实例中计算而得
实例化(instantiate)	一个类的方法创建了另一个类的实例
许可(permit)	允许一个元素使用另一个元素的内容
实现(realize)	说明和具体实现之间的映射
精化(refine)	两个不同语义层次上的元素之间的映射
发送(send)	信号发送者和信号接收者之间的关系
替代(substitute)	表明源类支持目标类的接口，可以替代目标类
跟踪依赖(trace)	不同模型中的元素之间存在一些连接，但不如映射精确
使用(use)	一个模型元素需要另一个模型元素的存在，才能够正确实现功能(包括调用、创建、实例化、发送以及其他可能的依赖)

泛化关系

泛化关系在用例图部分已经提及，该关系的应用范围较广，不仅可以用在用例图中，还可以在类图，对象图中使用。类图中的泛化关系描述了一般事物与该事物中的特殊种类之间的关系，也就是父类与子类之间的关系。继承关系是泛化关系的反关系，也就是说子类是从父类继承的，而父类则是子类的泛化。在UML中，使用带空心箭头的实线表示泛化关系，箭头指向父类，如图13-8所示。



图13-8泛化关系的图示

在UML中，对泛化关系有3个要求。

子类应与父类完全一致，父类所具有的关联、属性和操作，子元素都应具有。

子类中除了与父类一致的信息外，还包括额外的信息。

可以使用父类实例的地方，也可以使用子类实例。

例如，我们称“自行车”和“小轿车”从“交通工具”继承，而“交通工具”是“自行车”和“小轿车”的泛化，如图13-9所示。

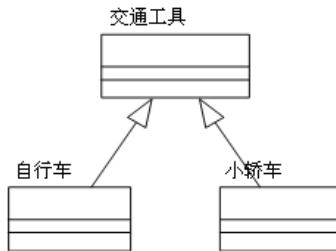


图13-9泛化关系的实例

关联关系

关联 (Association) 表示两个类之间存在某种语义上的联系。例如，一个人为一家公司工作，一家公司有许多办公室。我们就认为人和公司、公司和办公室之间存在某种语义上的联系。

关联关系提供了通信的路径，它是所有关系中最通用、语义最弱的。在UML中，使用一条实线来表示关联关系。而在关联关系中，有两种比较特殊的关系：聚合和组合。

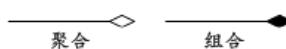


图13-10聚合与组合关系表示法

聚合关系：聚合（Aggregation）是一种特殊形式的关联。聚合表示类之间的关系是整体与部分的关系。聚合关系的含义是“聚”在一起的意义，也就是表示“部分”可以独立于“整体”而存在。在UML模型中，使用一个带空心菱形的实线表示，空心菱形指向的是代表“整体”的类，如图13-10所示。

组合关系：如果发现“部分”类的存在，是完全依赖于“整体”类的，那么就应该使用“组合”关系来描述。在UML模型中，组合关系是使用带有实心菱形的实线表示，实心菱形指向的是代表“整体”的类。

那么到底聚合与组合的区别在什么地方呢？许多书籍虽然举过很多例子，但是都忽略了，这种例子是必须依赖于“应用场景”的。也就是要根据应用场景来判断部分类和整体类之间的关系。例如：“电脑”是一个整体类，而“主板”、“CPU”……则是相对于它的部分类。那么它们之间应该整体类还是部分类呢？如果你是在固定资产管理系统中，可能适合的就是“组合”，甚至只是“电脑”类的属性；而如果对于在线DIY的系统，那么显然应该采用“聚合”关系。对于组合而言，最易于理解的例子是“订单”与“订单项”之间的关系，如果订单不存在，显然订单项也就没有意义了，因此必然是组合关系。

原则：判断是聚合还是组合关系，关键在于要放到具体的应用场景中讨论。

实现关系

实现关系是用来规定接口和实现接口的类或组件之间的关系。接口是操作的集合，这些操作用于规定类或组件的服务。在UML中，使用一个带空心箭头的虚线表示，如图13-11所示。



图13-11实现关系的图示

以上的几种关系是十分重要的，应对其概念与意义熟练的掌握。

实例分析

对于上面的例子中，很明显可以发现“计算机类书籍（ItBook）”、“非计算机类书籍（OtherBook）”与“书籍（Book）”之间是继承关系。而“书籍列表（BookList）”则是由多个“书籍”组成的，“借阅记录列表（BorrowList）”是由多条“借阅记录（BorrowRecord）”组成而成的。这种组成关系适用于组合还是聚合关系呢？显然，由于在本系统中“书籍”是可以独立于“书籍列表”存在的，“借阅记录”也是可以独立于“借阅记录列表”而存在的，因此使用聚合更合适一些。另外，我还可以发现“借阅记录”是与“书籍”关联的，离开“书籍”，“借阅记录”将失去意义。

为了反映和记录这些类之间的关联关系，就可以使用UML中的类图将其记录下来，如图13-12所示。

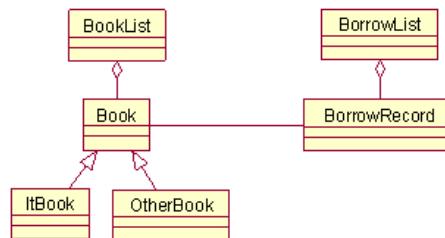


图13-12最初的类模型

3. 多重性关系

重复度（Multiplicity）又称多重性，多重性表示为一个整数范围n..m，整数n定义所连接的最

少对象的数目，而m则为最多对象数（当不知道确切的最大数时，最大数用*号表示）。最常见的多重性有：0..1；0..*（也可以表示为0..n）；1..1；1..*（也可以表示为1..n）；*（也可以表示为n）。

多重性是用来说明关联的两个类之间的数量关系，例如：

书与借书记录之间的关系，就应该是1对0..1的关系，也就是一本书可以有0个或1个借书记录。

经理与员工之间的关系，则应为1对0..*的关系，也就是一个经理可以领导0个或多个员工。

学生与选修课程之间的关系，就可以表示为0..*对1..*的关系，也就是一个学生可以选择1门或多门课程，而一门课程有0个或多个学生选修。

在解答这类题目时，最关键的是要根据题意来理解它们之间的多重性关系，而不是从概念上，因为多重性是描述类之间的关联的，而类则是对现实对象的抽象。下面的图13-13就是在图13-12所示类图的基础上添加多重性信息的结果。

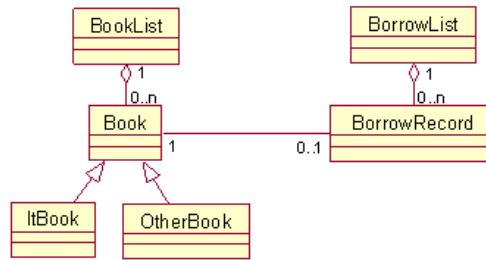


图13-13个人藏书管理类模型

由于该系统是应用于个人藏书管理，因此每本书都是唯一的，没有副本，即其要么被借出去，要么未被借出，也就是说，对于每一本书籍来说，要么没有借阅记录，要么也只有一条借阅记录。

所有的书籍组成书籍列表，借阅记录列表则也是由所有的借阅记录组成的。

4. 为类添加职责

当我们找到了反映问题域本质的主要概念类，而且理清了它们之间的协作关系之后，我们就可以为这些类添加其相应的职责。什么是类的职责呢？它包括两方面的内容：一是类所维护的知识；二是类能够执行的行为。相信大家从上面的两句中，马上会想到类的成员变量（也称为属性）和成员方法吧！是的，成员变量就是其所维护的知识，成员方法就是其能够执行的行为。

属性是用来描述一类对象的特性的值，它通常是名词或名词短语，因此也常使用名词动词法进行分析，先从备选类中选出候选类，然后再从被淘汰的备选类中进行选择。

方法则是该对象能够完成的行为与功能，它通常表现为动词或动词短语。

对于我们前面分析的那个例子而言，可以继续根据需求描述的内容，以及与客户的简单沟通将主要类的主要成员变量和成员方法标识出来，以便更好的理解问题域。

书籍类：从需求描述中，我们已经找到了描述书籍的几个关键成员变量，即书名、类别、作者、出版社；同时从统计的需要中，我们可以得知“定价”也是一个关键的成员变量。

书籍列表类：书籍列表就是全部的藏书列表，对于该类而言其主要的成员方法是新增、修改、查询（按关键字查询）、统计（按特定时限统计册数与金额）。

借阅记录类：而针对“借阅记录”这个类，其关键的成员变量也一目了然，即借阅人（朋友）、借阅时间。

借阅记录列表类：这也就是所有的借阅记录，因此其主要职责就是添加记录（借出）、删除记录（归还）以及打印借阅记录。

通过上面的分析，我们对这些概念类的了解更加深入，可以重新修改类图，将这些信息加入原先的模型，得到如图13-14所示的结果。

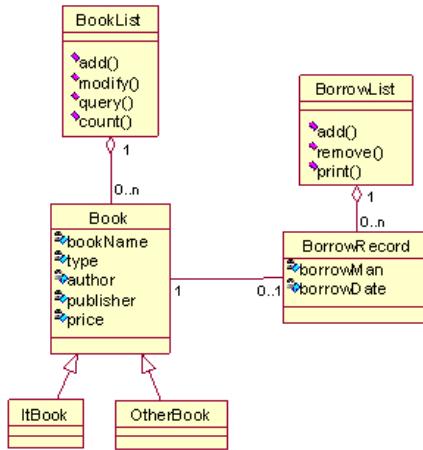


图13-14加入职责信息

下面，我们则通过一个例子来说明如何为类添加属性，如图13-15所示。

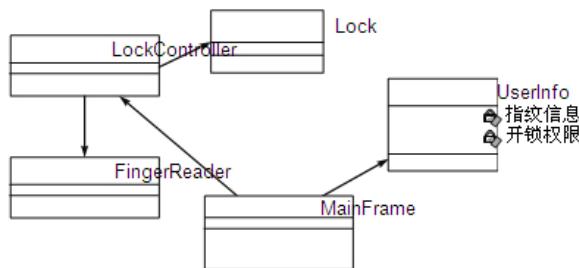


图13-15某门禁系统的类模型

系统描述：

某指纹门禁系统包括四个主要部件：主机、锁控器、指纹采集器和电控锁。

- 1) 系统中的每个电控锁都有一个唯一的编号。锁的状态有两种：“已锁住”和“未锁住”。
- 2) 主机上可以设置每把锁的安全级别及用户的开锁权限。只有当用户的开锁权限大于或等于锁的安全级别并且处于“已锁住”状态时，才能将锁打开。
- 3) 用户的指纹信息、开锁权限及锁的安全级别都保存在主机的数据库中。
- 4) 用户开锁时，只需按一下指纹采集器。指纹采集器将发送一个中断事件给锁控器，锁控器从指纹采集器读取用户的指纹并将指纹信息发送到主机，主机根据数据库中存储的信息来判断用户是否具有开锁权限，若有且锁当前处于“已锁住”状态，则将锁打开；否则系统报警。

首先，我们可以列出所有的名词及名词短语：

电控锁	编号	状态	已锁住	未锁住	主机
安全级别	用户	开锁权限	指纹信息	数据库	指纹采集器
中断事件	锁控器				

显然，带有下划线的是类名（从如图13-15所示的类图中可以看出），因此剩下的就是备选的属性。我们可以逐一地按上下文进行分析，如表13-3所示。

表13-3 备选属性

备选属性	上下文	结果
编号	每个电控锁都有一个唯一的编号	电控锁的属性
状态	锁的状态有两种	电控锁的属性
已锁住	它是属性的一种状态	应是属性的取值范围
未锁住	它是属性的一种状态	应是属性的取值范围
安全级别	主机上可以设置每把锁的安全级别	电控锁的属性
开锁权限	用户的开锁权限	用户的属性
指纹信息	用户的指纹信息	用户的属性
数据库	主机的设施	不是属性
中断事件	指纹采集器将发送一个中断事件给锁控器	不是属性，是系统机制

当然，在具体的应用中，还需要有更多的素材，获得更完整的属性信息，这样才能更有效地对

现实世界进行抽象。

5. 对象图

UML中对象图与类图具有相同的表示形式。对象图可以看做是类图的一个实例。对象是类的实例；对象之间的链（Link）是类之间的关联的实例。对象与类的图形表示相似，均为划分成三个格子的长方形（下面的两个格子可省略）。最上面的格子是对象名，对象名带有下划线；中间的格子记录属性值。链的图形表示与关联相似。对象图常用于表示复杂类图的一个实例。

版权方授权希赛网发布，侵权必究

上一节 本书简介 下一节

第 13 章：UML 建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月05日

考点精讲—顺序图

顺序图 (Sequence Diagram) 用来描述对象之间动态的交互关系，着重体现对象间消息传递的时间顺序。顺序图允许直观地表示出对象的生存期，在生存期内，对象可以对输入消息做出响应，并且可以发送信息。如图13-16就是一个顺序图，图中标识出了每个部分的名称。

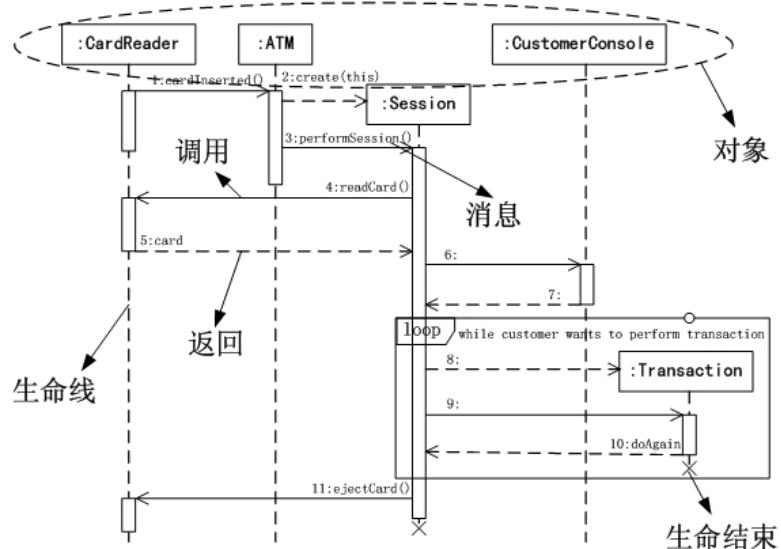


图13-16顺序图示意图

在考试中，对顺序图的考查形式主要是给出系统描述，要求考生根据描述填充消息名称。解答这类试题，需要以试题说明为主线来分析操作步骤然后按时间顺序与试题给出的图进行匹配，“对号入座”。在后面关于典型试题分析的章节会详细说明实战中的解题技巧。

版权方授权希赛网发布，侵权必究

上一节 本书简介 下一节

第 13 章：UML 建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月05日

考点精讲—活动图

类模型体现了系统的静态结构，用实例模型则从用户的角度对系统的动态行为进行了宏观建模，并通过交互模型将对象与消息有机地结合在一起。但有些时候，我们还需要更好地表示行为的细节，这就可以借助于活动图和状态图来实现。

1. 活动图基础

图13-17展示了一个用户订单处理过程的流程图，我们接下来就结合这个基本的活动图来学会正确的阅读方法。

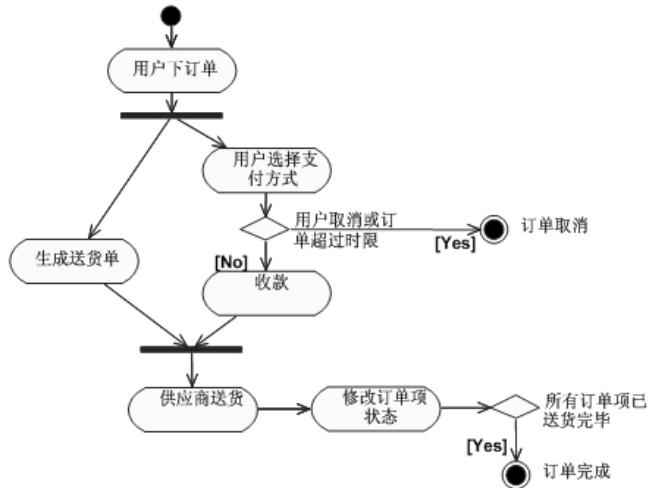


图13-17 用户订单处理简单活动图

初始节点和活动终点

在活动图中有两个特殊的节点，一个用来表示活动的初始节点，它用一个实心圆表示，在一张不包括子图的活动图中有且只有一个初始节点。而另一个则是表示活动处理完成的活动终点，它用一个圆圈内加一个实心圆来表示，在活动图中可能包含多个活动终点。例如，在本例中，用户取消和订单完成就是两个可能的活动终点。

活动节点

活动节点是活动图中最重要的元素之一，它用来表示一个活动，例如图13-18中的“用户下订单”、“用户选择支付方式”、“生成送货单”等都是活动节点。在UML中，活动节点所描述的活动可以是原子的动作，也可以是能进一步分解的一系列操作；它可以是文字描述、表达式、事件等。在图13-19中列出的就是一些可能的活动节点描述。

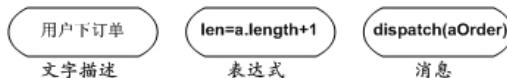


图13-18 活动节点

转换

当一个活动结束时，控制流就会马上传递给下一个活动节点，在活动图中称之为“转换”，用一条带箭头的直线来表示。

定义：从语义上说，这种转换称为无触发转换，也就是一旦前一个动作完成就会马上转到下一个。

如果你需要对这些转换设置一些条件，使其在满足特定的条件时才触发，则可以借助监护条件来完成。

分支与监护条件

对于任何一个控制流而言，都一定会存在分支、循环等形式的控制流。在活动图中，分支用一个菱形表示，它有一个进入转换（箭头从外指向分支符号），一个或多个离开转换（箭头从分支符号指向外）。而每个离开转换上都会有一个监护条件，用来表示满足什么条件的时候执行该转换。但要注意，在多个离开转换上的监护条件不能有矛盾，否则就会使得流程产生混乱。

虽然在活动图中，没有直接提供表示循环的建模元素，但可以利用分支来实现“循环”控制流的表示。例如，在图13-17所示的例子中，我们根据前面章节的描述知道，一个订单可能对应多个供应商，因此如果订单没有完成的话，说明还有供应商没有完成送货任务，因此可以在分支“所有订单项已送货完毕”中，增加一个离开转换，指向“供应商送货”活动节点来表示这种循环，修改后的活动图如图13-19所示。

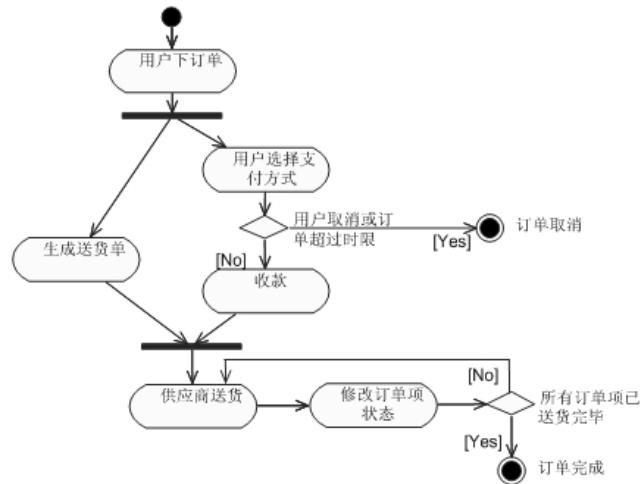


图13-19修改后的简单活动图

分岔与汇合

在实际的控制流中，除了顺序结构、分支结构和循环结构之外，还可能存在并发的事件流。在UML中，可以采用一个同步线来说明这些并行控制流的分岔和汇合。如图13-20所示，同步线是一条水平或垂直的粗线段。

正如图13-20所示，分岔是有一个进入转换，两个或多个离开转换；而汇合则是两个或多个进入转换，一个离开转换。例如，在本例中，当“用户下订单”之后，系统将并行处理两方面事务：一是根据订单所涉及的产品生成送货单；二是处理用户的支付。这两类事件是并发处理。当这两个并发处理都完成时，这时控制流汇合，转到“供应商送货”活动中。

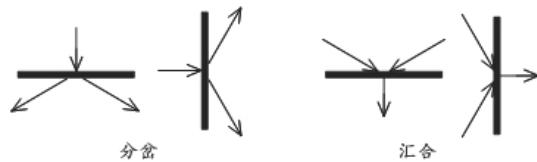


图13-20分岔与汇合图示

2. 带泳道的活动图

简单活动图虽然明确地说明了整个控制流的过程，但是却没有说明每个活动是由谁做的。对应到编程而言，就是没有明确地表示出每个活动是由什么类来负责的；对应到业务建模，就是没有明确地表示出机构中的哪一个部门负责实施什么操作。

为了在简单活动图的基础上，有效地表示各个活动由谁负责的信息，我们可以通过泳道（Swim Lane）来实现。例如针对图13-19所示的活动图，活动的主要负责人包括客户、系统、供应商，因此可以将其分成三个泳道，绘制出如图13-21所示的活动图。

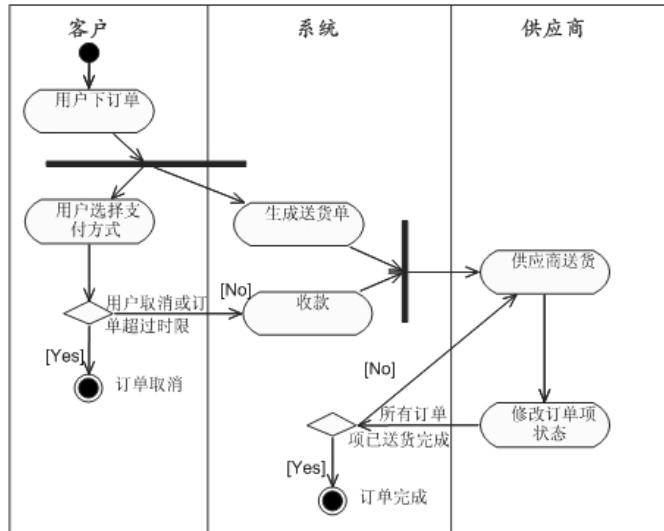


图13-21带泳道的活动图

在图13-21中，泳道将活动图中的活动节点分成了几个小组，每个小组都显示出了负责实施这些操作的角色。在本图中，这些都是一些现实世界中的实体，而同样，你也可以用来表示不同的类。

每个泳道在视觉上是用一条垂直的线将它们分开，并且每个泳道都必须有一个唯一的名称，例如本图中的客户、系统、供应商。从图中也可以看出，每个活动节点、分支是必须只属于一个泳道的，而转换、分岔与汇合是可以跨泳道的。通过泳道，我们不仅体现了整个活动控制流，还体现了每个活动的实施者。

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第 13 章：UML建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月06日

考点精讲—状态图

状态图 (State Diagram) 用来描述一个特定对象的所有可能状态及其引起状态转移的事件。大多数面向对象技术都用状态图表示单个对象在其生命周期中的行为。一个状态图包括一系列的状态及状态之间的转移。

例如我们可以采用如图13-22所示的状态机图，来描述一个烧水器在工作时的详细行为细节。

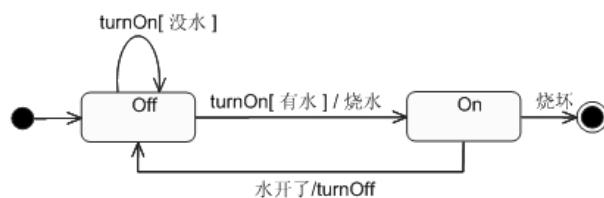


图13-22简单状态机图

从图13-22中不难看出，在一张状态机图中，最为核心的元素无外乎是两个：一个是用圆角矩形表示的状态（初态和终态例外）；另一个则是在状态之间的、包含一些文字描述的有向箭头线，这些箭头线称为转换。在前面我们已经说过了状态的含义和表示法，在此我们重点在于理解“转换”的含义和表示法。

定义：一个转换是两个状态之间的一种关系，表示对象将在第一个状态中执行一定的动作，并

在某个特定事件发生时、并且满足特定条件时进入第二个状态。

如图13-23所示，转换涉及的内容包括源状态、触发事件、监护条件、动作和目标状态五个方面的内容。

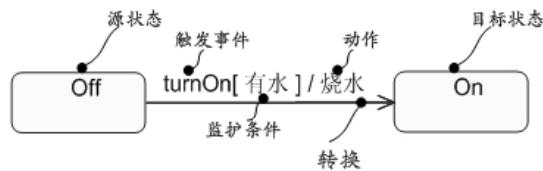


图13-23转换的五要素

源状态和目标状态

对于一个转换发生时，我们变为转换被激活。在转换激活前，处于源状态，在转换激活后就将进入目标状态了。也就是说，

源状态：即受转换影响的状态，如果一个对象处于源状态，那么当该对象收到转换的触发事件并满足监护条件（如果有）时，就会激活转换。

目标状态：即当转换完成后对象的状态。

触发事件

它用来为转换定义一个事件，当源状态中的对象接收到这个事件时，就会使转换合法的激活（如果有定义监护条件，则还需满足监护条件）。在UML中，事件主要可以分为调用事件、改变事件、信号事件和时间事件四种。

调用事件：用某对象的成员方法就称之为调用事件，它是一种同步的机制。例如在图13-23中，turnOn就是一种调用事件，用来将开关置于“on”状态。

改变事件：改变事件是指依赖于指定属性值的布尔表达式得到满足。这是一种一直等待到特定条件被满足的声明方式。它并不常使用。它和监护条件不同，改变事件中的条件是一直计算，直到布尔表达式为真为止。

信号事件：信号是两个对象之间通信的媒介，信号是一种异步机制。在计算机中，鼠标和键盘的操作均是属于此类事件。对于一个信号而言，对象一般都有相应的事件处理器，例如onMouseClick()等。例如，图13-23中“水开了”就是一个信号。

时间事件：时间事件代表时间的流逝。它可以指定为绝对形式（每天的某时，例如after(11:00)），也可以指定为相对形式（从某一指定事件发生开始所经过的时间，例如after(2 seconds)）。不过值得注意的是，对于前一种形式，也可以使用变化事件来描述：when(11:00)。

监护条件

大家对于监护条件应该不会陌生了，在前面的许多地方都提到过。在状态机图中，它的语义仍然没变。它通常是一个布尔表达式，当对象接收到触发事件时，就将对该布尔表达式求值；如果表达式取值为真，则激活转换；如果取值为假，则不激活转换。例如，在图13-23中，当收到turnOn事件时，还将判断壶中“有水”否；如果有水，激活转换；否则不激活转换，因此状态就不会发生改变了。

再次重申监护条件和改变事件的区别，监护条件只在对象收到触发事件时才会计算一次，而改变事件是一直计算的。

动作

当转换被激活后，如果定义了相应的动作，那么就将执行这个动作。它可以是一个赋值语句、

简单的算术运行、发送信号、调用操作、创建和销毁对象、读取和设置属性的值，甚至是一个包含多个动作的活动。例如，在图13-23中，当turnOn后，就将执行“烧水”的动作。

读图小结

当我们认识了转换的五个要素之后，就不难正确地理解整个状态机图中所蕴含的含义了。由于初态到off状态之间是没有任何描述的，因此说明初态就是off。

与状态off相关的转换有两个，其触发事件都是turnOn，只不过其监护条件不同。如果对象收到事件turnOn，那么将判断壶中是否有水；如果[没水]，则仍然处于off状态；如果[有水]，则转为on状态，并执行“烧水”动作。值得说明的是，其实从off状态到on状态的转换是可以不必绘出的，在此绘出只是为了例子的需要。

而与状态on相关的转换也有两个，如果“水开了”就执行turnOff，关掉开关；如果烧坏了，就进入了终态了。

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第13章：UML建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月06日

考点精讲—通信图

通信图和顺序图同属于交互图，但它强调收发消息的对象或角色的结构组织。顺序图和通信图表达了类似的基本概念，但每种图所强调的概念不同，顺序图强调的是时序，通信图则强调消息流经的数据结构。如图13-24所示，这就是一个通信图。

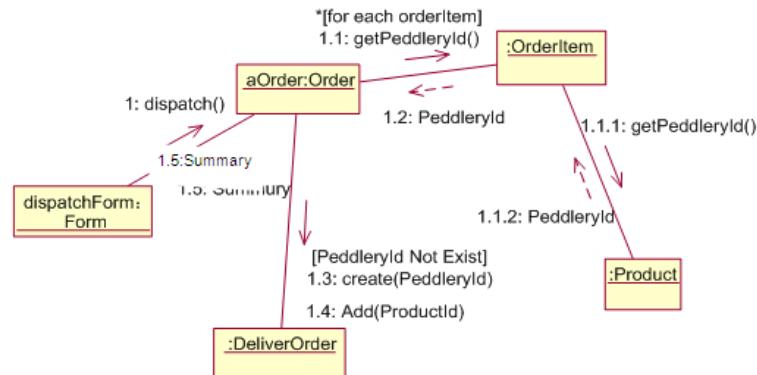


图13-24通信图示例

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第13章：UML建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月06日

考点精讲—构件图

构件图描述一个封装的类和它的接口、端口，以及由内嵌的构件和连接件构成的内部结构。构

件图用于表示系统的静态设计实现视图。对于由小的部件构建大的系统来说，构件图是很重要的。

构件图是类图的变体。如图13-25所示，这就是一个构件图。

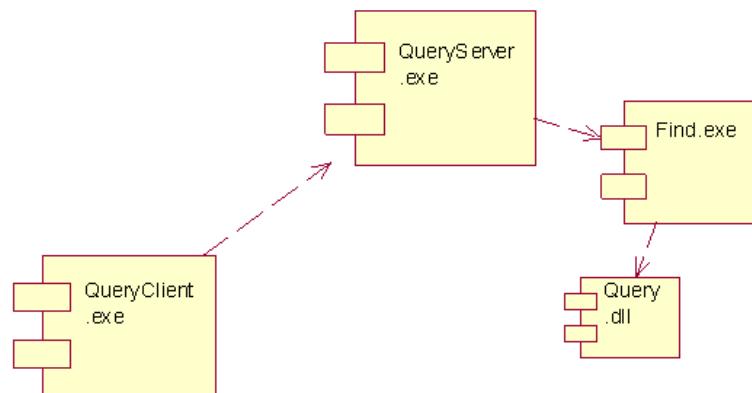


图13-25 构件图示例

通常构件指的是源代码文件、二进制代码文件和可执行文件等。而构件图就是用来显示编译、链接或执行时构件之间的依赖关系。例如，在上图中，就是说明QueryClient.exe将通过调用QueryServer.exe来完成相应的功能，而QueryServer.exe则需要Find.exe的支持，Find.exe在实现时调用了Query.dll。

通常来说，我们可以使用构件图完成以下工作。

对源代码进行建模：这样可以清晰地表示出各个不同源程序文件之间的关系。

对可执行体的发布建模：如图13-25所示，将清晰地表示出各个可执行文件、DLL文件之间的关系。

对物理数据库建模：用来表示各种类型的数据库、表之间的关系。

对可调整的系统建模：例如对于应用了负载均衡、故障恢复等系统的建模。

在绘制构件图时，应该注意侧重于描述系统的静态实现视图的一个方面，图形不要过于简化，应该为构件图取一个直观的名称，在绘制时避免产生线的交叉。

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第 13 章：UML建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月06日

考点精讲—部署图

我们通过构件图将能够理解系统的物理组成结构，但是它并没有办法体现出这些物理组成部分是如何反映在计算机硬件系统之上的。而部署图正是用来弥补这个不足的，它的关注点就在于系统如何部署。部署图，也称为实施图，和构件图一样是面向对象系统的物理方面建模的两种图之一。构件图相对来说，是说明构件之间的逻辑关系，而部署图则是在此基础上更进一步，描述系统硬件的物理拓扑结构以及在此结构上执行的软件。部署图可以显示计算结点的拓扑结构和通信路径、结点上运行的软件构件，常常用于帮助理解分布式系统。

部署图通常可以用于以下情况的建模工作，如表13-4所示。

表13-4 部署图建模工作

建模内容	说明	策略
对处理器和设备建模	通常包括对单机式、嵌入式、客户/服务器式和分布式的拓扑结构的处理器和设备进行建模	识别系统中的计算元素，并为每个元素建模为节点
		如果这些元素代表一般的处理器和设备，则使用相应的构造型
		与类建模一样，可以对节点进行属性和操作的设置
对构件的分布建模	用来可视化地指定某构件的位置与协作关系	将所有有意义的构件，分配到一个特定的节点上 充分考虑在某个节点上可能存在多个构件

开发团队通过构建和维护部署图，将可以为维护人员提供足够的技术信息支持，以保证部署、安装、维护工作的顺利实施。如图13-26所示，这就是一张部署图。

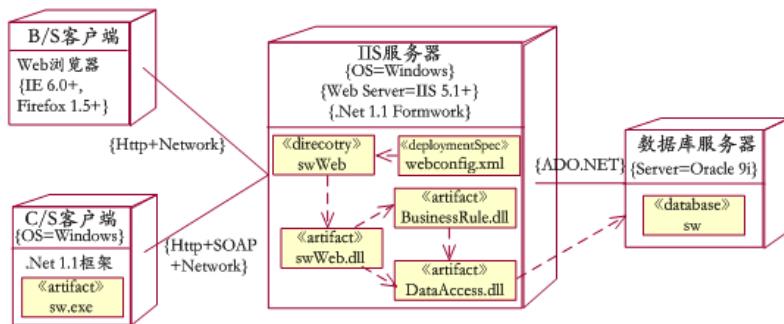


图13-26部署图示例

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第13章：UML建模技术

作者：希赛教育软考学院 来源：希赛网 2014年05月06日

一点一练

试题1

UML序列图是一种交互图，描述了系统中对象之间传递消息的时间次序。其中，异步消息与同步消息不同，（1）。图13-27中（2）表示一条同步消息，（3）表示一条异步消息，（4）表示一条返回消息。

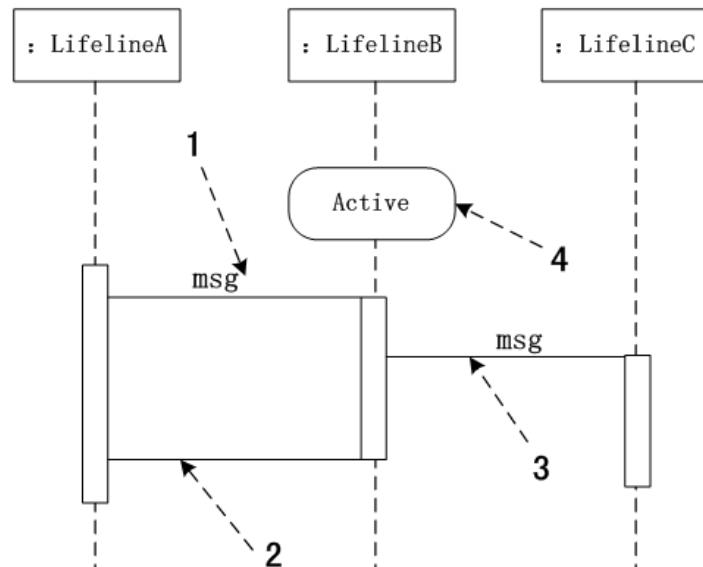


图13-27顺序图

- (1) A. 异步消息并不引起调用者终止执行而等待控制权的返回
 - B. 异步消息和阻塞调用有相同的效果
 - C. 异步消息是同步消息的响应
 - D. 异步消息和同步消息一样等待返回消息
- (2) A. 1 B. 2 C. 3 D. 4
 - (3) A. 1 B. 2 C. 3 D. 4
 - (4) A. 1 B. 2 C. 3 D. 4

试题2

采用UML进行面向对象开发时，部署图通常在__(5)阶段使用。

- (5) A. 需求分析 B. 架构设计 C. 实现 D. 实施

试题3

业务用例和参与者一起描述__(6)，而业务对象模型描述__(7)。

- (6) A. 工作过程中的静态元素 B. 工作过程中的动态元素
 - C. 工作过程中的逻辑视图 D. 组织支持的业务过程
- (7) A. 业务结构
 - B. 结构元素如何完成业务用例
 - C. 业务结构以及结构元素如何完成业务用例
 - D. 组织支持的业务过程

试题4

UML的设计视图包含了类、接口和协作，其中，设计视图的静态方面由__(8)和__(9)表现；动态方面由交互图、__(10)表现。

- (8) A. 类图 B. 状态图 C. 活动图 D. 序列图
 - (9) A. 交互图 B. 对象图 C. 通信图 D. 定时图
- (10) A. 状态图和类图 B. 类图和活动图
 - C. 对象图和状态图 D. 状态图和活动图

试题5

UML中关联的多重度是指__(11)。

- (11) A. 一个类中被另一个类调用的方法个数
- B. 一个类的某个方法被另一个类调用的次数
- C. 一个类的实例能够与另一个类的多少个实例相关联
- D. 两个类所具有的相同的方法和属性

试题6

__(12)是一种很强的“拥有”关系，“部分”和“整体”的生命周期通常一样。整体对象完全支配其组成部分，包括它们的创建和销毁等；__(13)同样表示“拥有”关系，但有时候“部分”对象可以在不同的“整体”对象之间共享，并且“部分”对象的生命周期也可以与“整体”对象不同，甚至“部分”对象可以脱离“整体”对象而单独存在。上述两种关系都是__(14)关系的特殊种类。

- (12) A. 聚合 B. 组合 C. 继承 D. 关联

(13) A . 聚合 B . 组合 C . 继承 D . 关联

(14) A . 聚合 B . 组合 C . 继承 D . 关联

试题7

面向对象技术中，组合关系表示__(15)__。

(15) A . 包与其中模型元素的关系 B . 用例之间的一种关系

C . 类与其对象的关系 D . 整体与其部分之间的一种关系

试题8

在采用标准UML构建的用例模型 (Use Case Model) 中，参与者 (Actor) 与用例 (Use Case) 是模型中的主要元素，其中参与者与用例之间可以具有__(16)__关系。

(16) A . 包含 (Include) B . 递归 (Recursive)

C . 关联 (Association) D . 组合 (Composite)

试题9

阅读下列说明和图，回答问题1至问题3。

【说明】

某网上购物平台的主要功能如下：

(1) 创建订单。顾客 (Customer) 在线创建订单 (Order)，主要操作是向订单中添加项目、从订单中删除项目。订单中应列出所订购的商品 (Product) 及其数量 (quantities)。

(2) 提交订单。订单通过网络来提交。在提交订单时，顾客需要提供其姓名 (name)、收货地址 (address)、以及付款方式 (form of payment) (预付卡、信用卡或者现金)。为了制定送货计划以及安排送货车辆，系统必须确定订单量 (volume)。除此之外，还必须记录每种商品的名称 (Name)、造价 (cost price)、售价 (sale price) 以及单件商品的包装体积 (cubic volume)。

(3) 处理订单。订单处理人员接收来自系统的订单；根据订单内容，安排配货，制定送货计划。在送货计划中不仅要指明发货日期 (delivery date)，还要记录每个订单的限时发送要求 (Delivery Time Window)。

(4) 派单。订单处理人员将已配好货的订单转交给派送人员。

(5) 送货 / 收货。派送人员将货物送到顾客指定的收货地址。当顾客收货时，需要在运货单 (delivery slip) 上签收。签收后的运货单最终需交还给订单处理人员。

(6) 收货确认。当订单处理人员收到签收过的运货单后，会和顾客进行一次再确认。

现采用面向对象方法开发上述系统，得到如图13-28所示的用例图和图13-29所示的类图。

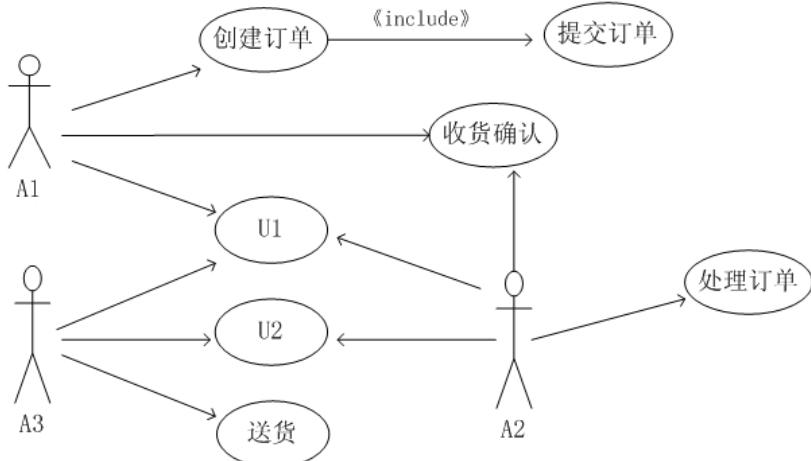


图13-28用例图

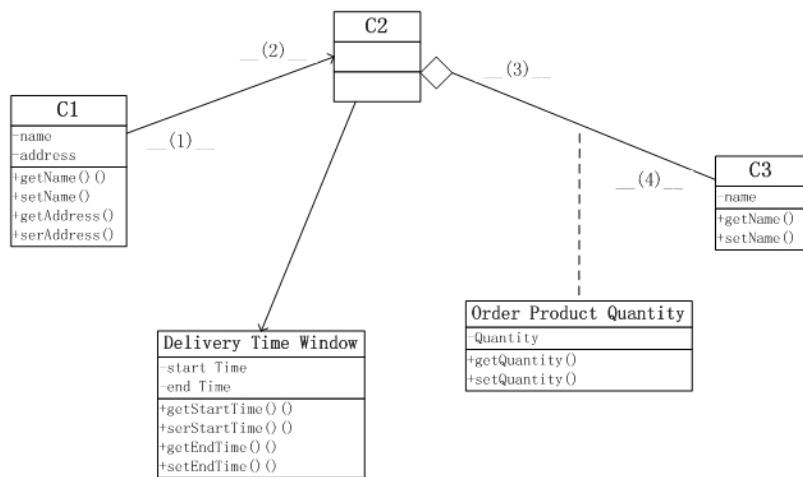


图13-29类图

【问题1】

根据说明中的描述，给出图13-28中A1~A3所对应的参与者名称和U1~U2处所对应的用例名称。

【问题2】

根据说明中的描述，给出图13-29中C1~C3所对应的类名以及(1)~(4)处所对应的多重度(类名使用说明中给出的英文词汇)。

【问题3】

根据说明中的描述，将类C2和C3的属性补充完整(属性名使用说明中给出的英文词汇)。

试题10

阅读下列说明和图，回答问题1至问题3。

【说明】

Pay&Drive系统(开多少付多少)能够根据驾驶里程自动计算应付的费用。

系统中存储了特定区域的道路交通网的信息。道路交通网由若干个路段(Road Segment)构成，每个路段由两个地理坐标点(Node)标定，其里程数(Distance)是已知的。在某些地理坐标点上安装了访问控制(Access Control)设备，可以自动扫描行驶卡(Card)。行程(Trajectory)由一组连续的路段构成。行程的起点(Entry)和终点(Exit)都装有访问控制设备。

系统提供了3种行驶卡。常规卡(Regular Card)有效期(Valid Period)为一年，可以在整个道路交通网内使用。季卡(Season Card)有效期为三个月，可以在整个道路交通网内使用。单次卡(Minitrip Card)在指定的行程内使用，且只能使用一次。其中，季卡和单次卡都是预付卡(Prepaid Card)，需要客户(Customer)预存一定的费用。

系统的主要功能有：客户注册、申请行驶卡、使用行驶卡行驶等。

使用常规卡行驶，在进入行程起点时，系统记录行程起点、进入时间(Date Of Entry)等信息。在到达行程终点时，系统根据行驶的里程数和所持卡的里程单价(Unit Price)计算应付费用，并打印费用单(Invoice)。

季卡的使用流程与常规卡类似，但是不需要打印费用单，系统自动从卡中扣除应付费用。

单次卡的使用流程与季卡类似，但还需要在行程的起点和终点上检查行驶路线是否符合该卡所规定的行驶路线。

现采用面向对象方法开发该系统，使用UML进行建模。构建出的用例图和类图分别如图13-30

和图13-31所示。

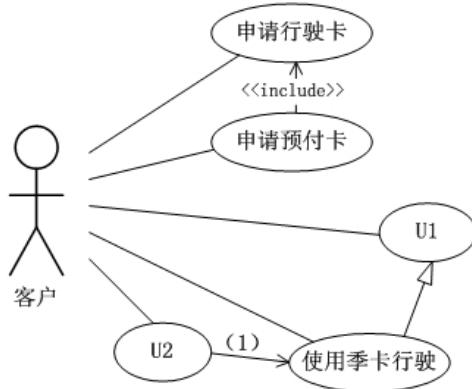


图13-30用例图

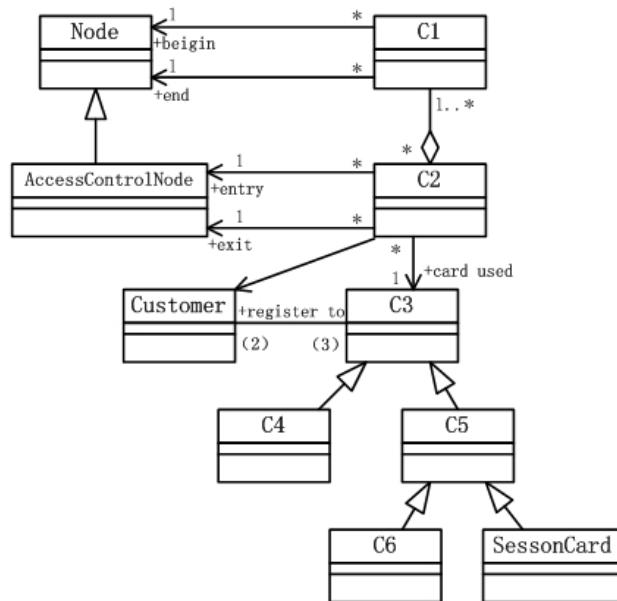


图13-31类图

【问题1】

根据说明中的描述，给出图13-30中U1和U2所对应的用例，以及(1)所对应的关系。

【问题2】

根据说明中的描述，给出图13-31中缺少的C1~C6所对应的类名以及(2)~(3)处所对应的多重度（类名使用说明中给出的英文词汇）。

【问题3】

根据说明中的描述，给出Road Segment、Trajectory和Card所对应的类的关键属性（属性名使用说明中给出的英文词汇）。

版权方授权希赛网发布，侵权必究

上一节 本书简介 下一节

在本题中，首先我们要理解同步消息与异步消息的区别。如果一个对象发送了一个同步消息，那么它要等待对方对消息的应答，收到应答后才能继续自己的操作。而发送异步消息的对象不需要等待对方对消息的应答便可以继续自己的操作。

在本题中，1表示的是同步消息，而2表示的是返回消息，3表示的是异步消息。一般情况下，在顺序图中，同步消息、异步消息、返回消息都是用本题图中的箭头表示，请注意它们的区别。

试题1答案

(1) A (2) A (3) C (4) B

试题2分析

部署图描述了一个运行时的硬件结点，以及在这些结点上运行的软件组件的静态视图。部署图显示了系统的硬件，安装在硬件上的软件，以及用于连接异构的机器之间的中间件。因此它是在实施阶段被使用。

试题2答案

(5) D

试题3分析

这里需要区分业务用例与用例（用例即系统）。业务用例其实是对用例思想的一种延续，只是改变了使用场合。用例是从使用者的角度定义“软件系统”的需求。而业务用例不研究“软件系统”需求，它关心的是一个“业务组织”对外提供哪些服务，支持哪些业务过程。业务用例描述的是业务参与者如何使用业务组织提供的服务的过程。因此业务用例其实是一种业务流程。

业务对象模型是描述业务用例实现的对象模型，即业务结构以及结构元素如何完成业务用例。

试题3答案

(6) D (7) C

试题4分析

本题主要考查UML的静态图与动态图。关于这些图的说明，请参看本章UML基础知识描述部分。

试题4答案

(8) A (9) B (10) D

试题5分析

在UML中，关联的多重度是指一个类的实例能够与另一个类的多少个实例相关联。它又称为重複度。多重度表示为一个整数范围n..m，整数n定义所连接的最少对象的数目，而m则为最多对象数（当不知道确切的最大数时，最大数用*号表示）。最常见的多重性有0..1、0..*、1..1和1..*。

试题5答案

(11) C

试题6分析

本题主要考查我们对类之间一些常用关系的理解。本题我们主要要清楚组合与聚合的联系和区别。组合与聚合都体现着“部分”和“整体”的关系，但组合是一种很强的“拥有”关系，“部分”和“整体”的生命周期通常一样。整体对象完全支配其组成部分，包括它们的创建和销毁等；而聚合有时候“部分”对象可以在不同的“整体”对象之间共享，并且“部分”对象的生命周期也可以与“整体”对象不同，甚至“部分”对象可以脱离“整体”对象而单独存在。

组合与聚合都是关联关系的特殊种类。

试题6答案

(12) B (13) A (14) D

试题7分析

在面向对象技术中，组合描述的是整体与部分的关系，组合关系中，整体与部分的生命周期一致。比如公司与部门就是一种组合关系，公司不存在了，部门自然就不存在了。

试题7答案

(15) D

试题8分析

参与者用于表示使用系统的对象，可以是一个物体或另一个系统。用例是用户期望系统具备的动作。参与者可以与多个用例相关，而用例也可以与多个参与者相关。所以参与者与用例之间可以具有关联关系。

试题8答案

(16) C

试题9分析

本题考查面向对象开发相关知识，涉及UML用例图、类图以及类图设计时的设计模式。UML目前在面向对象软件开发中广泛使用，是面向对象软件开发考查的重要内容。

【问题1】

本题主要考查用例图。

在本题中，从题目的描述中，我们不难知道，本系统的用例主要有：创建订单、提交订单、处理订单、派单、收货、送货及收货确认，本系统的参与者主要有：订单处理人员、顾客和派送人员。

其中在用例图中还没有给出的用例有派单和收货，因此U1和U2应该就是这两个用例，具体他们分别对应哪一个呢？就需要我们先来确认A1~A3所对应的参与者，A1与用例创建订单、U1及收货确认有关系，根据题目描述“顾客在线创建订单”可知A1应该是顾客，

同样的道理，我们不难得出A2是订单处理人员，A3是派送人员。

而用例U1与三个参与者都有关系，那么根据题目描述“派送人员将货物送到顾客指定的收货地址。当顾客收货时，需要在运货单（delivery slip）上签收。签收后的运货单最终需交还给订单处理人员”，不难得出U1应该是收货。而U2是派单。

【问题2】

本问题考查类图。对于这个题目，我们应该结合题目的描述及给出的类图来求解。从题目给出的类图中我们可以看出，C1中包含了属性姓名（name）和收货地址（address），由此不难推断出C1是顾客（Customer）类。

C2与C1和Delivery Time Window类有关联，可以推断出C2应该是订单（Order）类，而C3与C2是一种组合关系，其中C2是整体，而C3是部分，而C2是订单，订单是由商品组成的，由此可以推断出C3是商品（Product）类。

在UML中，多重度又称重复度，多重度表示为一个整数范围n..m，整数n定义所连接的最少对象的数目，而m则为最多对象数（当不知道确切的最大数时，最大数用*号表示）。最常见的多重性有0..1、0..*、1..1和1..*，而*与0..*是等价的。

顾客可以创建多个订单，也可以不创建订单，而一个订单必须属于而且只能属于1个顾客，因此

空(1)与空(2)分别为1和1..*。

一个订单中可以至少应该包含一个商品，也可以包含多个商品，而某商品可以不在任何订单中，也可以是多个订单中都有该商品，因此空(3)与空(4)应该分别是0..*和1..*。

【问题3】

根据题目描述，系统必须记录每种商品的名称 (Name)、造价 (cost price)、售价 (sale price) 以及单件商品的包装体积 (cubic volume)，因此C3除了名称 (Name) 外，还应该拥有造价 (cost price)、售价 (sale price) 以及单件商品的包装体积 (cubic volume) 等属性。

根据题目描述，每个订单应该有其付款方式 (form of payment)、订单量 (volume) 和发货日期 (delivery date)，因此订单的属性至少有volume、delivery date、form of payment。

试题9答案

【问题1】

A1：顾客 A2：订单处理人员 A3：派送人员

U1：收货 U2：派单

【问题2】

C1 : Customer C2 : Order C3 : Product

(1) 1 (2) 0 .. n或0 .. * (3) 0 .. n或0 .. * (4) 1 .. n或1 .. *

【问题3】

C2 : volume、delivery date、form of payment

C3 : cubic volume、cost price、sale price

试题10分析

本题考查面向对象开发相关知识，涉及UML用例图、类图以及类图设计时的设计模式。UML目前在面向对象软件开发中广泛使用，是面向对象软件开发考查的重要内容。

【问题1】

本题主要考查用例图。

用例之间的关系主要有以下三种：

(1) 包含关系。当可以从两个或两个以上的用例中提取公共行为时，应该使用包含关系来表示它们。用<<include>>表示。

(2) 扩展关系。如果一个用例明显地混合了两种或两种以上不同的场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样使描述可能更加清晰。用<<extend>>表示。

(3) 泛化关系。当多个用例共同拥有一种类似的结构和行为的时候，可以将它们的共性抽象成父用例，其他的用例作为泛化关系中的子用例。

在本题中，从题目的描述中，我们不难看出，用例图中缺失的用例有“使用常规卡行驶”和“使用单次卡行驶”，那么U1和U2具体对应哪个用例，我们根据题目说明，并结合用例图来看，“使用季卡行驶”与U1是泛化关系，由此可知U1应该是“使用常规卡行驶”，而U2是“使用单次卡行驶”，根据题目描述“单次卡的使用流程与季卡类似，但还需要在行程的起点和终点上检查行驶路线是否符合该卡所规定的行驶路线”，由此可知，U1是对“使用季卡行驶”的扩展，由此第1空应填<<extend>>。

【问题2】

本问题考查类图。对于这个题目，我们应该结合题目的描述及给出的类图来求解。根据题目的描述，本系统包含的类主要有路段 (Road Segment)、地理坐标点 (Node)、访问控制 (Access Control) 设备、自动扫描行驶卡 (Card)、行程 (Trajectory)、常规卡 (Regular Card)、季卡 (Season Card)、单次卡 (Minitrip Card)、预付卡 (Prepaid Card) 和客户 (Customer) 等。

从类图中C1与类Node的关系和C2与AccessControlNode的关系，再结合题目描述“路段由两个地理坐标点(Node)标定”可以知道C1应该是路段类，而由题目描述“行程 (Trajectory) 由一组连续的路段构成。行程的起点 (Entry) 和终点 (Exit) 都装有访问控制设备”可以知道C2应该是行程 (Trajectory) 类。

而从类图看，C4和C5是继承于C3的，再结合类图中C3与客户类和行程类的关系，可知C3应该是一切卡的抽象类，因此是自动扫描行驶卡 (Card)，而C5是C6和季卡的父类，再根据题目描述“季卡和单次卡都是预付卡 (PrepaidCard)”可知C5是预付卡 (PrepaidCard)，而C6是单次卡 (MinitripCard)，而C4是常规卡 (RegularCard)。

在UML中，多重度又称重复度，多重度表示为一个整数范围n..m，整数n定义所连接的最少对象的数目，而m则为最多对象数（当不知道确切的最大数时，最大数用*号表示）。最常见的多重性有0..1、0..*、1..1和1..*，而*与0..*是等价的。

空 (2) 和 (3) 描述的是客户与卡之间的多重度，题目告诉我们系统有三种卡，因此一个客户最多可以持有这三种卡，因此空 (3) 应填1..3。而一个卡只能被一个客户持有，因此客户这端应该填1。

【问题3】

根据题目描述，RoadSegment类应该包含的关键属性是里程数 (Distance)，因为这能标识一个路段的长度；而Trajectory类应该包含的关键属性是起点 (Entry)、终点 (Exit)、进入时间 (Date Of Entry)，这样才能说明某一次行程是何时从哪里开始到哪里结束的；而Card类的关键属性应包含有效期 (Valid Period) 和里程单价 (Unit Price)。

试题10答案

【问题1】

U1：使用常规卡行驶U2：使用单次卡行驶 (1) : extend

【问题2】

C1 : RoadSegment C2 : Trajectory C3 : Card

C4 : RegularCard C5 : PrepaidCard C6 : MinitripCard

(2) 1 (3) 1..3

【问题3】

RoadSegment的属性 : Distance

Trajectory的属性 : Entry、Exit、DateOfEntry

Card的属性 : UnitPrice、ValidPeriod

版权方授权希赛网发布，侵权必究

试题1

阅读下列说明和图，回答问题1至问题3。

【说明】

一个简单的图形编辑器提供给用户的基本操作包括：创建图形、创建元素、选择元素以及删除图形。图形编辑器的组成及其基本功能描述如下：

- (1) 图形由文本元素和图元元素构成,图元元素包括线条、矩形和椭圆。
 - (2) 显示在工作空间中,一次只能显示一张图形(即当前图形, current)。
 - (3) 提供了两种操作图形的工具:选择工具和创建工具。对图形进行操作时,一次只能使用一种工具(即当前活动工具, active)

- ① 创建工具用于创建文本元素和图元元素。
 - ② 于显示在工作空间中的图形，使用选择工具能够选定其中所包含的元素，可以选择一个元素，也可以同时选择多个元素。被选择的元素称为当前选中元素(selected)。

③种元素都具有对应的控制点。拖拽选定元素的控制点，可以移动元素或者调整元素的大小。现采用面向对象方法开发该图形编辑器，使用UML进行建模。构建出的用例图和类图分别如图13-32和13-33所示。

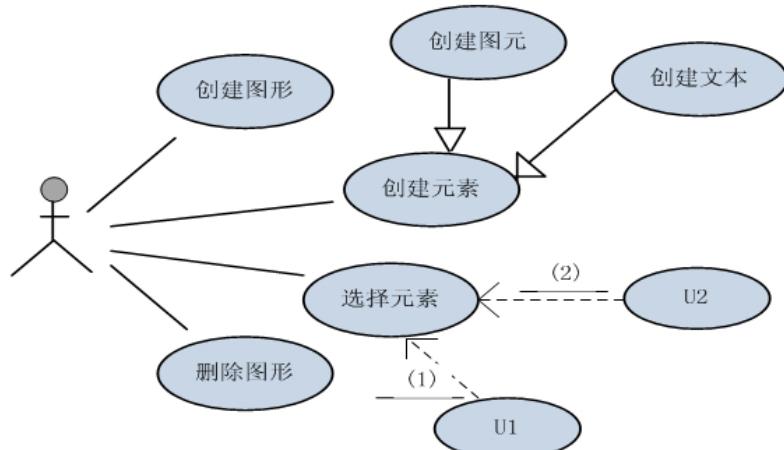


图13-32用例图

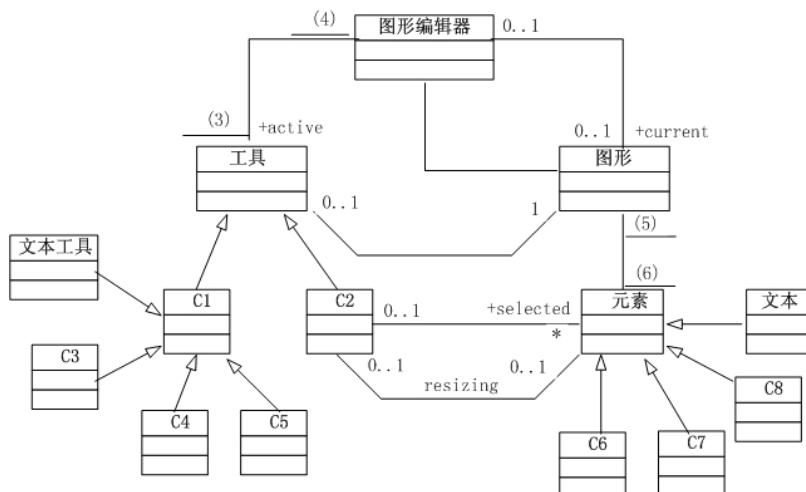


图 13-33 类图

【问题1】

根据说明中的描述，给出图13-32中U1和U2所对应的用例，以及(1)和(2)处所对应的关

系。

【问题2】

根据说明中的描述，给出图13-33中缺少的C1~C8所对应的类名以及(3)~(6)处所对应的多重度。

【问题3】

图13-33中的类图设计采用了桥接(Bridge)设计模式，请说明该模式的内涵。

试题2

阅读以下说明和图，回答问题1至问题3。

【说明】

S公司开办了在线电子商务网站，主要为各注册的商家提供在线商品销售功能。为更好地吸引用户，S公司计划为注册的商家提供商品（Commodity）促销（Promotion）功能。商品的分类（Category）不同，促销的方式和内容会有所不同。

注册商家可发布促销信息。商家首先要在自己所销售的商品的分类中，选择促销涉及的某一具体分类，然后选出该分类的一个或多个商品（一种商品仅仅属于一种分类）；接着制定出一个比较优惠的折扣政策和促销活动的优惠时间；最后由系统生成促销信息并将该促销信息公布在网站上。

商家发布促销信息后，网站的注册用户便可通过网站购买促销商品。用户可选择参与某一个促销（Promotion）活动，并选择具体的促销商品（Commodity），输入购买数量等购买信息。系统生成相应的一份促销订单（POrder）。只要用户在优惠活动的时间范围内，通过网站提供的在线支付系统，确认在线支付该促销订单（即完成支付），就可以优惠的价格完成商品的购买活动，否则该促销订单失效。

系统采用面向对象方法开发，系统中的类及类之间的关系用UML类图表示，图13-34是该系统类图中的一部分；系统的动态行为采用UML序列图表示，图13-35是发布促销的序列图。

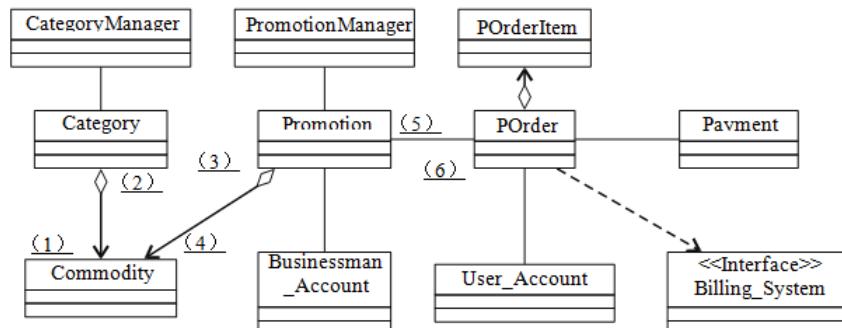


图13-34 在线促销系统部分类图

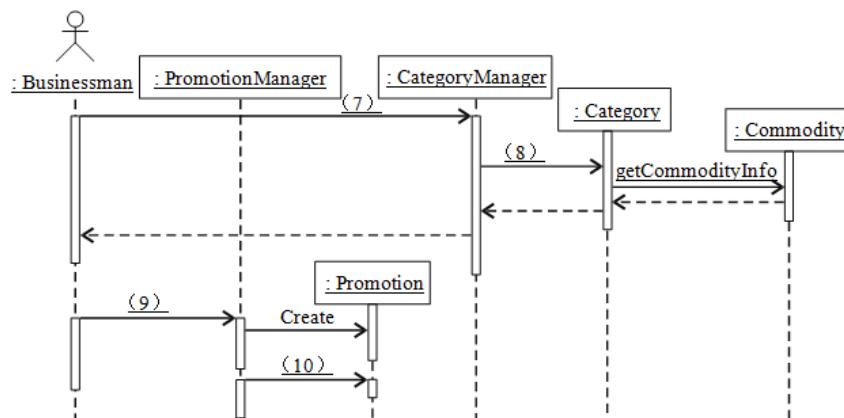


图13-35 发布促销序列图

【问题1】

识别关联的多重度是面向对象建模过程中的一个重要步骤。根据说明中给出的描述，完成图13-34中的(1)~(6)。

【问题2】

请从表13-5中选择方法，完成图13-35中的(7)~(10)。

表13-5 可选消息列表

功能描述	方法名
向促销订单中添加所选的商品	buyCommodities
向促销中添加要促销的商品	addCommodities
查找某个促销的所有促销订单信息列表	getPromotionOrders
生成商品信息	createCommodity
查找某个分类中某商家的所有商品信息列表	getCommodities
生成促销信息	createPromotion
生成促销订单信息	createPOrder
查找某个分类的所有促销信息列表	getCategoryPromotion
查找某商家所销售的所有分类列表	getCategories
查找某个促销所涉及的所有商品信息列表	getPromotionCommodities

【问题3】

关联 (Association) 和聚集 (Aggregation) 是UML中两种非常重要的关系。请说明关联和聚集的关系，并说明其不同点。

试题3

阅读下列说明和图，回答问题1至问题3。

【说明】

某图书管理系统的主要功能如下：

(1) 图书管理系统的资源目录中记录着所有可供读者借阅的资源，每项资源都有一个唯一的索引号。系统需登记每项资源的名称、出版时间和资源状态(可借阅或已借出)。

(2) 资源可以分为两类：图书和唱片。对于图书，系统还需登记作者和页数；对于唱片，还需登记演唱者和介质类型(CD或者磁带)。

(3) 读者信息保存在图书管理系统的读者信息数据库中，记录的信息包括：读者的识别码和读者姓名。系统为每个读者创建了一个借书记录文件，用来保存读者所借资源的相关信息。

现采用面向对象方法开发该图书管理系统。识别类是面向对象分析的第一步。比较常用识别类的方法是寻找问题描述中的名词，再根据相关规则从这些名词中删除不可能成为类的名词，最终得到构成该系统的类。表13-6给出了【说明】中出现的所有名词。

表13-6 题干名词列举

图书管理系统	资源目录	读者	资源
索引号	系统	名称	出版时间
资源状态	图书	唱片	作者
页数	演唱者	介质类型	CD
磁带	读者信息	读者信息数据库	识别码
姓名	借书记录文件	信息	

通过对表13-6中的名词进行分析，最终得到了如图13-36所示的UML类图(类的说明如表13-7所示)。

表13-7类的说明

类名	说明
LibrarySystem	图书管理系统
BorrowerDB	保存读者信息的数据库
CatalogItem	资源目录中保存的每项资源
Borrower	读者
BorrowerItems	为每个读者创建的借书记录文件

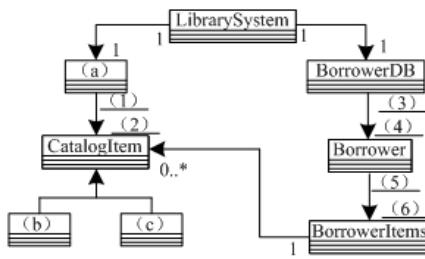


图13-36 类图

【问题1】

表13-7所给出的类并不完整，根据【说明】和表13-6，将图13-36中的(a)~(c)处补充完整。

【问题2】

根据说明中的描述，给出图13-36中的类 CatalogItem，以及(b)、(c)处所对应的类的关键属性（使用表13-6中给出的词汇），其中，CatalogItem 有 4 个关键属性；(b)、(c)处对应的类各有 2 个关键属性。

【问题3】

识别关联的多重度是面向对象建模过程中一个重要步骤。根据【说明】中给出的描述，完成图13-36中的(1)和(6)。

试题4

阅读下列说明和图，回答问题1至问题4。

【说明】

已知某唱片播放器不仅可以播放唱片，而且可以连接电脑并把电脑中的歌曲刻录到唱片上（同步歌曲）。连接电脑的过程中还可自动完成充电。

关于唱片，还有以下描述信息。

(1) 每首歌曲的描述信息包括：歌曲的名字、谱写这首歌曲的艺术家及演奏这首歌曲的艺术家。只有两首歌曲的这3部分信息完全相同时，才认为它们是同一首歌曲。艺术家可能是一名歌手或一支由2名或2名以上的歌手所组成的乐队。一名歌手可以不属于任何乐队，也可以属于一个或多个乐队。

(2) 每张唱片由多条音轨构成；一条音轨中只包含一首歌曲或为空，一首歌曲可分布在多条音轨上；同一首歌曲在一张唱片中最多只能出现一次。

(3) 每条音轨都有一个开始位置和持续时间。一张唱片上音轨的次序是非常重要的，因此对于任意一条音轨，播放器需要准确地知道，它的下一条音轨和上一条音轨是什么（如果存在的话）。

根据上述描述，采用面向对象方法对其进行分析与设计，得到了如表13-8所示的类列表、如图13-37所示的初始类图，以及如图13-38所示的描述播放器行为的UML状态图。

表13-8类列表

类名	说明
Artist	艺术家
Song	歌曲
Band	乐队
Musician	歌手
Track	音轨
Album	唱片

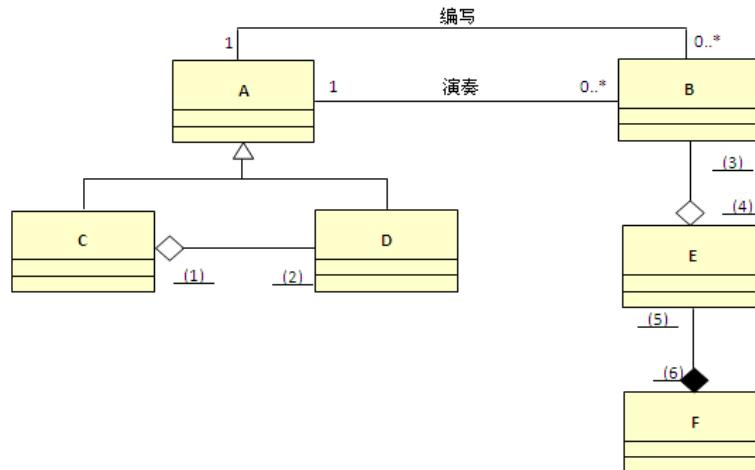


图13-37 初始类图

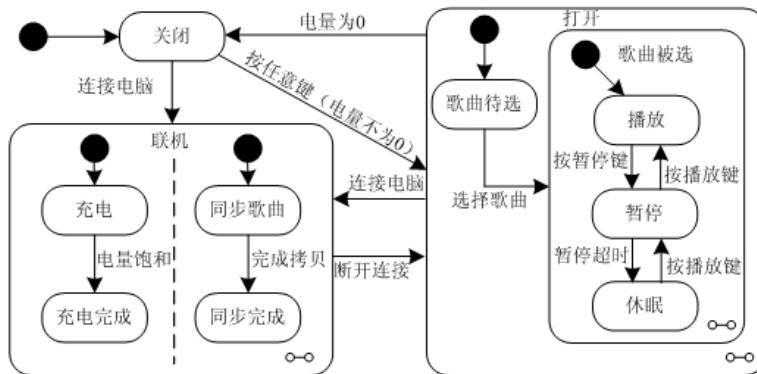


图13-38 播放器行为UML状态图

【问题1】

根据【说明】中的描述，使用表13-8给出的类的名称，给出图13-37中的A~F所对应的类。

【问题2】

根据【说明】中的描述，给出图13-37中(1)~(6)处的多重度。

【问题3】

图13-37中缺少了一条关联，请指出这条关联两端所对应的类，以及每一端的多重度。

类	多重度

【问题4】

根据如图13-38所示的播放器行为UML状态图，给出从“关闭”状态到“播放”状态所经过的最短事件序列（假设电池一开始就是有电的）。

试题5

阅读下列说明和图，回答问题1至问题4。

【说明】

某汽车停车场欲建立一个信息系统，已经调查到的需求如下：

(1) 在停车场的入口和出口分别安装一个自动栏杆、一台停车卡打印机、一台读卡器和一个车辆通过传感器，示意图如图13-39所示。

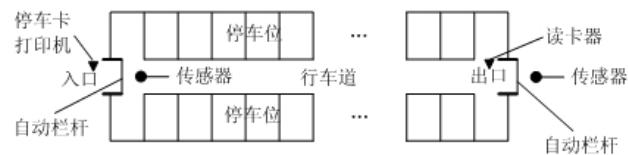


图13-39 停车场示意图

(2) 当汽车到达入口时，驾驶员按下停车卡打印机的按钮获取停车卡。当驾驶员拿走停车卡后，系统命令栏杆自动抬起；汽车通过入口后，入口处的传感器通知系统发出命令，栏杆自动放下。

(3) 在停车场内分布着若干个付款机器。驾驶员将在入口处获取的停车卡插入付款机器，并缴纳停车费。付清停车费之后，将获得一张出场卡，用于离开停车场。

(4) 当汽车到达出口时，驾驶员将出场卡插入出口处的读卡器。如果这张卡是有效的，系统命令栏杆自动抬起；汽车通过出口后，出口传感器通知系统发出命令，栏杆自动放下。若这张卡是无效的，系统不发出栏杆抬起命令而发出告警信号。

(5) 系统自动记录停车场内空闲的停车位的数量。若停车场当前没有车位，系统将在入口处显示“车位已满”信息。这时，停车卡打印机将不再出卡，只允许场内汽车出场。

根据上述描述，采用面向对象方法对其进行分析与设计，得到了如表13-9所示的类/用例/状态列表、如图13-40所示的用例图、如图13-41所示的初始类图，以及如图13-42所示的描述入口自动栏杆行为的UML状态图。

表 13-9类/用例/状态列表

用例名	说明	类名	说明	状态名	说明
Car entry	汽车进入停车场	CentralComputer	停车场信息系统	Idle	空闲状态，汽车可以进入车场
Car exit	汽车离开停车场	PaymentMachine	付款机器	Disable	没有车位
Report Statistics	记录停车场的相关信息	CarPark	停车场，保存车位信息	Await Entry	等待汽车进入
		Barrier	自动护栏	Await Ticket Take	等待打印停车卡
Car entry when full	没有车位时，汽车请求进入停车场	EntryBarrier	入口的护栏	Await Enable	等待停车场内有空闲车位
		ExitBarrier	出口的护栏		

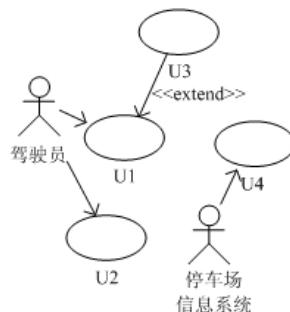


图13-40 用例图

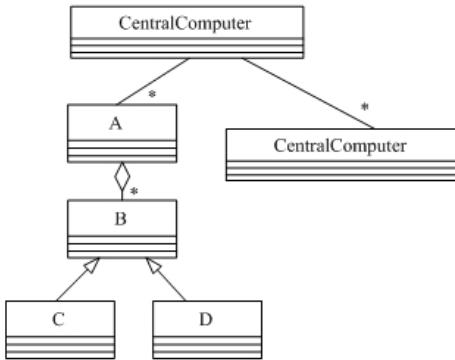
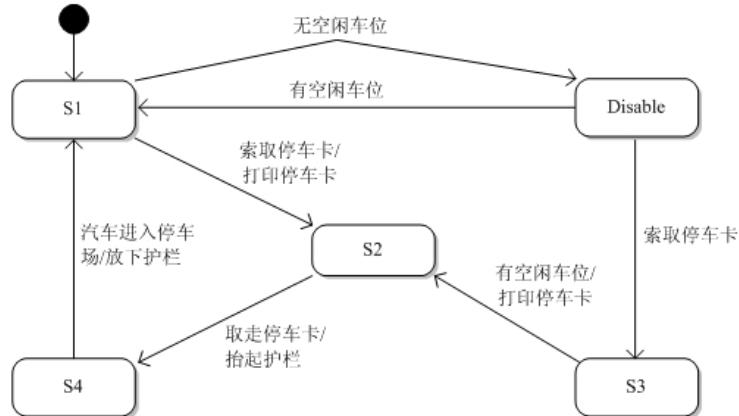


图13-41 初始类图



【问题1】

根据【说明】中的描述，使用表13-9给出的用例名称，给出图13-40中U1、U2和U3所对应的用例。

【问题2】

根据【说明】中的描述，使用表13-9给出的类的名称，给出图13-41中的A~D所对应的类。

【问题3】

根据【说明】中的描述，使用表13-9给出的状态名称，给出图13-42中S1~S4所对应的状态。

【问题4】

简要解释图13-40中用例U1和U3之间的extend关系的内涵。

试题6

阅读下列说明和图，回答问题1至问题4。

【说明】

在线会议审稿系统 (Online Reviewing System, ORS) 主要处理会议前期的投稿和审稿事务，其功能描述如下：

- (1) 用户在初始使用系统时，必须在系统中注册 (register) 成为作者或审稿人。
- (2) 作者登录 (login) 后提交稿件和浏览稿件审阅结果。提交稿件必须在规定提交时间范围内，其过程为先输入标题和摘要，选择稿件所属主题类型，选择稿件所在位置 (存储位置)。上述几步若未完成，则重复；若完成，则上传稿件至数据库中，系统发送通知。
- (3) 审稿人登录后可设置兴趣领域，审阅稿件给出意见，以及罗列录用和 (或) 拒绝的稿件。
- (4) 会议委员会主席是一个特殊审稿人，可以浏览提交的稿件、给审稿人分配稿件、罗列录用和 (或) 拒绝的稿件，以及关闭审稿过程。其中关闭审稿过程须包括罗列录用和 (或) 拒绝的稿

件。

系统采用面向对象的方法开发，使用 UML 进行建模。在建模用例图时，常用的方式是先识别参与者，然后确定参与者如何使用系统来确定用例，每个用例可以构造一个活动图。参与者名称、用例和活动名称分别参见表13-10、表13-11和表13-12。系统的部分用例图和提交稿件的活动图分别如图13-43和图13-44所示。

表13-10 参与者列表

名称	说明	名称	说明
User	用户	Author	作者
Reviewer	审稿人	PCChair	委员会主席

表13-11 用例名称列表

名称	说明	名称	说明
login	登录系统	register	注册
submit paper	提交稿件	browse review results	浏览稿件审阅结果
close reviewing process	关闭审稿过程	assign paper to reviewer	分配稿件给审稿人
set preferences	设定兴趣领域	enter review	审阅稿件给出意见
list accepted/rejected papers	罗列录用或/和拒绝的稿件	browse submitted papers	浏览提交的稿件

表13-12 活动名称列表

名称	说明	名称	说明
select paper location	选择稿件位置	upload paper	上传稿件
select subject group	选择主题类型	send notification	发送通知
enter title and abstract	输入标题和摘要		

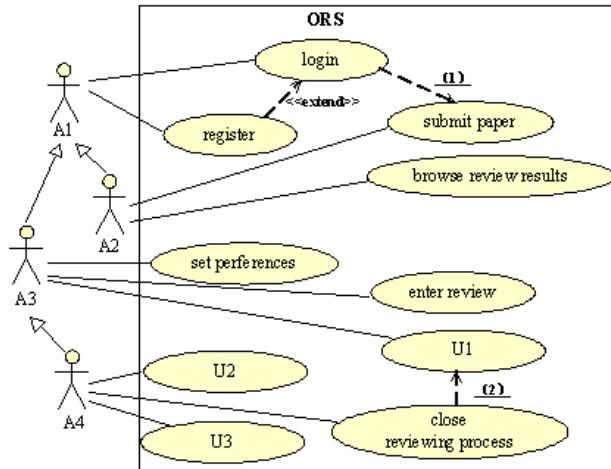


图13-43 ORS 用例图

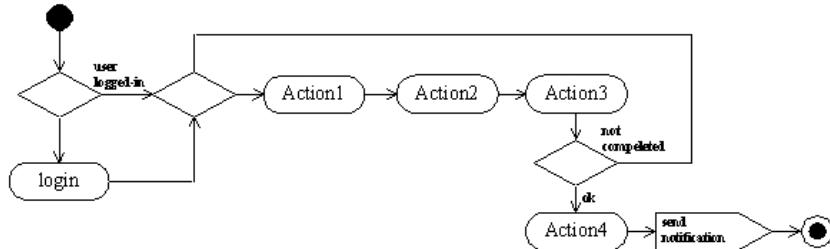


图13-44 提交稿件过程的活动图

【问题1】

根据【说明】中的描述，使用表13-10中的英文名称，给出图13-43中A1~A4所对应的参与者。

【问题2】

根据【说明】中的描述，使用表13-11中的英文名称，给出图13-43中U1~U3所对应的用例。

【问题3】

根据【说明】中的描述，给出图13-43中(1)和(2)所对应的关系。

【问题4】

根据【说明】中的描述，使用表13-11和表13-12中的英文名称，给出图13-44中Action1~Action4对应的活动。

试题7

阅读下列说明和UML图，回答问题1至问题4。

【说明】

某企业为了方便员工用餐，餐厅开发了一个订餐系统(COS: Cafeteria Ordering System)，企业员工可通过企业内联网使用该系统。

企业的任何员工都可以查看菜单和今日特价。

系统的顾客是注册到系统的员工，可以订餐(如果未登录，需先登录)、注册工资支付、预约规律的订餐，在特殊情况下可以覆盖预订。

餐厅员工是特殊顾客，可以进行备餐、生成付费请求和请求送餐，其中对于注册工资支付的顾客生成付费请求并发送给工资系统。

菜单管理员是餐厅特定员工，可以管理菜单。

送餐员可以打印送餐说明，记录送餐信息(如送餐时间)以及记录收费(对于没有注册工资支付的顾客，由送餐员收取现金后记录)。

顾客订餐过程如下：

1. 顾客请求查看菜单；
2. 系统显示菜单和今日特价；
3. 顾客选菜；
4. 系统显示订单和价格；
5. 顾客确认订单；
6. 系统显示可送餐时间；
7. 顾客指定送餐时间、地点和支付方式；
8. 系统确认接受订单，然后发送Email给顾客以确认订餐，同时发送相关订餐信息通知给餐厅员工。

系统采用面向对象方法开发，使用UML进行建模。系统的顶层用例图和一次订餐的活动图初稿分别如图13-45和图13-46所示。

【问题1】

根据【说明】中的描述，给出图13-45中A1和A2所对应的参与者。

【问题2】

根据【说明】中的描述，给出图13-45中缺少的四个用例及其所对应的参与者。

【问题3】

根据【说明】中的描述，给出图13-46中(1)~(4)处对应的活动名称或图形符号。

【问题4】

指出图13-45中员工和顾客之间是什么关系，并解释该关系的内涵。

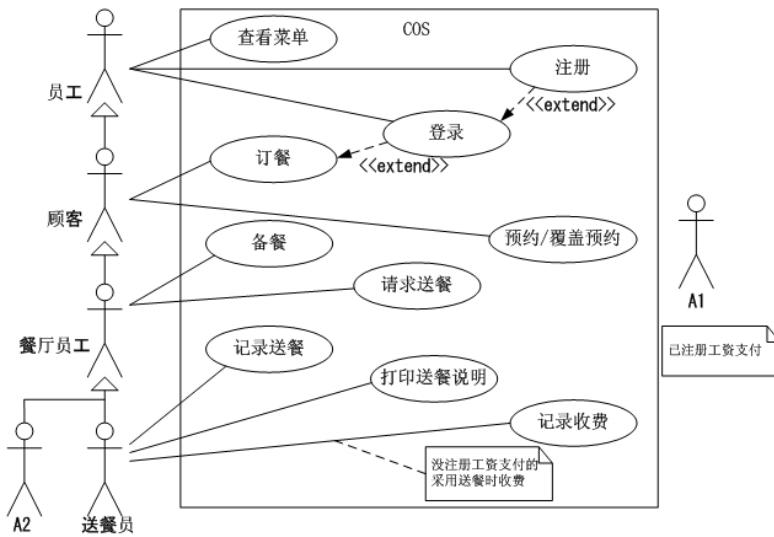


图13-45 COS系统顶层用例图

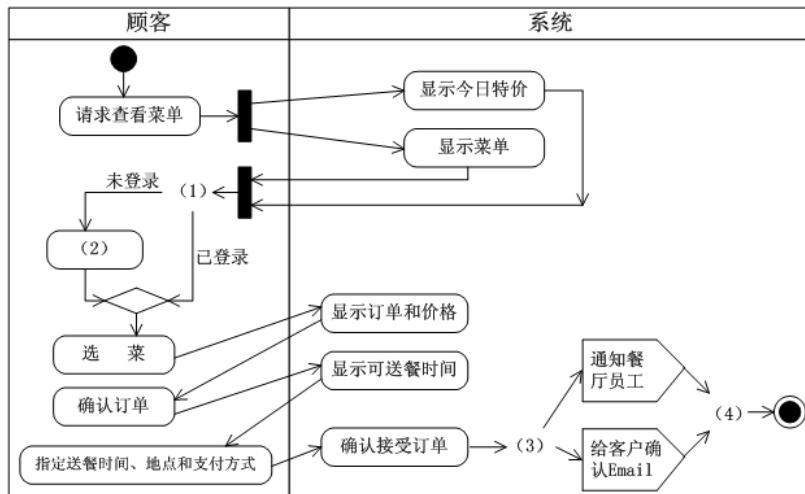


图13-46 一次订餐的活动图

试题8

阅读下列说明和图，回答问题1至问题3。

【说明】

某银行计划开发一个自动存提款机模拟系统 (ATM System)。系统通过读卡器 (CardReader) 读取 ATM 卡；系统与客户 (Customer) 的交互由客户控制台 (CustomerConsole) 实现；银行操作员 (Operator) 可控制系统的启动 (System Startup) 和停止 (System Shutdown)；系统通过网络和银行系统 (Bank) 实现通信。当读卡器判断用户已将 ATM 卡插入后，创建会话 (Session)。会话开始后，读卡器进行读卡，并要求客户输入个人验证码 (PIN)。系统将卡号和个人验证码信息送到银行系统进行验证。验证通过后，客户可从菜单选择如下事务 (Transaction)：

1. 从 ATM 卡账户取款 (Withdraw)；
2. 向 ATM 卡账户存款 (Deposit)；
3. 进行转账 (Transfer)；
4. 查询 (Inquire) ATM 卡账户信息。

一次会话可以包含多个事务，每个事务处理也会将卡号和个人验证码信息送到银行系统进行验证。若个人验证码错误，则转个人验证码错误处理 (Invalid PIN Process)。每个事务完成后，客户可选择继续上述事务或退卡。选择退卡时，系统弹出 ATM 卡，会话结束。系统采用面向对象方法

开发，使用 UML 进行建模。系统的顶层用例图如图 13-47 所示，一次会话的序列图（不考虑验证）如图13-48所示。消息名称参见表13-13。

表13-13可能的消息名称列表

名称	说明	名称	说明
cardInserted()	ATM 卡已插入	performTransaction()	执行事务
performSession()	执行会话	readCard()	读卡
readPIN()	读取个人验证码	PIN	个人验证码信息
creat(atm,this,card,pin)	为当前会话创建事务	create(this)	为当前 ATM 创建会话
Card	ATM 卡信息	doAgain	执行下一个事务
ejectCard()	弹出 ATM 卡		

【问题1】

根据【说明】中的描述，给出图 13-47 中 A1 和 A2 所对应的参与者，U1 至 U3 所对应的用例，以及该图中空 (1) 所对应的关系。（U1至U3的可选用例包括：Session、Transaction、Insert Card、Invalid PIN Process和Transfer）

【问题2】

根据【说明】中的描述，使用表13-13中的英文名称，给出图13-48中6~9对应的消息。

【问题3】

解释图13-47中用例U3和用例Withdraw、Deposit等四个用例之间的关系及其内涵。

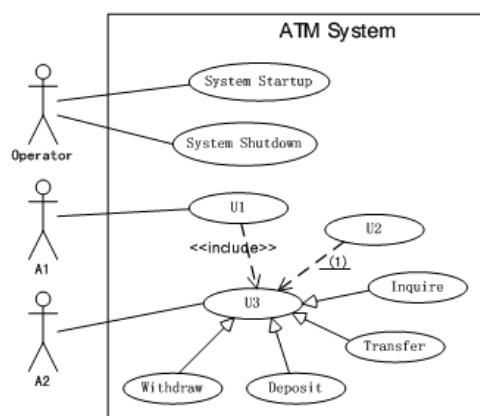


图13-47 ATM系统顶层用例图

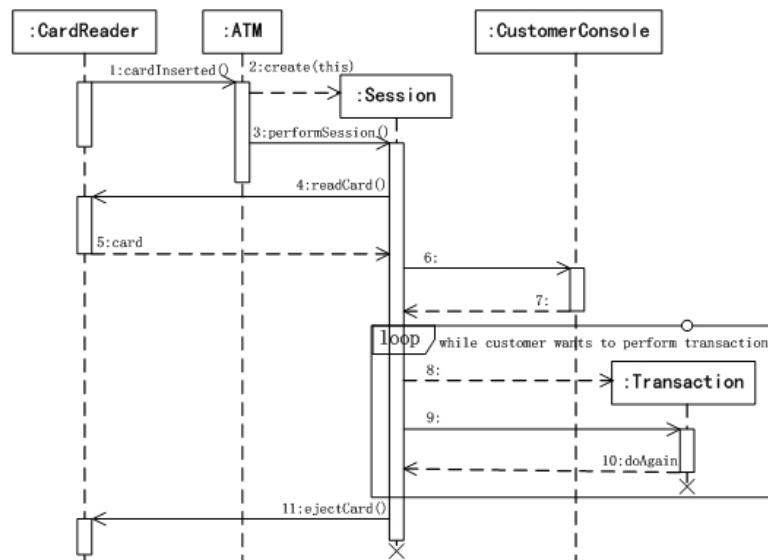


图13-48 一次会话的序列图（无验证消息）

习题解析

试题1分析

本题考查面向对象开发相关知识，涉及UML用例图、类图以及类图设计时的设计模式。UML目前在面向对象软件开发中广泛使用，是面向对象软件开发考查的重要内容。

【问题1】

本题主要考查用例图。

用例之间的关系请参看试题10的分析。

在本题中，从用例图中，我们不难看出U1和U2都与选择元素用例有关系。然后根据题目的描述，可知U1和U2应该分别是移动元素和调整元素的大小，这里我们假定U1是移动元素用例，而U2是调整元素的大小用例。那么接着我们再来确定空(1)与空(2)的内容。这里很显然U1和U2与选择元素用例的关系是扩展关系，因此空(1)与空(2)都应该填<<extend>>。

【问题2】

本问题考查类图。对于这个题目，我们应该结合题目的描述及给出的类图来求解。从题目给出的类图中我们可以看出，C1和C2是继承(泛化)于工具类的，而题目描述告诉我们系统提供了两种操作图形的工具，即选择工具和创建工具，因此C2与C2应该分别是选择工具和创建工具之一，然后我们可以看到文本工具类是继承于C1的，结合题目描述“创建工具用于创建文本元素和图元元素”，我们可以知道C1应该为创建工具类，而C2应该为选择工具类，另外，根据题目描述“图元元素包括线条、矩形和椭圆”，可以知道C6至C8应该分别是线条类、矩形类及椭圆类，当然这三者的答案可以互换。而要能得到这些图形元素，就应该有相应的画图工具，因此C3至C5应该分别是线条工具类、矩形工具类及椭圆工具类，这三者的答案也可以互换。

在UML中，多重度又称重复度，多重度表示为一个整数范围n..m，整数n定义所连接的最少对象的数目，而m则为最多对象数(当不知道确切的最大数时，最大数用*号表示)。最常见的多重性有0..1、0..*、1..1和1..*，而*与0..*是等价的。

由于一个图形编辑器实例可以有一个工具实例，当然也可以没有工具实例，而一个工具实例只能属于一个图形编辑器实例，因此空(3)与空(4)分别为0..1和1。而一个图形至少需要包含一个图形元素，也可以包含多个图形元素，而一个图形元素实例只能属于一个图形实例，所以空(5)与空(6)应该分别是1和1..*。

【问题3】

本问题主要考查桥接模式的基本内容。

桥接模式将抽象部分与它的实现部分分离，使它们都可以独立地变化，对一个抽象的实现部分的修改应该对使用它的程序不产生影响。(3分)

注：仅答出抽象部分与其实现部分分离，给1分；仅答出抽象部分与其实现部分可以独立地变化，对实现部分的修改对使用它的程序不产生影响，给2分；仅答出对实现部分的修改对使用它的程

序不产生影响，给1分。

试题1答案

【问题1】

U1：移动元素 U2：调整元素大小

(1) <<extend>> (2) <<extend>>

【问题2】

C1：创建工具 C2：选择工具 C3：线条工具 C4：矩形工具

C5：椭圆工具 C6：线条 C7：矩形 C8：椭圆

注：C3~C5的答案可以互换；C6~C8

(3) 0..1 (4) 1 (5) 1 (6) 1..*或*

【问题3】

桥接模式将抽象部分与它的实现部分分离，使它们都可以独立地变化，对一个抽象的实现部分的修改应该对使用它的程序不产生影响。

试题2分析

此题考查的是UML类图和顺序图的基本知识，其中第3问涉及的概念已经考过多次。

首先，我们来看问题1，此问是求重复度。由于一个商品分类中可以有多个商品，而一个商品仅对应一个商品分类，所以商品分类与商品之间的关系是：1 : 0..*，所以第(1)空填0..*，第(2)空填1。促销活动与商品之间的关系是这样的：一个促销活动至少得有一种促销商品，否则就无法成为促销活动；而一种商品可以参与多个促销活动，所以促销活动与商品之间的关系有些特别，应是0..* : 1..*，故第(3)空填0..*，第(4)空填1..*。再看订单与促销活动之间的关系：由于题目中说明“用户可选择参与某一个促销(Promotion)活动”，同时对于一个促销活动可以有多个客户下订单，所以它们之间的关系为：1 : 0..*，所以第(5)空填1，第(6)空填0..*。

下面看问题2，此题涉及顺序图基本知识。在顺序图中，消息的执行顺序为：在垂直方向自上至下地执行，其中的虚线表示消息结果的返回。在本顺序图中，包含着两个操作，第一个操作是得到某个商品的信息，其流程是：先在商品分类列表中找到相应的分类，再从分类中找到具体的商品，从此商品对应的类中得到相应信息。所以第(7)空应填getCategories，第(8)空应填getCommodities。第二个操作是创建一次促销活动，并为其指定促销品，所以第(9)空的答案为createPromotion，第(10)空的答案为addCommodities。

最后考的是关联与聚集的概念，这两个概念已多次考查过，下面是对关联以及聚集和组合这三个重要概念的描述。

关联(association)表示两个类的实例之间存在的某种语义上的联系。例如，一个老师为某个学校工作，一个学校有多间教室。我们就认为老师和学校、学校和教室之间存在着关联关系。关联关系为类之间的通信提供了一种方式，它是所有关系中最通用、语义最弱的。关联关系通常可以再细分成以下几种。

聚集关系(aggregation)：是关联关系的特例。聚集关系是表示一种整体和部分的关系。如一个电话机包含一个话筒，一个电脑包含显示器、键盘和主机等都是聚合关系的例子。

组合关系：如果聚集关系中表示“部分”的类的存在，与表示“整体”的类有着紧密的关系，例如“公司”与“部门”之间的关系，那么就应该使用“组合”关系来表示。

关联与聚集之间的不同点在于：聚集表示部分与整体关系的关联；若从生命周期的角度考虑，

则关联对象的生命周期一般无必然关系，聚集的整体对象往往对部分对象的生命周期负责。

试题2答案

【问题1】

(1) 0..* 或 1..*

(2) 1

(3) 0..*

(4) 1..*

(5) 1

(6) 0..*

【问题2】

(7) getCategories

(8) getCommodities

(9) createPromotion

(10) addCommodities

【问题3】

关系：聚集（聚合）是关联的特例。

不同点：聚集表示部分与整体关系的关联；若从生命周期的角度考虑，则关联对象的生命周期一般无必然关系，聚集的整体对象往往对部分对象的生命周期负责。

试题3分析

本题主要考查UML中的类图设计，3个问题都是对类图的元素进行补充。类图的设计是根据系统功能需求而来的，所以解题的关键在于对“系统功能说明”的理解。下面我们将通过对“系统功能说明”的分析来解答试题。

(1) 从系统功能说明中的“图书管理系统的资源目录中记录着所有可供读者借阅的资源”和“资源可以分为两类：图书和唱片”，可以得知1个资源目录中对应着多个可供读者借阅的资源，同时图书类与唱片类是资源类的子类。所以(a)为资源目录，(b)和(c)分别为图书和唱片，同时(1)处应填：1，(2)处应填：0..*（所有的可供读者借阅资源数有可能为0，即还未录入任何资源的状态）。

(2) 从“每项资源都有一个唯一的索引号。系统需登记每项资源的名称、出版时间和资源状态”，可以得知，资源目录中的每项资源，即类图中的CatalogItem，有索引号、名称、出版时间和资源状态4个关键属性。

(3) 从“对于图书，系统还需登记作者和页数；对于唱片，还需登记演唱者和介质类型（CD或者磁带）”可以得知，图书有作者和页数2个关键属性，唱片有演唱者和介质类型2个关键属性。

(4) BorrowerDB代表保存读者信息的数据库，而Borrower代表读者，所以它们之间的关系毫无疑问是1对多，但具体是0..*还是1..*呢？应是0..*，因为读者信息数据库可以为空，即还没有任何读者。即第(3)空填1，第(4)空填0..*。

(5) 由于Borrower代表读者，而BorrowerItems为借书记录文件，同时系统功能说明中有“系统为每个读者创建了一个借书记录文件，用来保存读者所借资源的相关信息”，所以它们之间的关系应为1对1，即第(5)空和第(6)空均填1。

试题3答案

【问题1】

(a) 资源目录

(b) 图书

(c) 唱片

注：(b) 和 (c) 的答案可以互换。

【问题2】

CatalogItem的属性：索引号、名称、出版时间、资源状态

图书的属性：作者、页数

唱片的属性：演唱者、介质类型

【问题3】

(1) 1 (2) 0..* (3) 1

(4) 0..* (5) 1 (6) 1或者 0..1

试题4分析

本道题主要考查考生对UML（统一建模语言）的类图、状态图的掌握。

【问题1】

根据“每首歌曲的描述信息包括：歌曲的名字、谱写这首歌曲的艺术家及演奏这首歌曲的艺术家”和图13-37中类A与类B之间约束为“编写”、“演奏”，所以类A与类B只能是艺术家和歌曲，又根据图上标示的关联关系(1,0..*)，可以确定类A为艺术家(Artist)；类B为歌曲(Song)。类B与类E之间是聚集关系，根据题中“一条音轨中只包含一首歌曲或为空，一首歌曲可分布在多条音轨上”，可以得到类E为音轨(Track)。接下来看，类E与类F之间存在组成的关系，根据“每张唱片由多条音轨构成”得到，类F为唱片(Album)。再来看类C和类D，它们与类A存在泛化关系，根据“艺术家可能是一名歌手或一支由2名或2名以上的歌手所组成的乐队”可知，类C与类D为歌手和乐队，又因为类C与类D存在聚集关系，根据题中“一名歌手可以不属于任何乐队，也可以属于一个或多个乐队”可知，类C为乐队(Band)，类D为歌手(Musician)。

【问题2】

由第一问可知，类C为乐队，类D为歌手，题中“一支由2名或2名以上的歌手所组成的乐队。一名歌手可以不属于任何乐队，也可以属于一个或多个乐队”，则第(1)空为：0..*；第(2)空为2..*。类B与类E存在聚集关系，题中“一条音轨中只包含一首歌曲或为空，一首歌曲可分布在多条音轨上”，所以第(3)空为0..1，第(4)空为1..*。类E与类F存在泛化关系，题中“每张唱片由多条音轨构成”，所以第(5)空为1..*，第(6)空为1。特别要说明一下，是0..*还是1..*，要看表述和实际，比如第(5)空，一张唱片至少有几条音轨，当然至少有一条，否则就不是唱片了，故是从1开始的。

【问题3】

本问题考查的是类/对象关联中的一种特殊关联：递归关联，它描述的是同一个类的不同实例之间的关系。而类Track的不同实例之间恰好具有这种关系（因此对于任意一条音轨，播放器需要准确地知道，它的下一条音轨和上一条音轨是什么）。所以缺少的那条联系的两端都是类Track，其多重度都为0..1。下限为0，是对应不存在上一条或下一条音轨的情况。

【问题4】

状态图是描述系统动态行为的一种模型。这里状态图的考查仅限于能够理解它所描述的行为。

状态图由状态及状态之间的迁移构成，迁移可以由相关的事件触发。问题4给定了两个状态“关闭”和“播放”，要求找出从“关闭”到“播放”的最短事件序列。这就要求我们能够在状态图上找到连接这两个状态的最短迁移，然后将迁移上的事件记录下来就可以了。

从“关闭”状态到“播放”状态可以选择经过迁移“连接电脑”到达“联机”状态，再经过迁移“断开连接”到达状态“打开”，再从“打开”状态的初始状态“歌曲待选”，经过迁移“选择歌曲”到达“播放状态”。这样经过的事件序列为：连接电脑→电量饱和/完成复制→断开连接→选择歌曲。显然这样的事件序列远比“关闭”经过“按任意键”直接到达“打开”状态要长得多。所以从“关闭”到“播放”的最短事件序列是：按任意键，选择歌曲。

试题4答案

【问题1】

A : Artist B : Song C : Band D : Musician
E : Track F : Album

【问题2】

(1) 0..* (2) 2..* (3) 0..1 (4) 1..* (5) 1..* (6) 1

【问题3】

类多重度

Track0..1

Track0..1

【问题4】

按任意键，选择歌曲。

试题5分析

本题主要考查面向对象分析中类图、用例图、状态图。

问题一是根据题目的描述及根据表13-9中的内容，给出图13-40中U1、U2和U3所对应的用例。

用例图描述了一组用例、参与者及它们之间的关系。用例图包括以下几个部分：用例

(Case)、参与者 (Actor)。用例视图中的参与者与系统外部的一个实体 (可以是任何人或事物)，以某种方式参与了用例的执行过程；用例是一个叙述型文档，用来描述参与使用系统，完成某个事情时发生的顺序。

泛化、包含和关联关系分述如下。

(1) 泛化关系 (Generalization)：用例的泛化关系与类的泛化关系相似，即在用例泛化中，子用例表示父用例的特殊形式，子用例从父用例继承了行为和属性，还可以添加行为和属性，改变已继承的行为。

(2) 包含关系 (Include)：包含关系把几个用例的公共步骤分离成一个被包含的用例，用例间的包含关系允许包含提供者用例的行为到客户用例中。包含用例称为客户用例，被包含用例称为提供者用例。包含用例给客户用例提供功能。

(3) 扩展关系 (Extend)：是把新行为插入到已有用例中的方法。基础用例提供了一组扩展点，这些扩展点可以添加新的行为，而扩展用例提供了一组插入片段，这些片段能插入到基础用例的扩展点中。

题目中车辆入场和出场，而入场时分有空位和无空位的情形，当无车位时显示“车位已满”信息。这时，停车卡打印机将不再出卡，只允许场内汽车出场。说明入场时，没有车位入场是一种扩

展关系。根据图13-40和表13-9可以得出U1为Car entry，U2为Car exit，U3为Car entry when full。

问题二是考查对类图的理解。根据题目的描述及表13-9中的内容，可以先来确定类B。汽车出入口，当卡有效时，系统自动抬起栏杆；当卡无效时，则系统不抬栏杆，且发出警告。所以自动护栏类（Barrier）有两种子类：一个是入口的护栏类（EntryBarrier），另一个就是出口的护栏类（ExitBarrier）。构成了这种父子关系的类在图13-41中表示为：类B为护栏类（Barrier），类C为入口护栏类（EntryBarrier），类D为出口护栏类（ExitBarrier）。

再确定类A，由于停车场管理系统管理着多张卡，从图13-40中可以看出类Centralcomputer与类A之间有1..*的关系；而且类A与类B（Barrier）之间存在聚集关系；题目的描述中有：当有车位时允许入场，无车位时停车卡打印机将不再出卡，只允许场内汽车出场。所以一张卡片可以确定多个护栏抬起或不发卡入场，由表13-9可以得出类A为停车场保存卡位信息类（CarPark）。

问题三是考查对状态图的理解。状态图用来建模对象是如何改变其状态的。状态定义为对象行为在某一时刻的快照或者转折点。例如，计算机的状态可以定义为开机、启动、工作中、空闲、关机和离线等。状态图的任务就是用来描述计算机如何从离线状态进入启动状态，以及如何从处理状态进入空闲状态。

根据题目的描述和表13-9，黑点表示开始状态，到达S1，很容易确定S1为状态：Idle（空闲状态，汽车可以进入停车场）。又因为状态Disable（没有车位）到S3有事件“索取停车卡”，而从S3到S2有事件“有空闲车位/打印停车卡”，由题目的第（4）、（5）点可知，车位满了后，若有车辆出去，则释放一个车位；若没有，则等待打印停车卡。所以可以确定S3的状态为Await Ticket Take（等待打印停车卡）。

S1到S2有事件“索取停车卡/打印停车卡”，S2到S4有事件“取走停车卡/抬起护栏”，包括S3到S2有事件“有空闲车位/打印停车卡”，则说明S2这个状态都与“有车位，才发卡”有关，要等待有车位才发卡，或取卡放行后进入等待。所以S2为状态Await Enable（等待停车场内有空位）。

最后来确定S4。由于S2到S4有事件“取走停车卡/抬起护栏”，S4到S1有事件“汽车进入停车场/放下护栏”。很显然，当取走停车卡/抬起护栏将车子放行后，管理系统将停车位的空闲车位数加1；当汽车进入停车场/放下护栏后，管理系统将停车位的空闲车位数减1。因此状态S4为Await Entry（等待汽车进入）。

问题四是考查对扩展关系的理解。题中汽车的入场，通常是指有空位入场；但也有要入场但没有空位的情况，这要等待。而这种关系就是扩展了的入场关系。扩展关系（Extend）是把新行为插入到已有用例中的方法。基础用例提供了一组扩展点，这些扩展点可以添加新的行为，而扩展用例提供了一组插入片段，这些片段能插入到基础用例的扩展点中。

试题5参考答案

【问题1】

U1 : Car entry U2 : Car exit U3 : Car entry when full

【问题2】

A : CarPark B : Barrier C : EntryBarrier

D : ExitBarrier

其中，C、D的答案可以互换。

【问题3】

S1 : Idle S2 : Await Ticket Take S3 : Await Enable S4 : Await Entry

【问题4】

用例之间的扩展关系用于对被用户看作是可选系统行为的用例的一部分建模。通过这种方式，可以把可选行为从必需的行为中分离出来。

试题6分析

该题是一道UML设计题。题目以“希赛公司在线会议审稿系统”为例，考查考生对UML用例图与活动图的掌握情况。

试题共有4个小问题，下面将介绍各个问题的解答要点。

【问题1】

要求考生补充用例图中的参与者，题目已经给出了4类参与者：用户、作者、审稿人、委员会主席。所以关键在于弄清楚各个参与者之间的关系，这些关系是通过题目中的系统功能描述来获得的。

(1) “用户在初始使用系统时，必须在系统中注册 (register) 成为作者或审稿人”，从此处可以得知系统中的用户分成了两类：作者和审稿人。

(2) “会议委员会主席是一个特殊审稿人”。

从上面两个条件得知：A1对应用户，A2对应作者，A3对应审稿人，A4对应会议委员会主席。同时由于UML图中不允许出现中文，且题目明确要求用英文名称给出A1~A4所对应的参与者，所以A1~A4处应填：A1 : User、A2 : Author、A3 : Reviewer、A4 : PCChair。

【问题2】

由“会议委员会主席是一个特殊审稿人，可以浏览提交的稿件，给审稿人分配稿件，罗列录用和(或)拒绝的稿件，以及关闭审稿过程”结合“用例名称列表”可以得知：会议委员会主席能操作的功能有浏览提交的稿件、分配稿件给审稿人、罗列录用或/和拒绝的稿件、关闭审稿过程。而从“其中关闭审稿过程须包括罗列录用和(或)拒绝的稿件”可以看出，用例“关闭审稿过程”与“罗列录用或/和拒绝的稿件”之间有包含关系。从这个关系，我们可以得知U1对应的用例为：罗列录用或/和拒绝的稿件。同时(2)对应的关系为包含关系，即U1填：list accepted/rejected papers，(2)填：<<include>>。这样剩余的两项功能“浏览提交的稿件”和“分配稿件给审稿人”对应的为U2与U3，所以U2和U3分别填browse submitted papers和assign paper to reviewer。

【问题3】

该小题考查考生对包含与扩展关系的理解。包含关系，用构造型<<include>>表示；而扩展关系，用构造型<<extend>>表示。

包含关系：如果可以从两个或两个以上的原始用例中提取公共行为，或者发现能够使用一个构件来实现某一个用例的部分功能是很重要的事时，应该使用包含关系来表示它们，如图13-49所示。

扩展关系：如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能发生多种事情，我们将这个用例分为一个主用例和一个或多个辅用例，描述可能更加清晰，如图13-50所示。

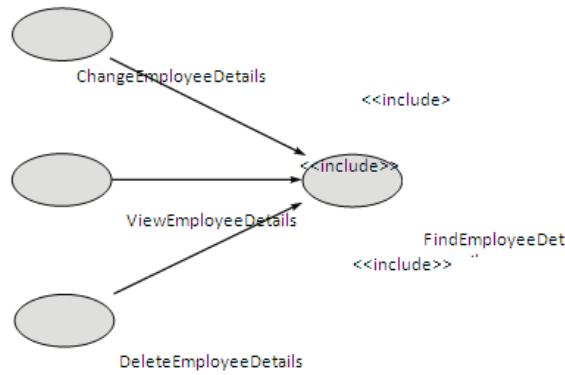


图13-49 包含关系示例图

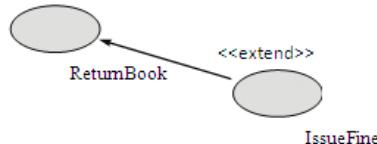


图13-50 扩展关系示例图

在对问题2的分析中，我们已经得出（2）填：`<<include>>`。现在来看（1），该空是填“登录”与“提交稿件”之间的关系，在提交稿件时，若用户已经登录，则可直接提交；但如果用户没有登录，则需要先登录再提交，所以它们之间的关系应是扩展关系。（1）应填：`<<extend>>`。

【问题4】

该题是补充活动图。该活动图所描述的是作者提交稿件的过程，对此过程题目有详细的描述：“作者登录（login）后提交稿件和浏览稿件审阅结果。提交稿件必须在规定提交时间范围内，其过程为先输入标题和摘要、选择稿件所属主题类型、选择稿件所在位置（存储位置）。上述几步若未完成，则重复；若完成，则上传稿件至数据库中，系统发送通知。”，所以Action1~Action4分别对应：输入标题和摘要、选择稿件所属主题类型、选择稿件所在位置、上传稿件。所以Action1~Action4分别填：enter title and abstract、select subject group、select paper location、upload paper。

试题6参考答案

【问题1】

A1 : User A2 : Author A3 : Reviewer A4 : PCChair

【问题2】

U1 : list accepted/rejected papers U2 : browse submitted papers

U3 : assign paper to reviewer

注：U2和U3的答案可互换

【问题3】

（1）：`<<extend>>` （2）：`<<include>>`

【问题4】

Action1 : enter title and abstract Action2 : select subject group

Action3 : select paper location Action4 : upload paper

试题7分析

本题考查面向对象系统开发时，采用UML模型进行建模的方法。

此类题目要求考生认真阅读题目说明中对现实问题的描述，使用UML建模时的原则，从中确定

用例图、活动图以及图中的各种关系。题目给出了未完成的用例图和活动图，需要根据描述给出参与者、用例、活动图中的活动和符号，以及参与者之间的关系内涵。

用例图是用例建模的一个重要产物，它以图形化的方式将系统描述成用例、参与者及其之间的关系。用例图在高层交流了系统必须处理的业务事件的范围，是描述系统与其他外部系统以及用户之间交互的图形。发起或者触发用例的外部用户称为参与者。为了完成某些业务任务，参与者发起系统活动，即用例。在构建用例图时，常用的方式是先识别参与者，然后确定用例以及用例之间的关系。

UML活动图用于建模系统的过程步骤或活动。构造活动图通常先为用例添加开始和结束点，为用例的主要步骤添加一个活动，从每个活动到其他活动、决策点和终点添加转换，并行活动的地方添加同步条。

【问题1】

识别参与者时，考查和系统交互的人员和外部系统。本题中，与系统交互的人员包括员工、注册到系统的员工（顾客）、餐厅员工、菜单管理员、送餐员以及工资系统。

由“菜单管理员是餐厅特定员工”以及图中A2和图中餐厅员工之间的“是一种”关系可知，A2为菜单管理员；图中还缺少描述中与工资系统的交互，由“……并发送给工资系统”可知，A1为工资系统。

【问题2】

考查用例及其和参与者之间的关系时，通过判断哪一个特定参与者发起或者触发了与系统的哪些交互，来识别用例并建立和参与者之间的关联。

本题中，由“任何员工都可以查看菜单和今日特价”可知，图中缺少用例查看今日特价，对应参与者是员工；由“系统的顾客是……，注册工资支付、……”可知，图中缺少用例注册工资支付，对应参与者是顾客和工资系统；由“餐厅员工是……，可以进行备餐、生成付费请求……发送给工资系统”可知，图中缺少用例“生成付费请求”，对应的参与者是餐厅员工和工资系统；由“菜单管理员是餐厅特定员工，可以管理菜单”可知，图中缺少用例管理菜单，对应的参与者是菜单管理员。

需要注意的是，在注册工资支付所对应的参与者中，虽然没有明确说明要和工资系统交互，但是由“对于注册工资支付的顾客生成付费请求并发送给工资系统”可知，工资支付是由工资系统控制，所以注册也需要和工资系统交互。

【问题3】

在顾客订餐过程的描述中，在“顾客选菜”之前，图中缺少符号和活动。由说明中顾客“可以订餐（如果未登录，需先登录）”可以判断，在系统“显示菜单和今日特价”之后“顾客选菜”之前，需要判断（判定符号 ）当前用户身份是否为顾客，如果不是，需先登录；由“……发送E-mail给顾客以确认订餐，同时发送相关订餐信息通知给餐厅员工”可知，发送E-mail和通知餐厅员工为并行活动，需要在前后有同步条 （或纵向）。

【问题4】

参与者之间的关系表示子类型“是一种”父类型，即泛化关系。其中父类型通常是一个抽象泛化的参与者，可以完成子类型可完成的共同行为，每个具体的子类型继承它，可以完成父类型参与者同样的任务，并可以补充额外的角色功能。

试题7答案

【问题1】

A1：工资系统 A2：菜单管理员

【问题2】

第13章：UML建模技术

作者：希赛教育软考

习题解析

参与者	
 	
注册工资支付	顾客和工资系统（或顾客和 A1）
生成付费请求	餐厅员工和工资系统（或餐厅员工和 A1）
管理菜单	菜单管理员（或 A2）

（注：四行顺序可以不同）

【问题3】



【问题4】

泛化关系（一般/特殊关系、继承关系）。泛化关系描述了一个参与者——另一个参与者同样的任务，并可补充额外的角色功能。

试题8分析

本题涉及面向对象系统开发时的UML用例图、序列图以及用例之间的关系。

【问题1】

构建用例图时，常用的方式是先识别参与者，然后确定用例以及用例之间的关系。

识别参与者时，考查和系统交互的人员和外部系统。本题中，与系统交互的人员包括客户（Customer）和银行操作员（Operator），与本模拟系统交互的外部系统包括银行。系统（Bank）。

考查用例时，通过判断哪一个特定参与者发起或者触发了与系统的哪些交互，来识别用例并建立和参与者之间的关联。考查用例之间的关系时，<<include>>（包含）定义了用例之间的包含关系，用于一个用例包含另一个用例的行为的建模；如果可以从一个用例的执行中，在需要时转向执行另一个用例，执行完返回之前的用例继续执行，用例即存在<<extend>>关系。

本题中，客户一旦插卡成功，系统就创建会话（Session），会话中可以执行用户从菜单选择的Withdraw、Deposit、Transfer和Inquire等事务（Transaction）。由图中U3和Withdraw之间的扩展关系，可知U3为Transaction；又由U1和U3之间的<<include>>关系，得知U1为Session，进而判定图中A1为Customer，A2为Bank。每个事务处理也会将卡号和个人验证码信息送到银行系统进行验证，若个人验证码错误，则转个人验证码错误处理（Invalid PIN Process，图中U2），所以（1）处应填<<extend>>。

【问题2】

序列图是场景的图形化表示，描述了以时间顺序组织的对象之间的交互活动。构造序列图时遵循如下指导原则：确定顺序图的范围，描述这个用例场景或一个步骤；绘制参与者和接口类，如果范围包括这些内容的话；沿左手边列出用例步骤；对控制器类及必须在顺序中协作的每个实体类，基于它拥有的属性或已经分配给它的行为绘制框；为持续类和系统类绘制框；绘制所需消息，并把每条消息指到将实现响应消息的责任的类上；添加活动条指示每个对象实例的生命期；为清晰起见，添加所需的返回消息；如果需要，为循环、可选步骤和替代步骤等添加框架。

本题中，根据说明中的描述，从ATM机判断卡已插入（ cardInserted() ）开始会话，即为当前ATM创建会话（ create(this) ）并开始执行会话(performSessiono)；读卡器读片（ readCard() ）获得ATM卡信息（ card ），然后从控制台读取个人验证码输入（ readPIN()，图中标号6处 ）并获得个人验证码信息（ PIN，图中标号7处 ）；然后根据用户选择启动并执行事务，即为当前会话创建事务（ creat(atm, this, card, pin) ，图中标号8处 ）和执行事务（ performTransaction()，图中标号9处 ）；可以选择继续执行某个事务（ doAgain ）循环，或者选择退卡（ ejectCard() ）。

【问题3】

用例之间的继承关系表示子类型“是一种”父类型。其中父类型通常是一个抽象泛化用例，具有子类型共有的属性和行为，每个具体的子类型继承它，并实现适合自己的特定的操作。

本题中Transaction和Withdraw、Deposit等四个用例之间的关系即为继承关系，Transaction即是一个抽象泛化用例，具有其他事务类型共有的属性和行为，每个具体的事务类型继承它，并实现适合自己的特定的操作。

试题8答案

【问题1】

A1 : Customer A2 : Bank U1 : Session

U2 : Invalid PIN Process U3 : Transaction (1) : <<extend>>

【问题2】

6 : readPIN() 7 : PIN 8 : creat(atm, this, card, pin)

9 : performTransaction()

【问题3】

Transaction是一个抽象泛化用例，具有其他事务类型共有的属性和行为，每个具体的事务类型继承它，并实现适合自己的特定的操作。

版权所有希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第 14 章：算法设计

作者：希赛教育软考学院 来源：希赛网 2014年05月06日

考点突破

根据考试大纲，本章要求考生掌握以下几个方面的知识点。

（1）数据结构设计：线性表、查找表、树、图的顺序存储结构和链表存储结构的设计和实现。

（2）算法设计：迭代、穷举搜索、递推、递归、回溯、贪心、动态规划、分治等算法设计。

从历年的考试情况来看，本章的考点主要集中以下方面。

在数据结构设计中，主要考查基本数据结构如栈，二叉树的常见操作代码实现。

在算法设计中，主要考查动态规划法、分治法、回溯法、递归法、贪心法。

版权所有希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)