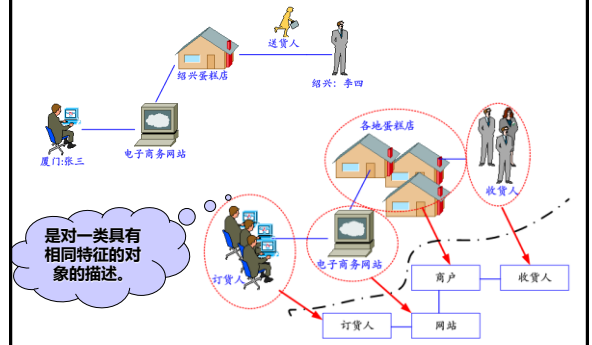




类图

1

类



类图

类图描述了系统中的类及其相互之间的各种关系，其本质是反映了系统中包含的各种对象类型以及对象间的各种静态关系。

3

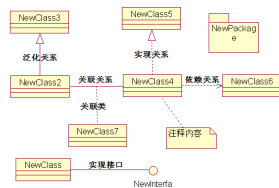
类图

类图的含义

- 类图 (Class diagram) 显示了系统的静态结构，而系统的静态结构构成了系统的概念基础。
- 类图，就是用于对系统中的各种概念进行建模，并描绘出它们之间关系的图。
- 在大多数的 UML 模型中，我们可以将这些概念的类型概括为以下四种，分别是：
 - (1) 类
 - (2) 接口
 - (3) 数据类型
 - (4) 构件

类图

- 在类图中，具体来讲它一共包含了以下几种模型元素，分别是：类、接口、泛化关系、关联关系以及实现关系。
- 类图可以创建约束、注释和包等。



类图的抽象层次

可以使用UML的版型机制标明类图的抽象层次

Circle	概念层类图, 描述应用领域的概念, 与实现它的软件没有直接的联系			
Circle	说明层类图, 描述软件的接口部分, 不是实现部分。			
-center -radius +area() +move() +scale()				
<table> <tr> <td><<implementation>></td> </tr> <tr> <td>Circle</td> </tr> <tr> <td> -center: float -radius: float <<create>>+Circle() <<query>>+area(): float <<update>>+move(location: Point): void <<update>>+scale(ratio: float): void </td> </tr> </table>		<<implementation>>	Circle	-center: float -radius: float <<create>>+Circle() <<query>>+area(): float <<update>>+move(location: Point): void <<update>>+scale(ratio: float): void
<<implementation>>				
Circle				
-center: float -radius: float <<create>>+Circle() <<query>>+area(): float <<update>>+move(location: Point): void <<update>>+scale(ratio: float): void				

实现层类图考虑实现问题, 提供类的细节。

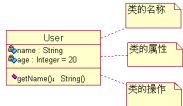
6

类图的作用

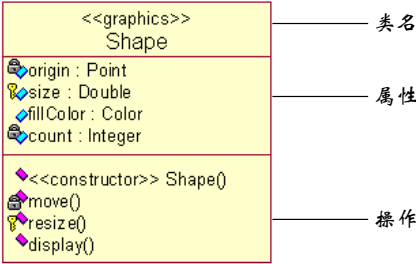
- 类图的作用是对系统的静态视图进行建模。当对系统的静态视图进行建模时，通常是以以下三种方式来使用类图。
 - (1)为系统的词汇建模。
 - (2)模型化简单的协作。
 - (3)模型化逻辑数据库模式。
- 在设计数据库时，通常将数据库模式看作为数据库概念设计的蓝图。在很多领域中，都需要在关系数据库或面向数据库中存储永久信息。系统分析者可以使用类图来对这些数据库进行模式建模。

类图的组成

1. 类
- 类是面向对象系统组织结构的核心。类是对一组具有相同属性、操作、关系和语义的事物的抽象。
 - 在UML的图形表示中，类的表示法是一个矩形，这个矩形由三个部分构成，分别是：类的名称 (Name)、类的属性 (Attribute) 和类的操作 (Operation)。



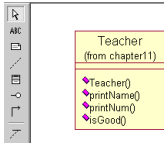
类的定义



9

类的名称

- 类的名称是每个类的图形中所必须拥有的元素，用于同其它类进行区分。类的名称通常来自于系统的问题域，并且尽可能地明确表达要描述的事物，不会造成类的语义冲突。



类的名称

- 应该来自系统的问题域。
- 应该是一个名词，且不应该有前缀或后缀。
- 分为简单名称和路径名称。



11

类的属性

[可见性]属性名[:类型][['多重性[次序]']][=初值][{特性}]




可见性	UML	Rose
private	-	
protected	#	
public	+	

+size: area = (100,100)
#visibility: Boolean = false
-colors: Color[3]
-points: Point[2..* ordered]
name: String[0..1]

12

类的操作

[可见性]操作名[(参数列表)][: 返回值类型][(特性)]

可见性	UML	Rose
private	-	
protected	#	
public	+	

```
+display(): Location
+hide()
#create()
-attachXWindow(xwing: XwindowPtr)
```

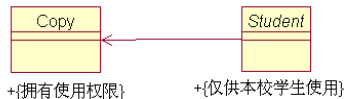
13

类的职责

- 在标准的UML定义中，有时还应当指明类的另一种信息，那就是类的职责。类的职责指的是对该类的所有对象所具备的那些相同的属性和操作共同组成的功能或服务的抽象。
- 在声明类的职责的时候，可以非正式的在类图的下方增加一栏，将该类的职责逐条描述出来。类的职责的描述并不是必须的，因此也可以将其作为文档的形式存在，也就是说类的职责其实只是一段或多段文本描述。一个类可以有多种职责，设计得好的类一般至少有一种职责。

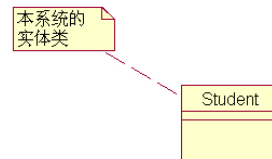
类的约束

- 类的约束指定了该类所要满足的一个或多个规则。在UML中，约束是用一个大括号括起来的文本信息。



类的注释

- 类的注释



边界类、控制类和实体类

UML中三种主要的类版型

- 边界类, boundary class
- 控制类, control class
- 实体类, entity class

引入多种类版型帮助分析和设计人员确定系统中的类。

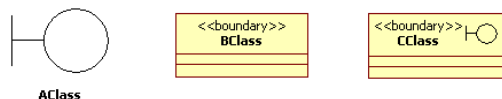
17

边界类

边界类位于系统与外界的交界处,包括:

- 用户界面类,如:窗口、对话框、报表类等
- 通讯协议类,如:TCP/IP的类
- 直接与外部设备交互的类
- 直接与外部系统交互的类

边界类的UML表示方法:

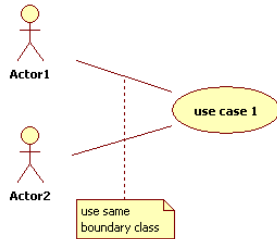


18

边界类

通过用例图可以确定需要的边界类, 每个Actor/User case对至少需要一个边界类.

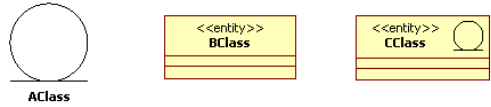
但并不是每个Actor/Use case都需要生成惟一边界类, 多个actor启动同一-use case可以使用同一边界类.



19

实体类

实体类保存要放进持久存储体(数据库/文件等)的信息.

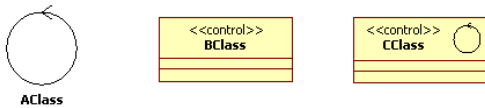


实体类通过事件流和交互图发现, 采用目标领域术语命名. 通常实体类对应数据库中的表, 其属性对应表的字段, 但实体类与数据库中的表不一定是一一对应关系.

20

控制类

控制类是负责管理或控制其他类工作的类.



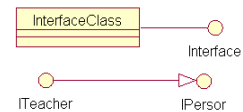
每个用例通常有一个控制类, 控制用例中的事件顺序, 控制类也可以在多个用例间共用.

21

类图的组成

2. 接口

- 类接口是在没有给出对象的实现和状态的情况下对对象行为的描述. 通常, 在接口中包含一系列操作但是不包含属性, 并且它没有对外界可见的关联.
- 接口是一种特殊的类, 所有接口都是有构造型<<interface>>的类. 一个类可以通过实现接口从而支持接口所指定的行为.
- 在UML中, 接口的表示方式是使用一个带有名称的小圆圈来进行表示的, 并且我们可以通过一条Realize (实现关系) 线与其实现的类相连接



选择题

- 在类图中, “+”表示的可见性是 ()
 - A. Public
 - B. Protected
 - C. Private
 - D. Package

• 选A

23

选择题

- 在类图中, “#”表示的可见性是 ()
 - A. Public
 - B. Protected
 - C. Private
 - D. Package

• 选B

24

选择题

- 类通常可以分为实体类，（ ）和边界类
 - A. 父类
 - B. 子类
 - C. 控制类
 - D. 祖先类

• 选C

25

选择题

- （ ）是一组用于描述类或组件的一个服务的操作
 - A. 包
 - B. 节点
 - C. 接口
 - D. 组件

• 选C

26

填空题

- 接口是在整个模型中反复使用的一组行为，是一个没有_____而只有方法的类。

• 属性

27

填空题

- 对象模型描述了系统的静态结构，通常使用UML提供的_____图来描述。

• 类图

28

填空题

- UML提供一系列的图支持面向对象的分析与设计,其中_____给出系统的静态设计视图。

• 类图

29

填空题

- 在类图中一共包含了以下几种元素，分别是:类、_____, 关系、协作、注释、约束。

• 接口

30

填空题

- 可以用概念类类别表和_____两种方法找概念类。
- 标识名词短语

31

习题

- 设有1个学生类(Student)，包括私有变量学号(xh)和姓名(xm)，都是string 类型的；有1个受保护的变量stdType，为整形；有四个公开的方法，分别是setXm, setXh, getXh, getXm。请画出这个类的UML图。

32

类之间的关系

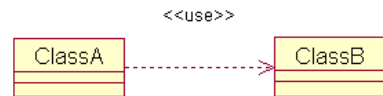
常见的类之间的关系包括：依赖、关联、泛化、实现等。

33

依赖关系

依赖关系：表示的是两个或多个模型元素之间语义上的连接关系。它只将模型元素本身连接起来而不需要用一组实例来表达它的意思。

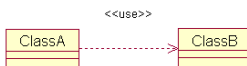
它表示了这样一种情形，提供者的某些变化会要求或指示依赖关系中客户的变化。也就是说依赖关系将行为和实现与影响其他类的类联系起来。



34

依赖关系

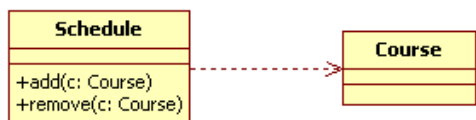
- 依赖关系：表示的是两个或多个模型元素之间语义上的连接关系。它只将模型元素本身连接起来而不需要用一组实例来表达它的意思。
- 它表示了这样一种情形，提供者的某些变化会要求或指示依赖关系中客户的变化。也就是说依赖关系将行为和实现与影响其他类的类联系起来。



依赖关系

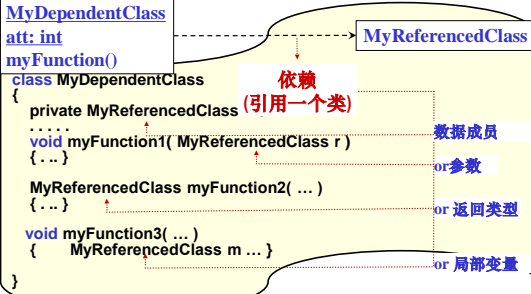
如果修改元素X会导致另一个元素Y的修改，则Y依赖于X

- 一个类是另一个类的数据成员类型
- 一个类的方法使用另一个类作为形式参数
- 消息的发送者与接收者之间的关系
- 一个类的方法创建了另一个类的实例
- ...



36

依赖关系



37

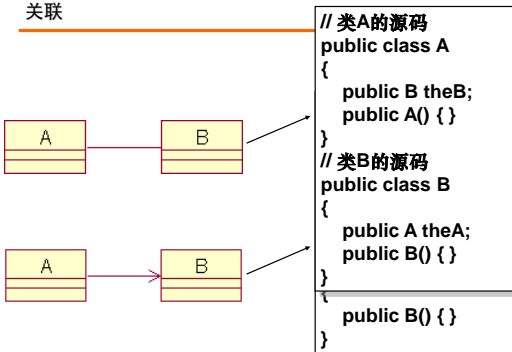
关联

关联(association)是模型元素间的一种语义联系,它是对具有共同的结构特性、行为特性、关系和语义的链(Link)的描述。

- 链是关联的实例
- 关联表示类与类之间的关系
- 链表示对象与对象之间的关系

38

关联



39

关联的导航特性

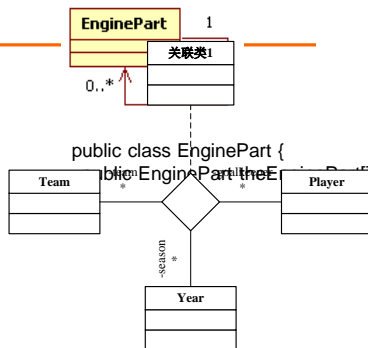
单向关联: 只在一个方向上存在导航表示。
双向关联: 在两个方向上都有导航表示。

双向关联的限制: 角色的互逆

40

关联的种类

- (1) 自反关联
- (2) 二元关联
- (3) N元关联



41

习题

- 在一个班级中,一个班长可以管理班级的同学,但是班长和同学都是学生。请根据这个描述画出UML图,标明重数,并且解释你是怎么设定重数的。

42

关联名

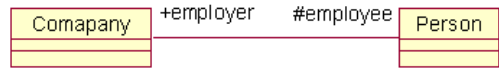
用来描述关联的作用, 在含义十分明显时, 名字可以省略。



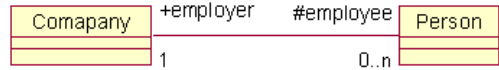
43

关联的角色

关联的两端可以以某种角色参与关联, 如果不标注角色则隐含用类名作为角色名。



角色的多重性表示可以有多少个对象参与该关联。



44

角色的多重性

多重性用非负整数的一个子集来表示
0..1, 0..*, 1..n, ...

如果多重性的上界大于1, 则称为**多值角色**, 通常把这种多值角色看成一个集合。

由于集合是无序的, 因此对于多值角色可使用一些**约束**。

45

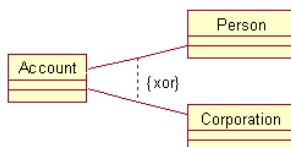
习题

- 设有1个班级类和学生类。1个班级是由若干学生构成, 1个学生只能属于一个班级。当学生还没有入学时, 可以只存在班级, 而没有学生。请根据描述画出这两个类UML图, 以及相互的联系。注意注明重数。

46

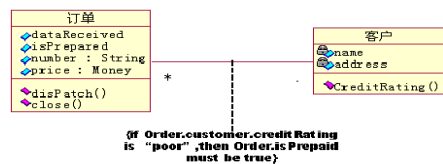
关联的约束

- (1) UML没有为约束定义严格的语法。
- (2) 约束应写在()括号中
- (3) 约束可用自然语言写, 也可用程序语言写。

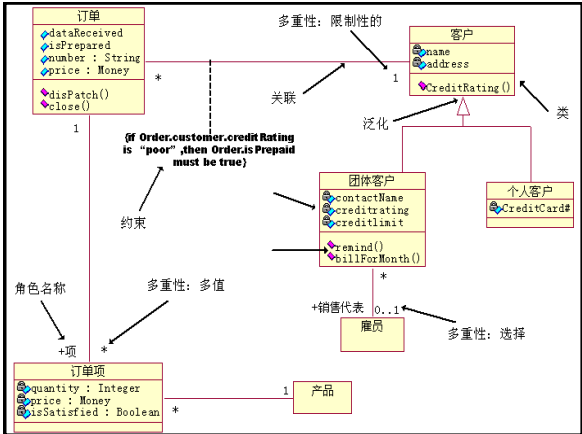


47

关联的约束



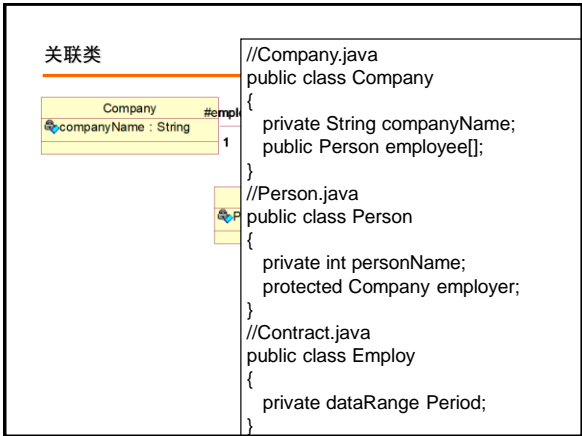
48



关联类

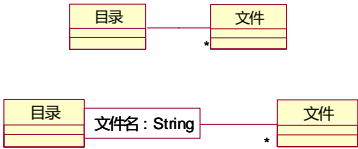
- 有时需要为关联设置一些属性、操作及其它特性。UML用建模元素“关联类”来表示。
- UML规定：在任何两个相关的对象之间，只能存在关联类的一个实例。

50

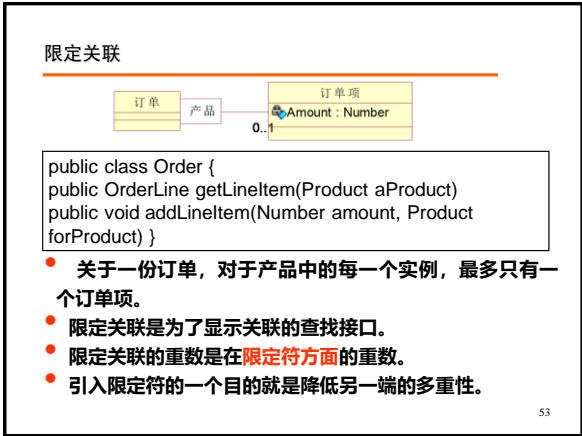


限定关联

- 限定关联是一对多或多对多关联的另一种表达形式。它通过在关联的一方添加限定符，明确地标识在关联的另一方出现的多个对象中的每一个对象。

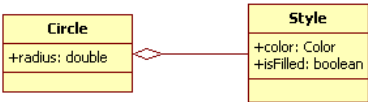


52



聚集与组合

聚集(aggregation)是一种特殊的关联,表示类之间的整体与部分关系。



53

54

聚集与组合

组合(composition)是一种特殊的聚集, 强调整体和部分具有相同的生命期。



55

泛化关系

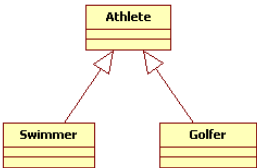
泛化关系指一般元素和特殊元素之间的分类关系。

- 在概念层, 泛化表示类与子类的关系。
- 在说明层, 泛化意味着子类的接口必须包括父类型接口中的每个元素。
- 在实现层, 泛化可以用类继承技术和授权技术来实现。

泛化涉及可替代性, 凡适用于父类的代码也同样适用于子类。

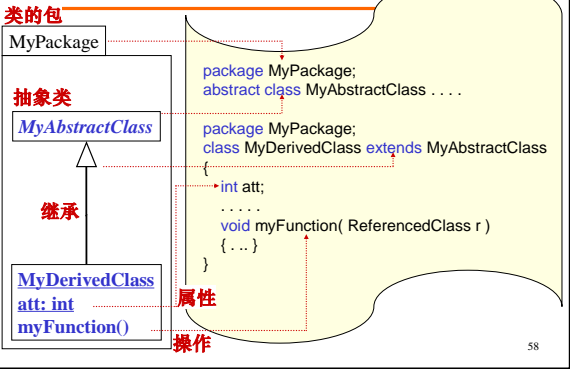
56

泛化关系



57

泛化关系



58

继承机制

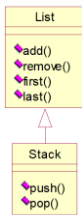
用继承机制实现泛化关系的条件

- ① 一个父类的任何子类都必须具有其父类的所有行为。这种行为的一致性要求类中的操作不仅在语法上一致, 而且在语义上也要一致。
- ② 语义上的一致意味着每个对象都应该是其父类的一个对象, 父类中的所有属性和操作都完全适用于子类对象。
- ③ 即便子类重载父类中的操作, 必须保证这个操作提供与父类操作同样的服务。只不过服务的内容可以更多更具体。

59

授权技术

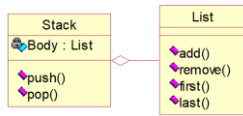
但是程序员在很多时候, 没有严格确保子类与父类在语义上的一致性。往往违背语义的一致性, 采用继承机制借用一个类的部分行为, 导致行为继承的错误。



60

授权技术

授权就是把原来属于类A的职责或任务转交给类B来完成。类B成为类A不可分割的一个组成部分。

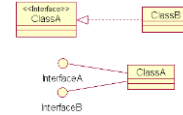


类List成为类Stack的私有成员

61

实现关系

- 实现关系将一种模型元素（如类）与另一种模型元素（如接口）连接起来，是说明和其实现之间的关系。
- 在实现关系中，接口只是行为的说明而不是结构或者实现，而类中则要包含了其具体的实现内容，可以通过一个或多个类实现一个接口，但是每个类必须分别实现接口中的操作。虽然实现关系意味着要有像接口这样的说明元素，它也可以用一个具体的实现元素来暗示它的说明（而不是它的实现）必须被支持。



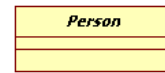
抽象类和接口

- 抽象类**
 - 一种不能够被直接实例化的类，也就是说不能够创建一个属于抽象类的对象。
 - 抽象类可以含有属性和某些方法的具体实现。
- 接口**
 - 一个接口是一个不带实现的类，它只规定类的外部特性。
 - 接口可通过抽象类来描述。抽象类可提供一些实现，但主要用于声明接口。

63

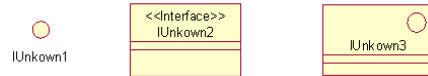
抽象类和接口

抽象类



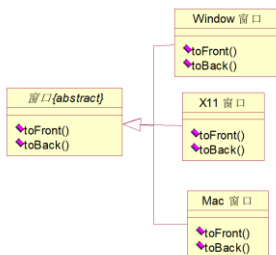
UML用斜体表示抽象元素（包括类、属性、方法等）的名称。

接口



64

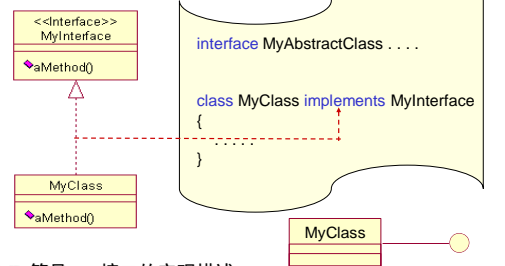
抽象类实现接口



定义独立于平台的抽象窗口类

65

接口的另一种表示方法



UML 符号 — 接口的实现描述
与继承不同，它必须实现接口中规定的所有操作

66

选择题

在类图中，下面哪个符号表示继承关系（ ）

- A.  B. 
C.  D. 

选C

67

选择题

在类图中，下面哪个符号表示聚合关系（ ）

- A.  B. 
C.  D. 

选D

68

选择题

在类图中，哪种关系表达总体与局部的关系（ ）

- A. 泛化
B. 实现
C. 依赖
D. 聚合

选D

69

选择题

UML中关联的多重度是指()

- A. 一个类有多个方法被另一个类调用
B. 一个类的实例能够与另一个类的多个实例相关联
C. 一个类的某个方法被另一个类调用的次数
D. 两个类所具有的相同的方法和属性

选B

70

选择题

当部分类的生命周期独立于整体类的生命周期时,两个的关系应为（ ）

- A. 聚合
B. 组合
C. 泛化
D. 继承

选A

71

填空题

在泛化关系中，子类可以替代_____。也就是说，后者出现的地方，前者都可以出现。但是反过来却不成立。

父类

72

填空题

- 领域模型是一组表示真实世界的_____, 不是软件部件的模型。
- 概念类

73

填空题

- 当一个类的对象可以充当多种角色时, _____关联就可能发生。
- 自身

74

填空题

- 客观世界中的若干类, 通常有两种主要的结构关系: 即 _____和整体与部分结构。
- 分类结构

75

名词解释

- 组成关联
用于表示类的对象之间更强形式的关联, 整体拥有各部分, 部分与整体共存, 如整体不存在了, 部分也会随之消失。(2分)
用一个实菱形物附在组成端表示。(1分)

76

习题

- 在设计一个企业工资管理系统时, 假设有一个抽象的基类员工类。这个抽象类中有一个公开的抽象方法getSalary, 以及私有的属性employeeType。在企业中有三种类型的员工: 全职员工、小时工、临时工。这三种员工的计算薪水的方式不同。请根据这个描述, 结合面向对象的知识, 画出系统的UML图。

习题

- 我校凌云楼有24层, 用电梯内的和每个楼层的按钮来控制3部电梯的运动。当按下电梯按钮以请求在某一指定楼层停下时, 按钮指示灯亮; 当请求获得满足时, 指示灯灭。当电梯无升降操作时, 关门, 并停在当前楼层。经分析得知, 按钮、电梯可作为两个候选类, 请画出按钮(包括电梯按钮、楼层按钮)和电梯的类结构图。

构造类图

寻找类的方法

- 根据用例描述中的名词确定候选类。
- 使用CRC分析法寻找类, CRC指: 类(class)、职责(responsibility)、协作(collaboration)。
- 根据边界类、控制类和实体类的划分来帮助发现类。
- 参考设计模式来确定类。
- 根据软件开发过程的指导寻找类, 如:RUP等。

构造类图时注意

- 不要试图使用所有的符号。
- 不要过早陷入细节, 根据阶段、层次逐步细化。
- 构造完成要将模型与目标问题对照验证其是否合理。

79

构造类图

建立类图的步骤

- 研究分析问题领域, 确定系统需求。
- 确定类, 明确类的含义和职责, 确定属性和操作。
- 确定类之间的关系。
- 调整和细化类及类之间的关系。
- 绘制类图并增加相应的说明。

80

从用例中识别类

- 用例图实质上是一种系统描述的形式, 自然可以根据用例描述来识别类。针对各个用例, 通常可以根据如下问题辅助识别:
 - (1) 用例描述中出现了那些实体?
 - (2) 用例的完成需要哪些实体合作?
 - (3) 用例执行过程中会产生并存储哪些信息?
 - (4) 用例要求与之关联的每个角色的输入是什么?
 - (5) 用例反馈与之关联的每个角色的输出是什么?
 - (6) 用例需要操作哪些硬设备?

领域分析

- 建立类图的过程就是对领域及其解决方案的分析与设计过程。
- 领域分析包括:
 - 通过对某一领域中的已有应用系统、理论、技术、开发历史等的研究, 来标识、收集、组织、分析和表示领域模型及软件体系结构的过程;
 - 根据这一过程得到的结果。

82

发现类

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, 因此需要一个人图书管理系统。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。

筛选备选类

- “小王”、“人”、“家里”很明显是系统外的概念, 无须对其建模;
- 而“个人图书管理系统”、“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。

筛选备选类

- “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，因此无需对其建模
- “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；

筛选备选类

- “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；

筛选备选类

- “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”；

筛选备选类

- “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，也无需将其建模；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。

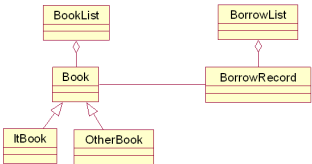
得到候选类

- 在使用“名词识别法”寻找类的时候，很多团队会在此耗费大量的时间，特别是对于中大型项目，这样很容易迷失方向。其实在此主要的目的是对问题领域建立概要的了解，无需太过咬文嚼字

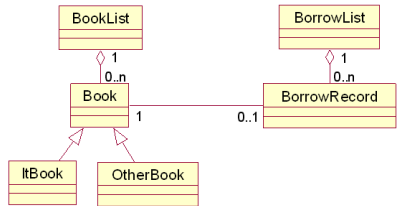
书籍 计算机类书籍 非计算机类书籍
借阅记录 借阅记录列表 书籍列表

类图建模技术

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

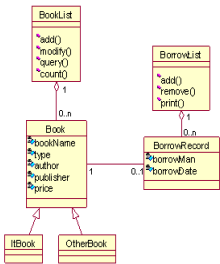


多重性分析



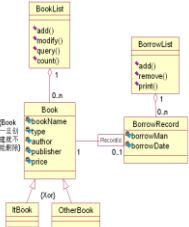
职责分析

- 书籍类：从需求描述中，可找到书名、类别、作者、出版社；同时从统计的需要中，可得知“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，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：借阅人（朋友）、借阅时间。
- 借阅记录列表类：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



限定与修改

- 导航性分析：Book与BookList之间、BorrowRecord和BorrowList之间是组合关系均无需添加方向描述，而Book与BorrowRecord之间则是双方关联，也无需添加
- 约束：Book对象创建后就不能够被删除只能被修改，因此在Book类边上加上用自由文本写的约束；一本书要么属于计算机类，要么属于非计算机类，因此在ItBook和OtherBook间加了“[Xor]”约束
- 限定符：一本书只有一册，因此只能够被借一次，因此对于一本Book而言只能有一个RecordId与其对应



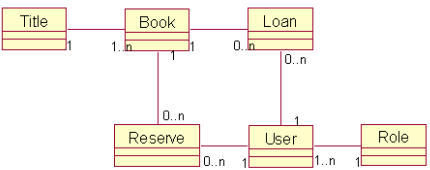
应用题

- 图书管理系统功能性需求说明如下：
图书管理系统能够为一定数量的借阅者提供服务。每个借阅者能够拥有唯一标识其存在的编号。图书馆向每一个借阅者发放图书证，其中包含每一个借阅者的编号和个人信息。提供的服务包括：提供查询图书信息、查询个人信息服务和预定图书服务等。
当借阅者需要借阅图书、归还书籍时需要通过图书管理员进行，即借阅者不直接与系统交互，而是通过图书管理员充当借阅者的代理和系统交互。

应用题

系统管理员主要负责系统的管理维护工作，包括对图书、数目、借阅者的添加、删除和修改。并且能够查询借阅者、图书和图书管理员的信息。
可以通过图书的名称或图书的ISBN/ISSN号对图书进行查找。
画出类图，并指明各个关系的重数。

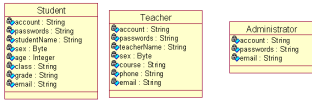
应用题



每个类1分，每个关系的重数1分（一共只需对10个）

习题

(1) 以“远程网络教学系统”为例，在该系统中，系统的参与者为学生、教师 and 系统管理员。学生包括登录名称、登录密码、学生编号、性别、年龄、班级、年级、邮箱等属性。教师包含自己的登录名称、登录密码、姓名、性别、教授课程、电话号码和邮箱等属性。系统管理员包含系统管理员用户名、系统管理员密码、邮箱等属性。根据这些信息，创建系统的类图。



(2) 在上题中，如果我们把参与者学生、教师 and 系统管理员进行抽象出一个单独的人员类，学生、教师 and 系统管理员分别是人员类的继承。根据这些信息，重新创建类图。

