第 5 章　面向对象系统分析与设计

5.1　大纲要求

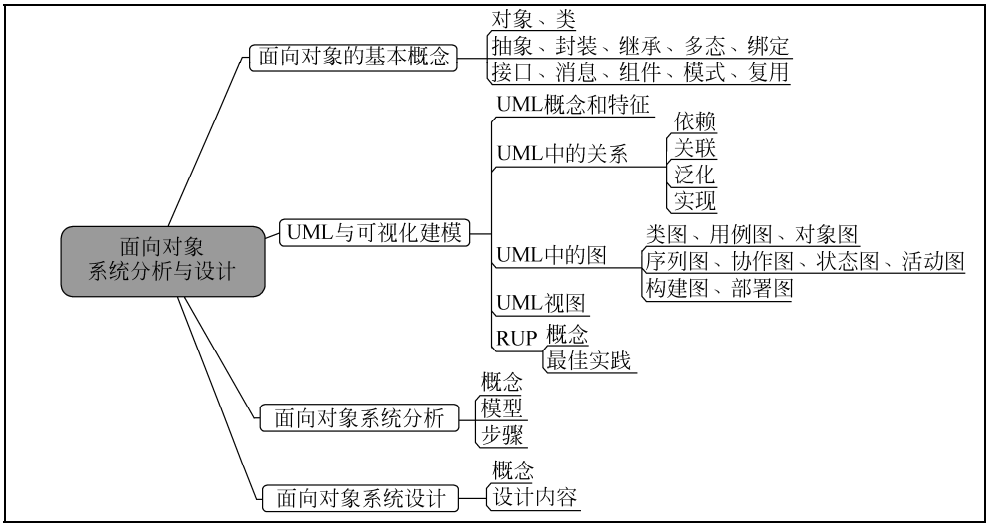
考试大纲中对本章的要求有：

- 面向对象的基本概念
- 统一建模语言 UML 与可视化建模
- 面向对象的系统分析
- 面向对象的系统设计

根据考试大纲及历年考试情况分析，本章重点知识包括：

- 面向对象的基本概念的理解
- UML 的特征理解
- UML 中的关系
- UML 中的 9 种图的定义
- UML 中的视图
- RUP 的概念和最佳实践
- 面向对象系统分析的概念、模型和步骤
- 面向对象系统设计的概念和内容

5.2　知识结构图



5.3 要点详解

5.3.1 面向对象的基本概念

面向对象是一种运用对象、类、继承、封装、聚合、消息传递、多态性等概念来构造系统的软件开发方法。面向对象的主要思想是将现实世界中的对象映射为问题域中的要素，从而有效地连接计算机和现实问题域。面向对象的基本概念有对象、类、抽象、封装、继承、多态、消息、模式和复用等。

1．对象

对象是由数据及其操作所构成的封装体，是系统中用来描述客观事物的一个模块，是构成系统的基本单位。采用计算机语言描述，对象是由一组属性和对这组属性进行的操作构成的。

对象的三要素是对象标识、对象状态和对象行为。例如对于教师 Joe，其个人状态信息包括性别、年龄和职位等，其行为特征为授课，则 Joe 就是封装后的一个典型对象。

2．类

类是现实世界中实体的形式化描述，类将该实体的数据和函数封装在一起。类的数据也叫属性、状态或特征，它表现类静态的一面。类的函数也叫功能、操作或服务，它表现类动态的一面。

如教师类中，教师共同的状态通过属性表现出来，共同的行为通过操作表现出来，如图 5.1 所示。

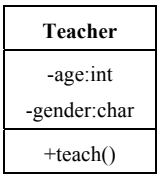


图 5.1 教师类的属性和操作

对象是类的实际例子，如将所有教师抽象为教师类，则教师 Joe 是教师类的一个对象。

类和对象的关系如下：

- 每个对象都是某个类的实例。
- 每个类在某一时刻都有零或更多的实例。
- 类是静态的，它们的存在、语义和关系在程序执行前就已经定义好了；对象是动态的，它们在程序执行时可以被创建和删除。
- 类是生成对象的模板。

3．抽象

抽象是通过特定的实例抽取共同特征以后形成概念的过程。它强调主要特征，忽略次要特征。一个对象是现实世界中一个实体的抽象，一个类是一组对象的抽象，抽象是一种单一化的描述，它强调给出与应用相关的特性，抛弃不相关的特性。

4．封装

封装是将相关的概念组成一个单元，然后通过一个名称来引用它。面向对象封装是将数据和基于数据的操作封装成一个整体对象，对数据的访问或修改只能通过对象对外提供的接口进行。

5．继承

继承表示类之间的层次关系，这种关系使得某类对象可以继承另外一类对象的属性和操作。继承又可分为单继承和多继承，单继承是子类只从一个父类继承，而多继承中的子类可以从多于一个的父类继承，Java 是单继承的语言，而 C++允许多继承。

如类 B 继承了类 A，则类 B 中的对象具有类 A 的一切特征，类 B 还可以有一些扩展。类 A 称为基类、父类或超类，类 B 称为类 A 的派生类或子类。

6．多态

多态性是一种方法，这种方法使得在多个类中可以定义同一个操作或属性名，并在每个类中可以有不同的实现。多态使得某个属性或操作在不同的时期可以表示不同类的对象特征。

7．动态绑定

绑定是一个把过程调用和响应调用所需要执行的代码加以结合的过程。在一般的程序设计语言中，绑定是在编译时进行的，叫作静态绑定。动态绑定则是在运行时进行的，因此，一个给定的过程调用和代码的结合直到调用发生时才进行。

动态绑定是和类的继承以及多态相联系的。在继承关系中，子类是父类的一个特例，所以父类对象可以出现的地方，子类对象也可以出现。因此在运行过程中，当一个对象发送消息请求服务时，要根据接收对象的具体情况将请求的操作与实现的方法进行连接，即动态绑定。

8．接口

接口就是对操作规范的说明。接口只是说明操作应该做什么，但没有定义操作如何做。接口可以理解为类的一个特例，它只规定实现此接口的类的操作方法，而把实现细节交由实现该接口的类完成。

9．消息

消息体现对象间的交互，通过它向目标对象发送操作请求。

10．组件

组件也叫构件。组件是软件系统可替换的、物理的组成部分，封装了模块功能的实现。组件应当是内聚的，并且具有相对稳定的公开接口。

11．模式

每个模式描述了一个不断重复发生的问题，以及该问题的解决方案。它是一条由三部分组成的规则，表示了一个特定环境、一个问题和一个解决方案之间的关系。

设计模式通常是对于某一类软件设计问题的可重用的解决方案。设计模式使人们可以更加简单和方便地去复用成功的软件设计和体系结构，能够帮助设计者更快更好地完成系统设计。

设计模式的四要素：

- 模式名称（Pattern Name）。
- 问题（Problem）。
- 解决方案（Solution）。
- 效果（Consequences）。

设计模式确定了所包含的类和实例，它们的角色、协作方式以及职责分配。按照设计模式的目的可以分为三类：

- 创建型模式：与对象的创建有关。
- 结构型模式：处理类或对象的组合。
- 行为型模式：对类或对象怎样交互和怎样分配职责进行描述。

12．复用

软件复用是指将已有的软件及其有效成分用于构造新的软件或系统。构件技术是软件复用实现的关键。

5.3.2 统一建模语言与可视化建模

1．UML 概述

1）概念

统一建模语言（Unified Modeling Language，UML）是一种通用的可视化建模语言，它是面向对象分析和设计的一种标准化表示，用于对软件进行描述、可视化处理、构造和建立软件系统的文档。

UML 描述了系统的静态结构和动态行为，它将系统描述为一些独立的相互作用的对象，构成为外界提供一定功能的模型结构，静态结构定义了系统中重要对象的属性和服务，以及这些对象之间的相互关系，动态行为定义了对象的时间特性和对象为完成目标而相互进行通信的机制。

2）特征

UML 具有如下特征：

- 不是一种可视化的程序设计语言，而是一种可视化的建模语言。
- 是一种建模语言规范说明，是面向对象分析与设计的一种标准表示。

- 不是过程，也不是方法，但允许任何一种过程和方法使用它。
- 简单且可扩展，具有扩展和专有化机制，便于扩展，无须对核心概念进行修改。
- 为面向对象的设计与开发中涌现出的高级概念（如协作、框架、模式和组件）提供支持，强调在软件开发中对架构、框架、模式和组件的重用。
- 与最佳软件工程实践经验进行了集成。

2．UML 中的关系

UML 中有四种关系：依赖、关联、泛化和实现。

1）依赖（Dependency）

依赖表示两个事物间的语义关系，其中一个事物（独立事物）发生变化会影响另一个事物（依赖事物）的语义。用可能有方向的虚线表示，如图 5.2 所示。

图 5.2　依赖关系

2）关联（Association）

关联是一种结构关系，它描述了一组链，链是对象之间的连接。聚集（Aggregation）是一种特殊类型的关联，它描述了整体和部分间的结构关系。关联用图 5.3 表示，可以标注重复度和角色。

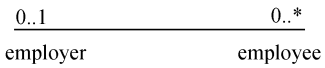


图 5.3　关联关系

聚集的图形化表示如图 5.4 所示。

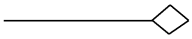


图 5.4　聚集的图形化表示

3）泛化（Generalization）

泛化是一种特殊/一般关系，特殊元素（子元素）的对象可替代一般元素（父元素）的对象。用这种方法，子元素共享了父元素的结构和行为。泛化关系在图形上表示为一条带有空心箭头的实线，指向父元素，如图 5.5 所示。

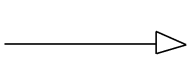


图 5.5　泛化关系

4）实现（Realization）

实现是类元之间的语义关系，其中一个类元指定了由另一个类元保证执行的契约。有两种情况需要用到实现关系：一种是在接口和实现它们的类或构件之间；另一种是在用例和实现它们的协作之间。实现关系在图形上用一条带有空心箭头的虚线表示，如图 5.6 所示。

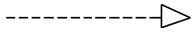


图 5.6　实现关系

3．UML 中的图

UML 提供了 9 种主要的图来对待建系统进行建模。

1）类图（Class Diagram）

类图显示了一组对象、接口、协作和它们之间的关系。在面向对象系统的建模中所建立的最常见的图就是类图。类图给出系统的静态设计视图，如图 5.7 所示。

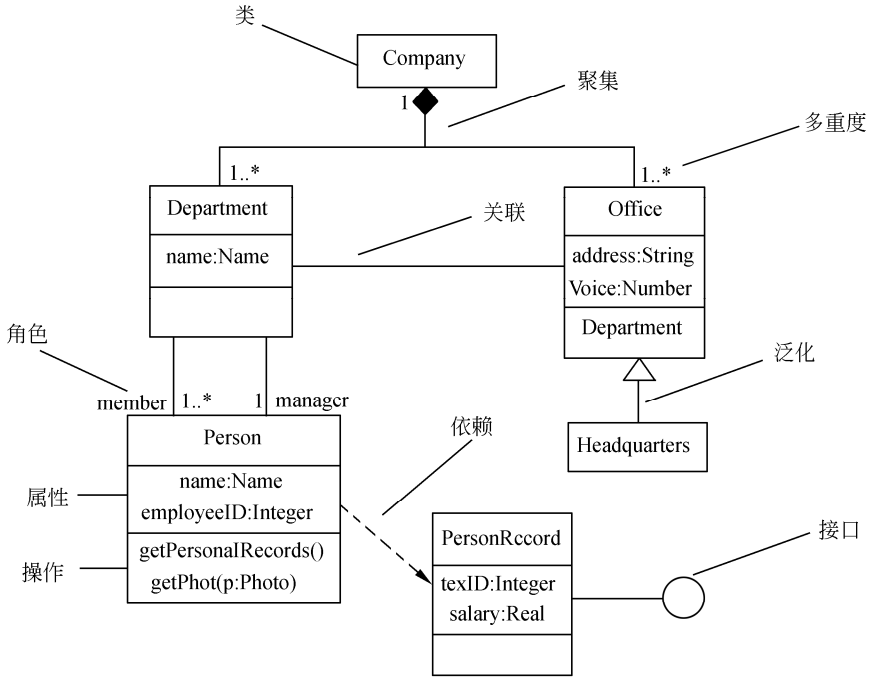


图 5.7　类图

2）对象图（Object Diagram）

对象图显示了一组对象以及它们之间的关系。对象图描述了在类图中所建立的事物的

实例的静态快照。和类图一样，对象图给出系统的静态设计视图或静态进程视图，但它们是从真实的或原型案例的角度建立的。这种视图主要支持系统的功能需求。利用对象图可以对静态数据结构建模。

3）用例图（Use Case Diagram）

用例图显示了一组用例、参与者（actor）以及它们之间的关系。用例图通常包括用例、参与者、扩展关系和包含关系，如图 5.8 所示。

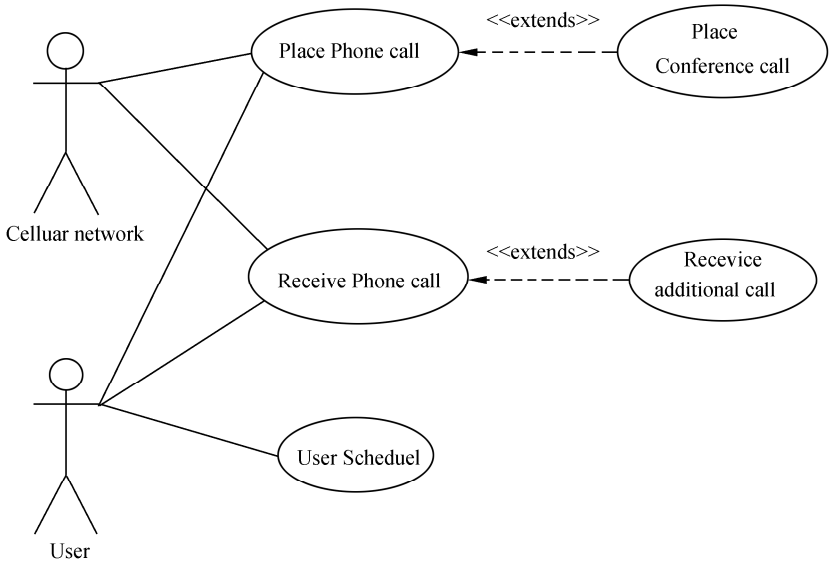


图 5.8　用例图

包含（include）关系为用例建模提供了从两个或更多用例的描述中抽取通用部分的能力。一般情况下，如果若干个用例的某些行为是相同的，则可以把这些相同的行为提取出来作为一个单独的用例，这个用例称作抽象用例，其他用例可以包含该抽象用例。所以，在描述用例之前就开始抽取包含用例是不可取的。在 UML 的较早版本中也有 uses 关系，在 UML 2.2 中 uses 和 includes 被 include 取代，称为包含关系。

扩展（extend）关系提供了使用另外的可选流程来补充或插入到一个已存在的用例中的能力。因此，这是一种能够扩展原用例却不用对原用例进行重新描述的方法。

包含关系和扩展关系的区别：

- 包含关系中，对基用例来说，如果缺少了被包含用例，则基用例不完整；扩展关系中，如果去掉扩展关系，基用例仍然完整。
- 包含关系中，被包含用例对基用例是可见的；扩展关系中，基用例对扩展用例可见，而扩展用例对基用例不可见。

- 扩展关系中，扩展用例本身具有独立的功能，而非从其他用例中抽取。
- 包含关系中，被包含用例通常应被两个以上的其他用例所包含。

用例图用于对系统的静态用例视图进行建模。这个视图主要支持系统的行为，即该系统在它的周边环境的语境中所提供的外部可见服务。

4）交互图

序列图和协作图均被称为交互图，它们用于对系统的动态方面进行建模。一张交互图显示的是一个交互，由一组对象和它们之间的关系组成，包含它们之间可能传递的消息。序列图是强调消息时间顺序的交互图；协作图是强调接收和发送消息的对象的结构组织的交互图。交互图一般包含对象、链和消息。

（1）序列图（Sequence Diagram）。

序列图是场景的图形化表示，描述了以时间顺序组织对象之间的交互活动，如图 5.9 所示。

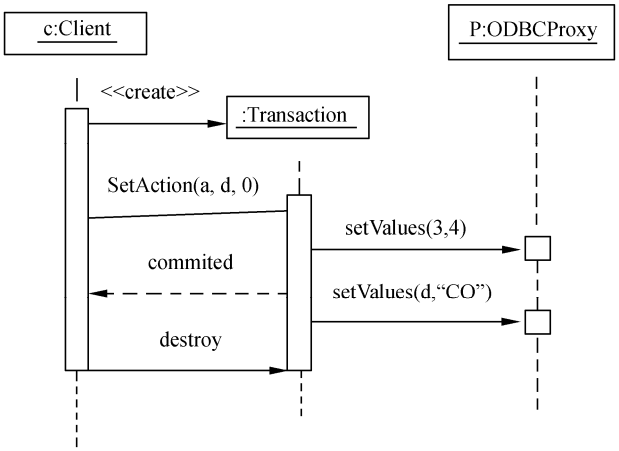


图 5.9 序列图

序列图有两个不同于协作图的特征：

- 序列图有对象生命线。对象生命线是一条垂直的虚线，表示一个对象在一段时间内存在。
- 序列图有控制焦点。控制焦点是一个瘦高的矩形，表示一个对象执行一个动作所经历的时间段，既可以是直接执行，也可以是通过下级过程执行。

（2）协作图（Collaboration Diagram）。

协作图强调收发消息的对象的结构组织。协作图有两个不同于序列图的特征：

- 协作图有路径。为了指定一个对象如何与另一个对象链接，可以在链的末端附上一个路径构造型。通常只需要显式地表示 local（局部）、parameter（参数）、global

（全局）以及 self（自身）这几种链的路径，不必表示 association（关联）。

- 协作图有顺序号。为表示一个消息的时间顺序，可以给消息加一个数字前缀（从 1 号开始），在控制流中，每个新消息的顺序号单调增加（如 2、3 等）。为了显示嵌套，可使用带小数点的号码（1 表示第一个消息，1.1 表示嵌套在消息 1 中的第一个消息，等等）。嵌套可为任意深度。另外，沿同一个链可以显示许多消息，且每个消息都有唯一一个顺序号。

协作图和序列图是同构的，它们之间可以相互转换。

5）状态图（Statechart Diagram）

状态图显示了一个状态机，它由状态、转换、事件和活动组成。状态图关注系统的动态视图，它对于接口、类和协作的行为建模尤为重要，强调对象行为的事件顺序。状态图通常包括简单状态和组合状态、转换（事件和动作），如图 5.10 所示。

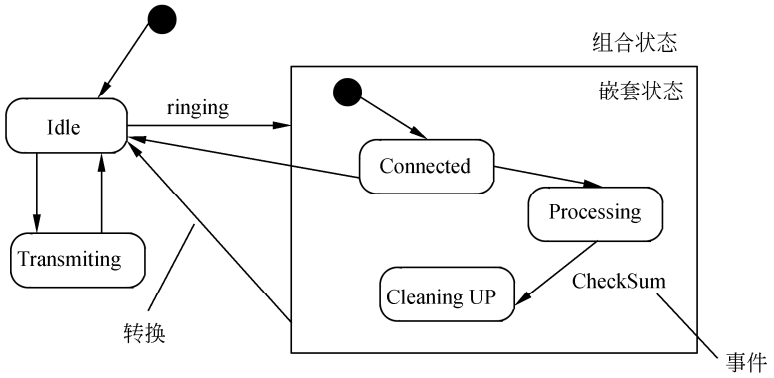


图 5.10　状态图

6）活动图（Activity Diagram）

活动图是一种特殊的状态图，它显示了在系统内从一个活动到另一个活动的流程。活动图专注于系统的动态视图，它对于系统的功能建模特别重要，并强调对象间的控制流程。活动图一般包括活动状态和动作状态、转换和对象。

活动图可以表示分支和汇合。当为一个系统的动态建模时，通常有两种使用活动图的方式：

- 对工作流建模：此时所关注的是与系统进行协作的参与者所观察到的活动。
- 对操作建模：此时把活动图作为流程图使用。

7）构件图（Component Diagram）

构件图显示了一组构件之间的组织和依赖。构件图关注系统的静态实现视图，它与类图相关，通常把构件映射为一个或多个类、接口或协作。

8）部署图（Deployment Diagram）

部署图显示了运行处理节点以及其中构件的配置。部署图给出了体系结构的静态实施视图。它与构件图相关，通常一个节点包含一个或多个构件。

4．UML 视图

为方便起见，可用视图来划分 UML 中的概念和组件。视图只是表达系统某一方面特征的 UML 建模组件的子集，在每一类视图中使用一种或多种特定的图来可视化地表示视图中的各种概念。

表 5.1 列出了 UML 的视图和视图所包括的图以及每种图有关的主要概念。

表 5.1 UML 视图

主要的域	视　　图	图	主要的概念
结构	静态视图	类图	类、关联、泛化、依赖关系、实现、接口
	用例视图	用例图	用例、参与者、关联、扩展、包含、用例泛化
	实现视图	构件图	构件、接口、依赖关系、实现
	部署视图	部署图	节点、构件、依赖关系、位置
动态	状态机视图	状态机图	状态、事件、转换、动作
	活动视图	活动图	状态、活动、完成转换、分叉、结合
	交互视图	顺序图	交互、对象、消息、激活
		协作图	协作、交互、协作角色、消息
模型管理	模型管理视图	类图	包、子系统、模型
可扩展性	所有	所有	约束、构造型、标记值

5．RUP

RUP 是使用面向对象技术进行软件开发的最佳实践之一，是软件工程的过程。它对所有关键开发活动提供了使用准则、模板及工具等。

RUP 有六个基本最佳实践，分别如下：

- 迭代式开发：RUP 支持专注于处理生命周期每个阶段中最高风险的迭代开发方法，极大地减少了项目的风险性。
- 需求管理：RUP 描述了如何提取、组织和文档化需要的功能和限制。
- 使用基于构件的体系结构：RUP 提供了使用新的及现有构件定义体系结构的系统化方法。
- 可视化软件建模：RUP 开发过程显示了对软件如何可视化建模、捕获体系结构及构件的构架和行为。
- 验证软件质量：RUP 帮助计划、设计、实现、执行和评估软件质量，并且不是事后型的。
- 控制软件变更：RUP 开发过程描述了如何控制、跟踪和监控修改以确保成功的迭代开发。

5.3.3　面向对象系统分析

面向对象系统分析（OOA）运用面向对象方法分析问题域，建立基于对象、消息的业务模型，形成对客观世界和业务本身的正确认识。

1．面向对象系统分析的模型

面向对象系统分析的模型由用例模型、类-对象模型、对象-关系模型和对象-行为模型组成。

- 用例模型：可用若干幅用例图组成。用例描述了用户和系统之间的交互，其重点是系统为用户做什么。用例模型描述全部的系统功能行为。
- 类-对象模型：描述系统所涉及的全部类以及对象。每个类和对象都通过属性、操作来进行进一步描述。
- 对象-关系模型：描述对象之间的静态关系，同时定义了系统中所有重要的消息路径，它也可以具体化到对象的属性、操作。对象-关系模型包括类图和对象图。
- 对象-行为模型：描述了系统的动态行为，包括状态图、顺序图、协作图和活动图。

2．面向对象分析步骤

（1）发现角色/参与者。
（2）发现用例。
（3）建立用例模型。
（4）进行领域分析。
（5）建立对象-关系模型。
（6）建立对象-行为模型。
（7）建立功能模型。

5.3.4　面向对象系统设计

面向对象系统设计（OOD）对系统分析得出的问题域模型，用面向对象的方法设计出软件基础架构（概要设计）和完整的类结构（详细设计），以实现业务功能。

面向对象系统设计主要包括用例设计、类设计和子系统设计等。

5.4　真题分析

（1）以下关于对象、类和继承的叙述中，不正确的是_____。

A．对象是系统中用来描述客观事务的一个模块，是构成系统的基本单位
B．类是现实世界中实体的形式化描述
C．对象是类的实例，类是对象的模板
D．继承表示对象之间的层次关系

试题分析

继承表示类之间的层次关系，而不是对象之间的层次关系。

参考答案 D

（2）RUP 模型是一种过程方法，它属于_____的一种。

A．瀑布模型　　B．V 模型　　C．螺旋模型　　D．迭代模型

试题分析

迭代式开发是 RUP 的最佳实践，所以 RUP 模型属于迭代模型的一种。

参考答案 D

（3）以下关于面向对象方法的描述中，不正确的是_____。

A．选择面向对象程序设计语言时需要考虑开发人员对其熟悉程度

B．使用设计模式有助于在软件开发过程中应用对象技术

C．在软件生命周期的分析、设计、实现和测试过程中均可以应用面向对象技术

D．UML 是一种可视化建模语言，它需要与 RUP 开发过程同时使用

试题分析

UML 是一种可视化建模语言，它可以和多种软件开发过程结合使用，除了 RUP 开发过程以外，还有敏捷开发过程、CMMI 模型等。

参考答案 D

（4）如果在一个课程注册系统中，定义了类 CourseSchedule 和类 Course，并且在类 CourseSchedule 中定义了方法 Add（c：Course）和方法 Remove（c：Course），那么类 CourseSchedule 和类 Course 之间的是一种_____关系。

A．包含　　B．实现　　C．依赖　　D．泛化

试题分析

依赖关系表示两个事物间的语义关系，其中一个事物（独立事物）发生变化会影响另一个事物（依赖事物）的语义。

参考答案 C

（5）在面向对象分析中，其分析过程的第一步是_____。

A．发现角色/参与者　　B．发现用例

C．进行领域分析　　D．建立功能模型

试题分析

面向对象分析的步骤包括：发现角色/参与者、发现用例、建立用例模型、进行领域分析、建立对象-关系模型、建立对象-行为模型和建立功能模型。所以分析过程的第一步为“发现角色/参与者”。

参考答案 A

（6）“容器是一个构件，构件不一定是容器；一个容器可以包含一个或多个构件，一个构件只能包含在一个容器中”。根据上述描述，如果用 UML 类图对容器和构件之间的关

系进行面向对象分析和建模，则容器类和构件类之间存在_____关系。

① 继承　②扩展　③聚集　④ 包含

A．① ②　B．② ④　C．① ④　D．① ③

试题分析

“容器是一个构件”说明容器类和构件类之间存在继承关系，“一个容器可以包含一个或多个构件”说明容器类和构件类之间存在聚集关系。包含和扩展是用例之间的关系。

参考答案　D

（7）面向对象分析与设计技术中，_____是类的一个实例。

A．对象　B．接口　C．构件　D．设计模式

试题分析

面向对象的分析与设计技术中，对象是类的一个实例，类是生成对象的模板。

参考答案　A

（8）UML 中的用例和用例图的主要用途是描述系统的_____。

A．功能需求　B．详细设计　C．体系结构　D．内部接口

试题分析

用例是在不展现一个系统或子系统内部结构的情况下，对系统或子系统的某个连贯的功能单元的定义和描述。用例是对系统功能的描述，在使用 UML 的开发过程中，用用例来表达需求。

参考答案　A

（9）在用例设计中，可以使用 UML 中的_____来描述用户和系统之间的交互，说明系统功能行为。

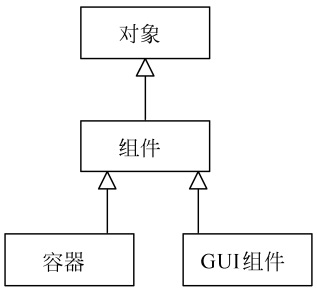
A．序列图　B．构件图　C．类图　D．部署图

试题分析

序列图是场景的图形化表示，描述了以时间顺序组织对象之间的交互活动。

参考答案　A

（10）根据下面的 UML 类图，以下叙述中_____是不正确的。



A．容器是一个组件　　B．GUI 组件就是一个容器

C．GUI 组件是一个对象　　D．容器和 GUI 组件都是组件

试题分析

从 UML 类图中可以看出，类之间的关系都是泛化关系，泛化关系是一种特殊/一般关系，特殊元素（子元素）的对象可替代一般元素（父元素）的对象。用这种方法，子元素共享了父元素的结构和行为，图形上表示为一条带有空心箭头的实线，指向父元素。图中组件是对象的子元素，容器和 GUI 组件是组件的子元素，所以 A、C、D 说法都正确，B 说法错误。

参考答案　B

（11）用于显示运行的处理节点以及居于其上的构件、进程和对象的配置的图是_____。

A．用例图　　B．部署图　　C．类图　　D．构件图

试题分析

部署图显示了运行处理节点以及其中构件的配置。

参考答案　B

（12）关于 UML，错误的说法是_____。

A．UML 是一种可视化的程序设计语言

B．UML 不是过程，也不是方法，但允许任何一种过程和方法的使用

C．UML 简单且可扩展

D．UML 是面向对象分析与设计的一种标准表示

试题分析

UML 不是一种可视化的程序设计语言，而是一种通用的可视化建模语言。

参考答案　A

（13）在 UML 中，动态行为描述了系统随时间变化的行为，下面不属于动态行为视图的是_____。

A．状态机视图　　B．实现视图

C．交互视图　　D．活动视图

试题分析

在 UML 视图中，属于动态行为视图的有状态机视图、交互视图和活动视图。

参考答案　B

（14）面向对象中的_____机制是对现实世界中遗传现象的模拟。通过该机制，基类的属性和方法被遗传给派生类。

A．复用　　B．消息　　C．继承　　D．变异

试题分析

继承表示类之间的层次关系，这种关系使得某类对象可以继承另外一类对象的属性和操作。这种机制类似于现实世界中的遗传现象，子类继承了父类的属性和操作，即父类的

属性和方法被遗传给了子类。

参考答案　C

（15）_____是指把数据以及操作数据的相关方法组合在同一单元中，使我们可以把类作为软件复用中的基本单元，提高内聚度，降低耦合度。

A．多态　　B．封装　　C．抽象　　D．接口

试题分析

封装是将相关的概念组成一个单元，然后通过一个名称来引用它。面向对象封装是将数据和基于数据的操作封装成一个整体对象，对数据的访问或修改只能通过对象对外提供的接口进行。通过封装，我们可以把类作为软件复用中的基本单元，提高内聚度，降低耦合度。

参考答案　B

5.5　本章练习

（1）下列关于面向对象的分析与设计的描述，正确的是_____。

A．面向对象设计描述软件要做什么

B．面向对象分析不需要考虑技术和实现层面的细节

C．面向对象分析的输入是面向对象设计的结果

D．面向对象设计的结果是简单的分析模型

（2）以下陈述正确的是_____。

A．对象是类的实例　　B．类是对象集合的抽象定义

C．对象有生命期　　D．以上都对

（3）雇员类含有计算报酬的行为，利用面向对象的_____可以使其派生类专职雇员类和兼职雇员类计算报酬的行为有相同的名称、不同的计算方法。

A．多态性　　B．继承性　　C．封装性　　D．复用性

（4）某订单处理系统中，“创建新订单”和“更新订单”两个用例都需要检查客户的账号是否正确，为此定义一个通用的用例“检查客户账户”，用例“创建新订单”和“更新订单”与“检查客户账户”之间是_____。

A．包含关系　　B．聚合关系　　C．泛化关系　　D．关联关系

（5）下面_____不是对象的特性。

A．状态　　B．行为　　C．标识　　D．多态

（6）_____可以帮助人们简单方便地复用已经成功的设计或体系结构。

A．商业构件　　B．设计模式　　C．遗留系统　　D．需求规格说明

（7）下列关于 UML 叙述正确的是_____。

A．UML 是一种语言，语言的使用者不能对其扩展

B．UML 仅是一组图形的集合

C．UML 仅适用于系统的分析与设计阶段

D．UML 是独立于软件开发过程的

（8）属于 RUP 最佳软件开发实践的是_____。

A．迭代式开发/控制变更

B．管理需求/验证软件质量

C．可视化建模/分层架构

D．以上都是

（9）某软件公司欲开发一个在线交易系统。为了能够精确表达用户与系统的复杂交互过程，应该采用 UML 的_____进行交互过程建模。

A．类图　　B．部署图　　C．序列图　　D．对象图

（10）当_____时，用例是捕获系统需求最好的选择。

A．系统具有很少的用户　　B．系统具有很少的接口

C．系统算法复杂，功能单一　　D．系统有很多参与者