中山大学软件工程学院

SCHOOL OF SOFTWARE ENGINEERING

第6讲 软件体系结构设计-Part II

接口设计+关系数据库映射

授课教师：张能　助理教授

zhangn279@mail.sysu.edu.cn

综合实验楼A323-3

2023年05月31日、06月02日

目录

- 实验内容
- 任务讲解
- SSE210 作业问题

实验内容

1. 接口设计

- 将对象之间的**消息映射**为类/任务的接口

2. 关系数据库映射

- 将实体类的**属性及**实体类之间的**关联映射**为关系数据库表

接口设计的消息映射
实体类的关系数据库映射

任务讲解

- 接口设计
- 关系数据库映射

接口设计

- 类的**接口**由类提供的**操作**组成
  - 一个操作包括操作名、输入参数、输出参数(即返回值)

- 类的操作可根据**静态模型**或**动态模型**确定

基于交互模型设计类操作

- **一个类的操作**可通过考虑该类的对象与其它对象的交互确定
  - 当两个对象交互时，一个对象为另一个对象提供一个操作

- **动态交互模型**描述了一个对象向另一对象发送消息的方向；发送消息的对象调用接收消息的对象的操作；**消息被映射为一个操作调用**

基于交互模型设计类操作

- 分析模型中，强调的是对象间传递的信息，而不是操作的准确语义
  - 交互图上的消息可能是个**名词（反映所传递的数据）**，也可能是个**动词（反映所要执行的动作）**

- **设计模型中，①需要定义类的操作**
  - 如果消息为名词，则需要定义接收消息的操作
  - 如果消息为动词，则该动词就表示操作的名称

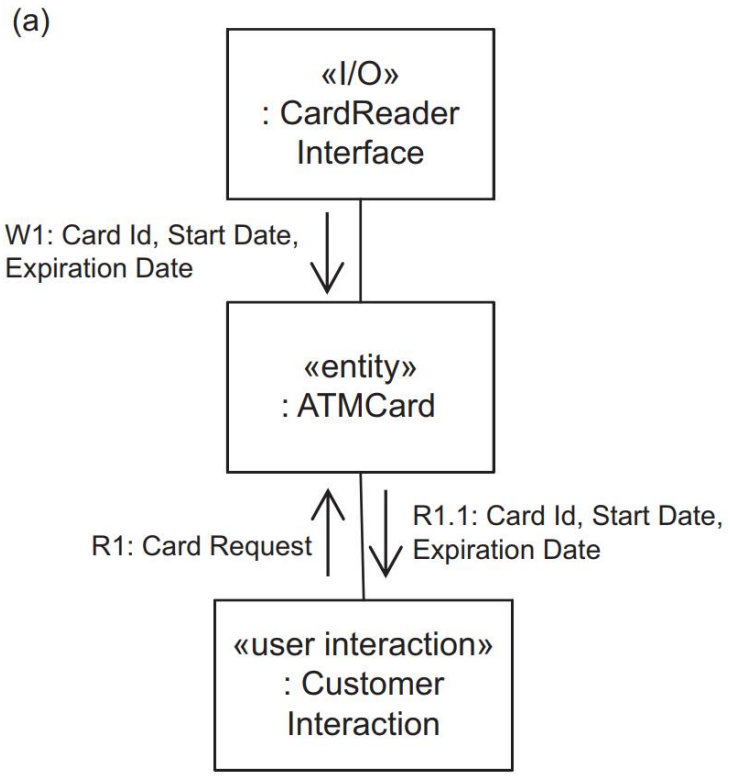
基于交互模型设计类操作

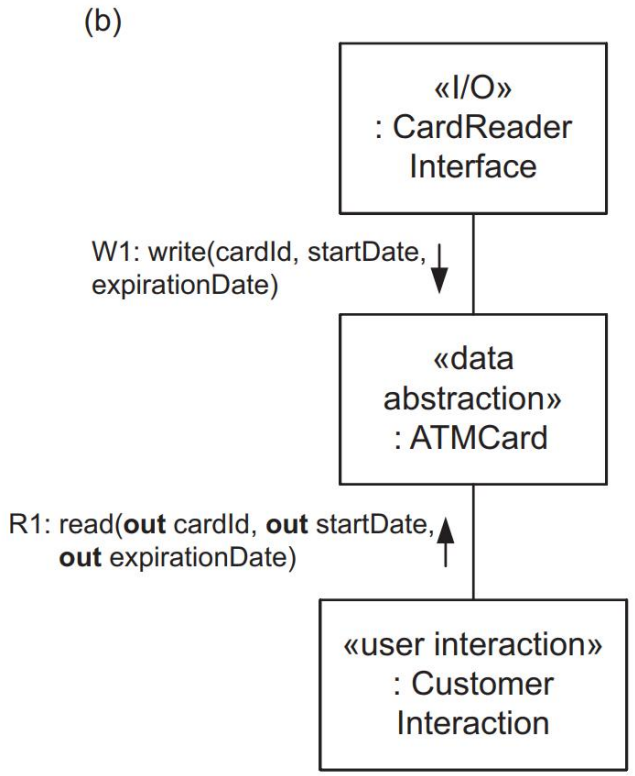
- **②还需考虑操作是否包含输入/输出参数**
- 在分析模型中，消息被描述为从发送对象向接收对象发送的**简单消息**
- 在设计模型中，**所有调用操作的消息都被描述为同步消息**；应答消息则被映射为操作的**返回参数**
  - **变量**、**对象**都可以作为操作的参数进行传递

NOTE: 发送给主动对象的消息通常直接由该对象的内部线程接收并处理，不会采用操作调用的形式

基于交互模型设计类操作

- 确定对象的操作后，将**操作**与**提供操作的类**在**类图**中进行描述





(c)

«data abstraction» ATMCard
- atmCardId: String - atmStartDate: Date - atmExpirationDate: Date
+ write (**in** cardId, **in** startDate, **in** expirationDate) + read (**out** cardId, **out** startDate, **out** expirationDate)

第1种操作表示法

操作名(**in** 参数名, **out** 参数名); 当未标记in和out时，所有参数默认为in。

Q: 如何表示参数类型？

Figure 14.1. Example of data abstraction class: (*a*) Analysis model: communication diagram. (*b*) Design model: communication diagram. (*c*) Design model: class diagram

基于静态模型设计类操作

- 通过静态模型的类图确定类的操作是可能的, 尤其是实体类
- **属性的标准操作**: create(创建)、read(读取)、update(更新)、delete(删除)

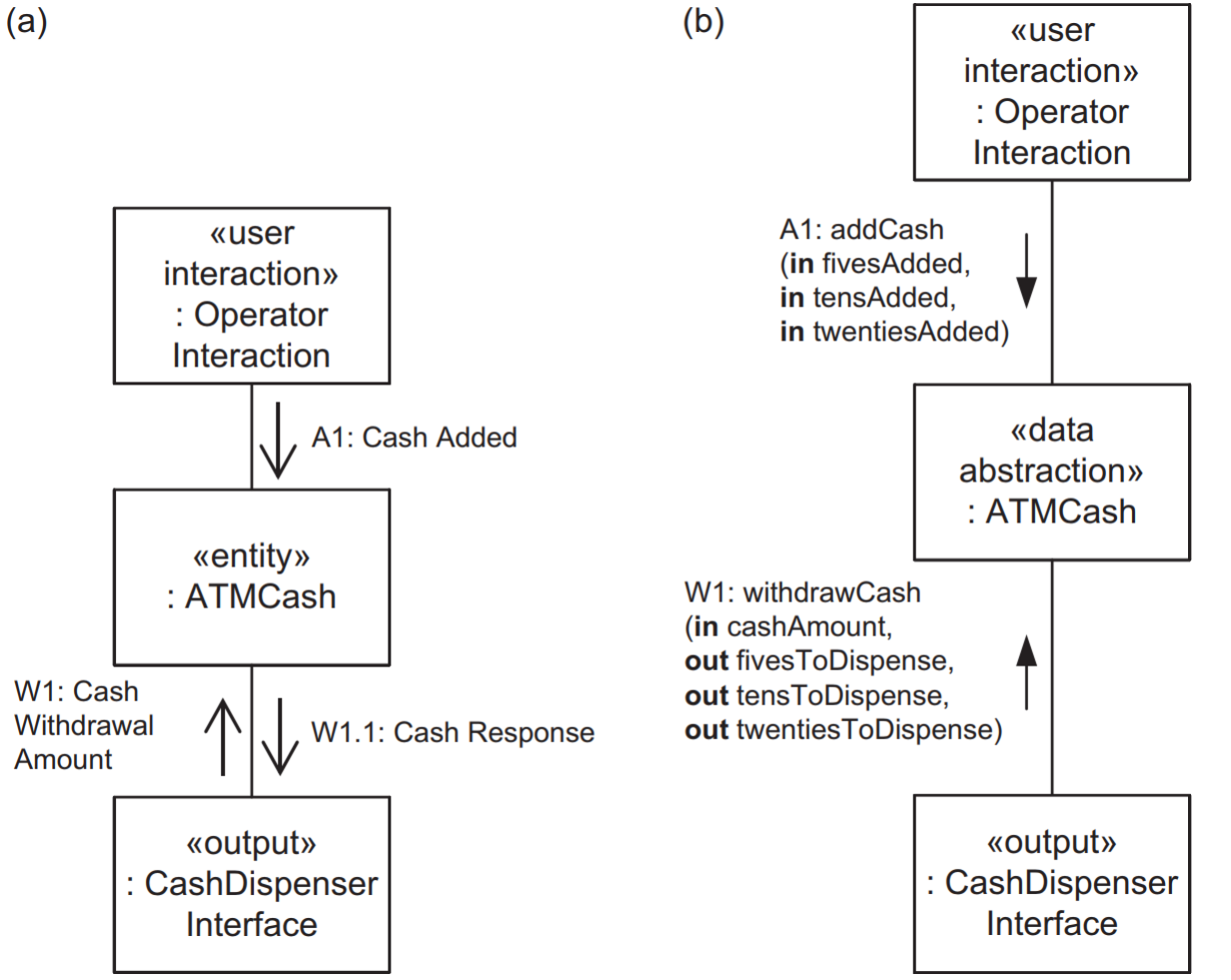
- 这些标准的操作可按照特定需要进行**裁剪!**

数据抽象类(Data Abstraction Class)

- **数据抽象类**: 封装了数据的实体类
  - 一个实体类保存了一些数据, 并提供了读写数据的操作

- 数据抽象类用于封装数据结构, 隐藏数据结构的内部实现细节
  - **数据抽象类所封装的属性**可从问题域的静态模型获取
  - **数据抽象类的操作**可考虑使用数据抽象对象间接访问数据的对象的需要确定

数据抽象类: 示例



(c)

«data abstraction» ATMCash
- cashAvailable: Integer = 0 - fives: Integer = 0 - tens: Integer = 0 - twenties: Integer = 0
+ addCash (**in** fivesAdded, **in** tensAdded, **in** twentiesAdded) + withdrawCash (**in** cashAmount, **out** fivesToDispense, **out** tensToDispense, **out** twentiesToDispense)

Figure 14.2. Example of data abstraction class: (*a*) Analysis model: communication diagram. (*b*) Design model: communication diagram. (*c*) Design model: class diagram.

状态机类(State-Machine Class)

- **状态机类**: 封装了一个状态图的信息
  - 在设计阶段, 需要对分析模型中确定的状态机类进行设计

- 状态机对象所执行的状态图被封装在一个**状态转换表**中
- **状态机类隐藏了状态转换表的内容, 并维护对象的当前状态**
- 状态机类提供了访问状态转换表和改变当前状态的操作

状态机类

- **状态机类的操作设计**
- **设计一个可复用的状态机类**
  - 隐藏状态转换表的内容，提供两个非特定应用的可复用操作：

    processEvent、**currentState** → processEvent 用于处理事件，将事件作为一个输入参数；currentState 用于返回当前状态，是可选的

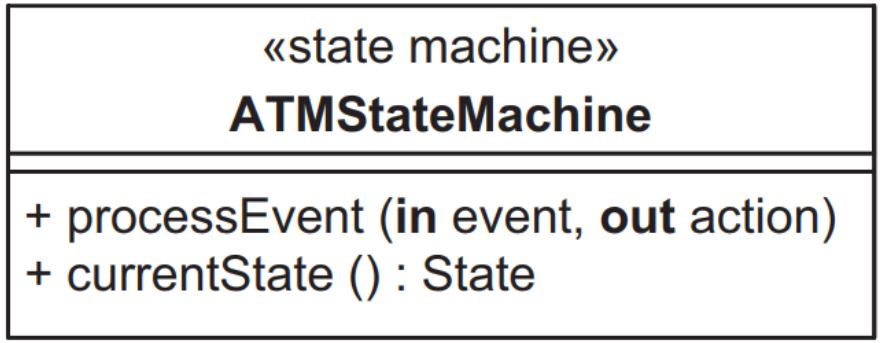


Figure 14.3. Example of state-machine control class

图形用户交互类(GUI Class)

- **图形用户交互类**: 隐藏了与用户界面相关的细节
  - 用户界面可能是一个简单的命令行界面或一个复杂的GUI界面

- 命令行界面通常由一个**用户交互类**处理
- 图形用户界面通常需要多个GUI类
  - **底层的GUI类**: 窗口、菜单、按钮、对话框等
  - **高层的聚合用户交互类**: 包含较低层次的GUI类的聚合类

图形用户交互类: 示例

➢ **例**：银行系统中与客户交互的GUI类

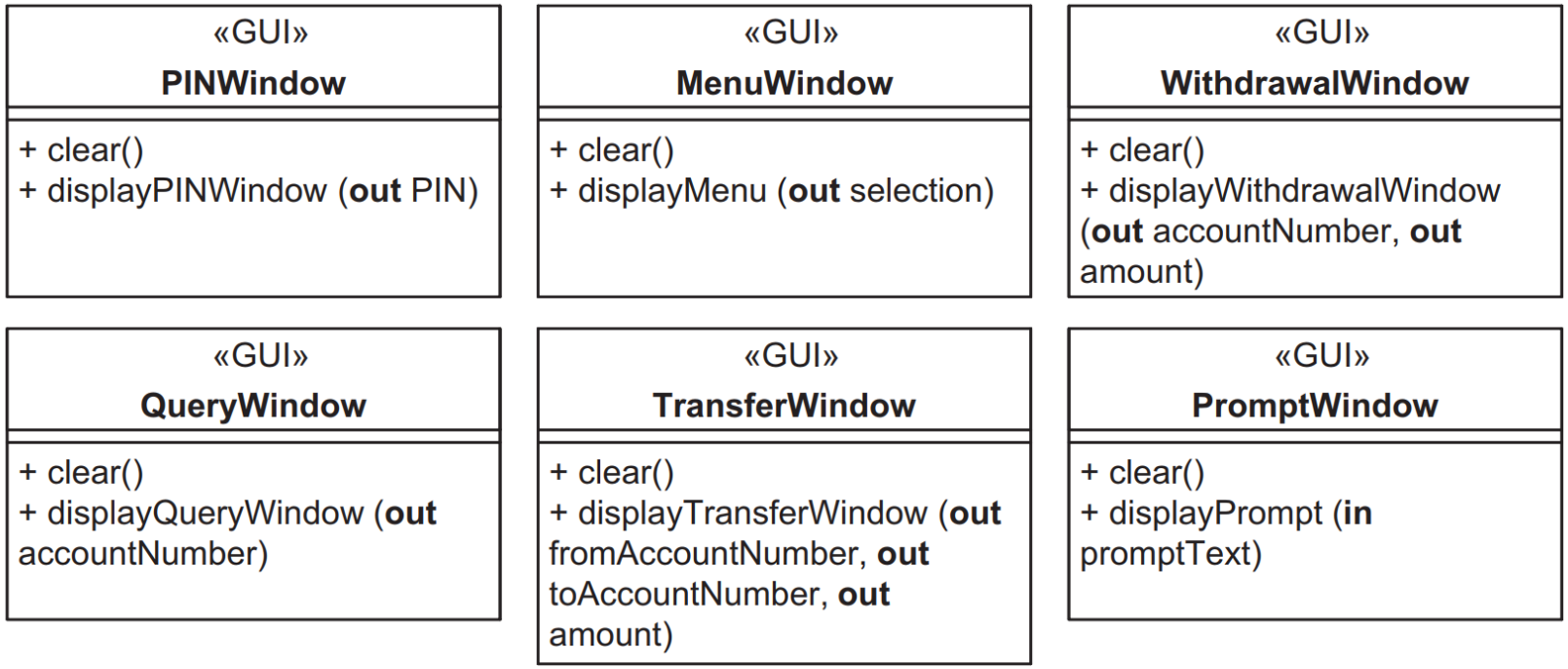


Figure 14.4. Example of graphical user interaction (GUI) classes

每个GUI类拥有:
1. **一个窗口显示操作**, 通过窗口与客户交互
2. **一个清空操作**, 可清空窗口屏幕
3. **至少一个与所提供的输出功能相关的操作**
4. 对于每个显示窗口, 显示操作都会向用户输出提示信息, 也可以接收用户输入作为该显示操作的输出参数返回

业务逻辑类(Business Logic Class)

- **业务逻辑类**: 封装了处理特定业务的应用逻辑
  - **目的**:将可能会独立变化的业务规则封装到不同的业务逻辑类中

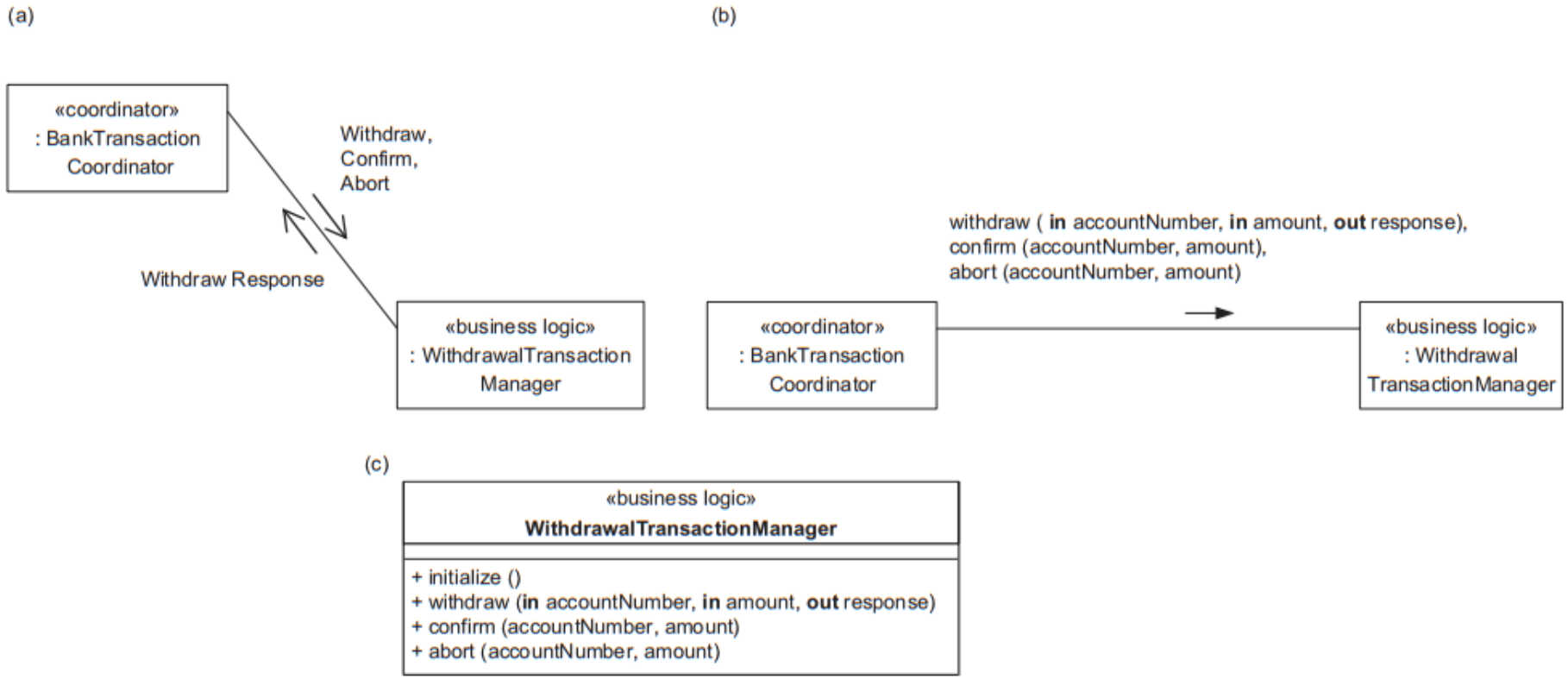


Figure 14.5. Example of business logic class: (*a*) Analysis model: communication diagram. (*b*) Design model: communication diagram. (*c*) Design model: class diagram

继承(Inheritance)

- **继承**: 用于设计相似却不相同的类
- **设计类时应考虑继承**, 以使**代码共享**和**代码的适应性**在详细设计和编码中得到充分利用
- **父类的内部对子类是可见的**

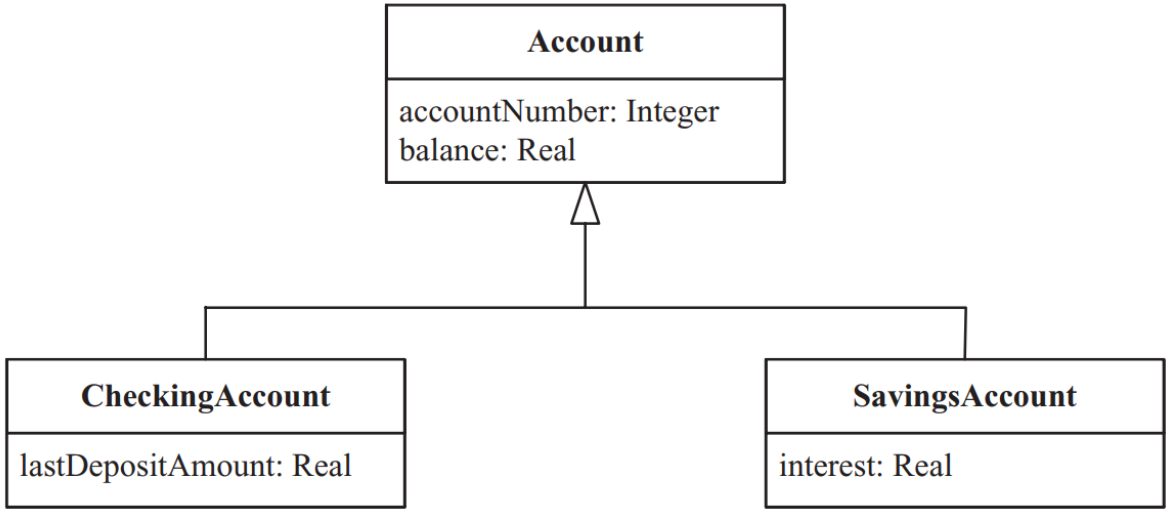


Figure 14.6. Example of abstract superclass and subclasses: analysis model

抽象类(Abstract Class)

- **抽象类**: 没有实例的类
- **抽象类可用作创建子类的模板**, 而不是创建对象的模板
- **一个抽象类必须至少包含一个抽象操作**: 只有声明但没有实现的操作

- **一些操作可在抽象类中实现**: 当部分或全部子类都需要使用相同实现时
  - 抽象类可定义一个操作的**缺省实现**, 子类可对该操作进行**重定义** (override)

抽象类: 示例

- **例:** 银行系统中不同类型的账户
- **设计起点**: 静态建模中开发的泛化/特化类图

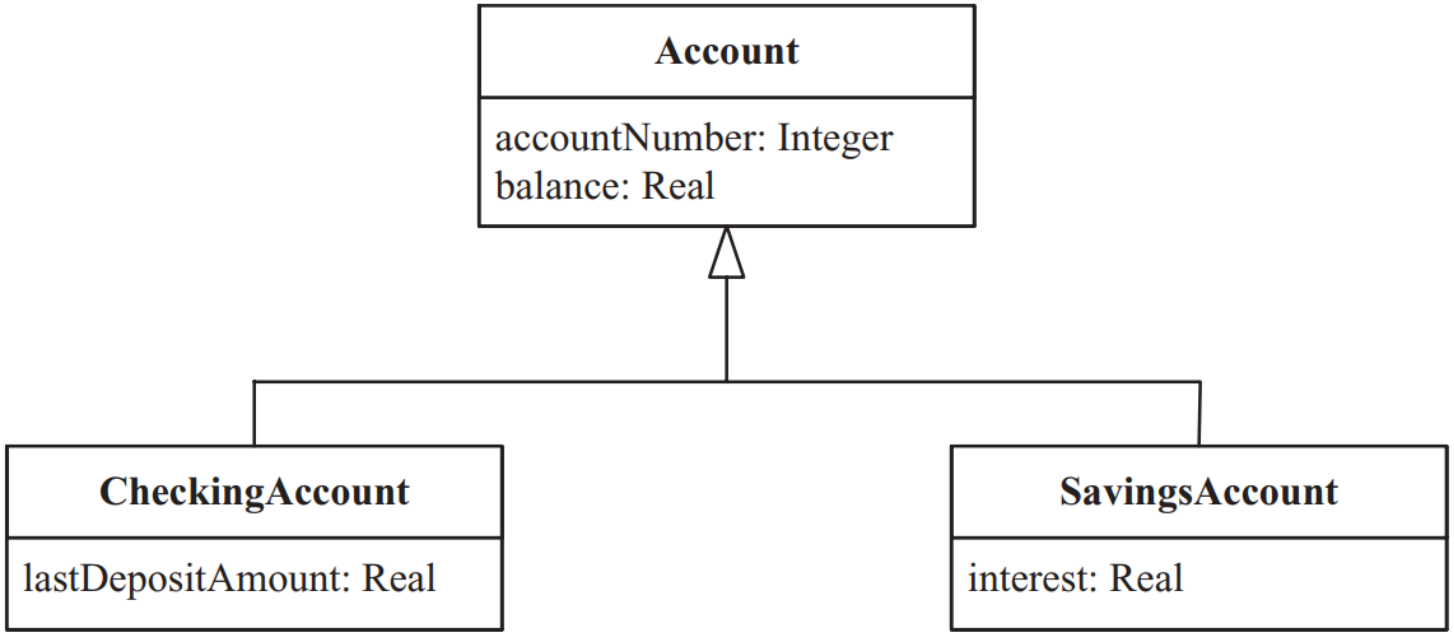


Figure 14.6. Example of abstract superclass and subclasses: analysis model

抽象类: 示例

- **例:** 银行系统中不同类型的账户
- UML采用**斜体**表示抽象类

第2种操作表示法

操作名(参数名:类型):返回值类型

Q:如何表示多个返回值?

综合前述第1种和第2种操作表示法,更全面的操作表示法: 操作名(**in** 参数名:类型, **out** 参数名:类型)

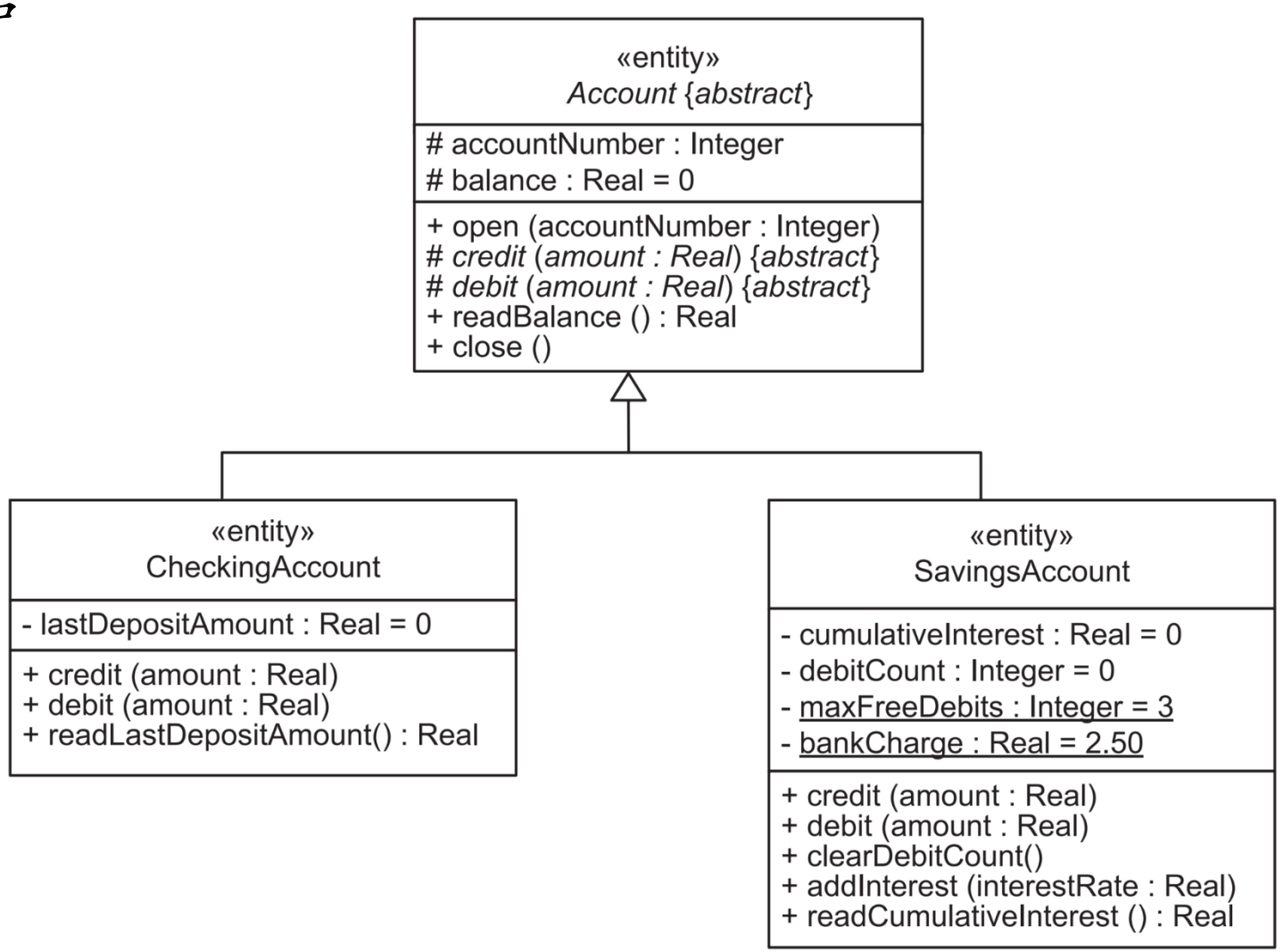


Figure 14.7. Example of an abstract superclass and subclasses: design model

任务讲解

- 接口设计
- 关系数据库映射

从静态模型到关系数据库设计

- **关系数据库**(relational database): 由一系列名称唯一的**关系表**组成
- **最简单的情况**下，静态模型的每个实体类都被设计为一张关系表
  - 实体类的类名 → 关系表名
  - 实体类的每个属性 → 关系表的一列
  - 实体类的每个对象 → 关系表的一行

Table 15.1. Account relational table

accountNumber	balance
1234	398.07
5678	439.72
1287	851.65

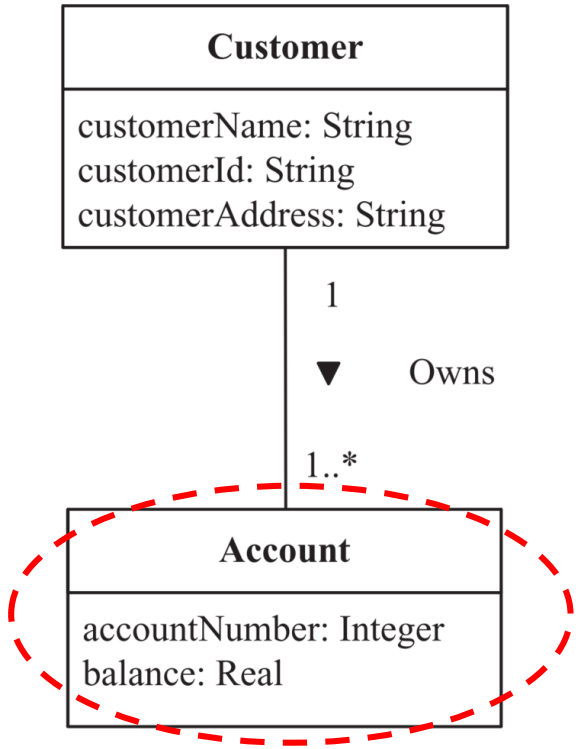


Figure 15.15. Identifying primary and secondary keys (one-to-many association)

从静态模型到关系数据库设计

- 每张关系表必须有一个**主键**(primary key)
- **最简单的情况**下，主键是一个能唯一确定表中某一行的属性

- **关系表可表示为**: **表名(主键属性, 属性, ...)**

Account (accountNumber, balance)

主键用下划线标识

从静态模型到关系数据库设计

■ 有些关系表需要用多个属性表示主键

➢ **例**：若账户表既包含支票账户又包含储蓄账户，且账户号可能重复，则需要两个属性作为主键

Account (accountNumber, accountType, balance)

从静态模型到关系数据库设计

- 关系数据库中的**关联**有多种表示法
- **最简单的方法**: 用**外键**(foreign key)表示**一对一**关联和**一对多**关联
  - **外键**: 在一张关系表中的另一张关系表的主键

Table 15.2. Navigation between relational tables

Navigation from *customerId* (foreign key) in Account table . . .

accountNumber	Balance	*customerId*
1234	398.07	**24193**
5678	439.72	**26537**
1287	851.65	**21849**

. . . to customerId (primary key) in Customer Table

customerName	customerId	customerAddress
Smith	**21849**	New York
Patel	**26537**	Chicago
Chang	**24193**	Washington

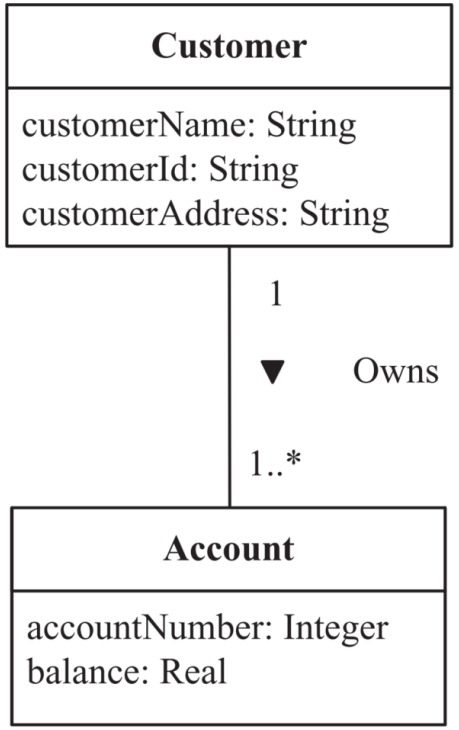


Figure 15.15. Identifying primary and secondary keys (one-to-many association)

从静态模型到关系数据库设计

- 类间的**一对一关联**: 任意一张关系表的主键可为另一张关系表的外键
- 类间的**零或一关联**: **外键必须在可选的关系表中**, 避免出现**空引用**

Customer (customerName, customerId, customerAddress)

Debit Card (cardId, PIN, expirationDate, status, *customerId*)

(underline = primary key, italic = *foreign key*)

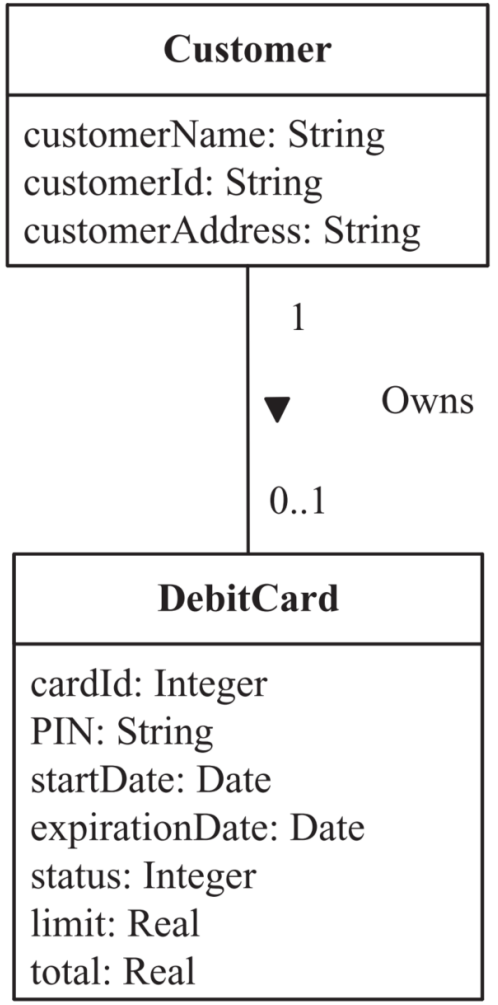


Figure 15.16. Identifying primary and secondary keys (zero-or-one association)

从静态模型到关系数据库设计

- 类间的**一对多关联**: **外键放在多的一方的关系表中**

Customer (customerName, customerId, customerAddress)

Account (accountNumber, balance, *customerId*)

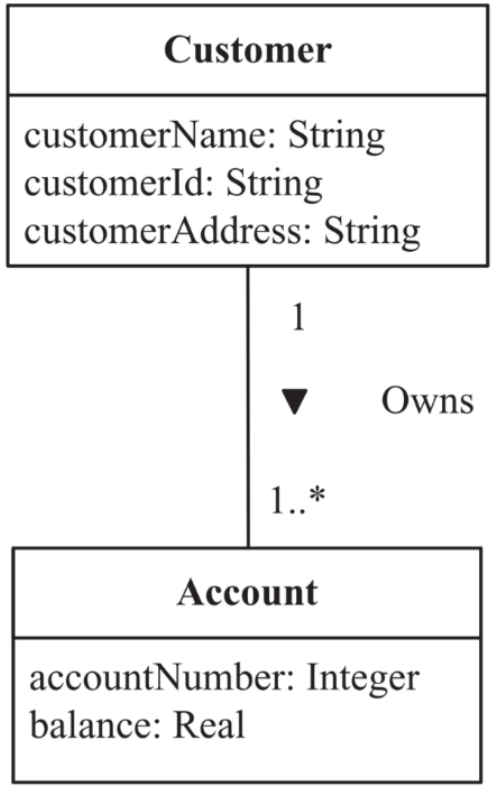


Figure 15.15. Identifying primary and secondary keys (one-to-many association)

从静态模型到关系数据库设计

- **关联类**表示两个或多个类之间的关联关系，常用于表示**多对多关联**
- 关联类需要被映射为**关联表**(association table)，**关联表的主键是一个复合键**，由参与关联的关系表的主键组成

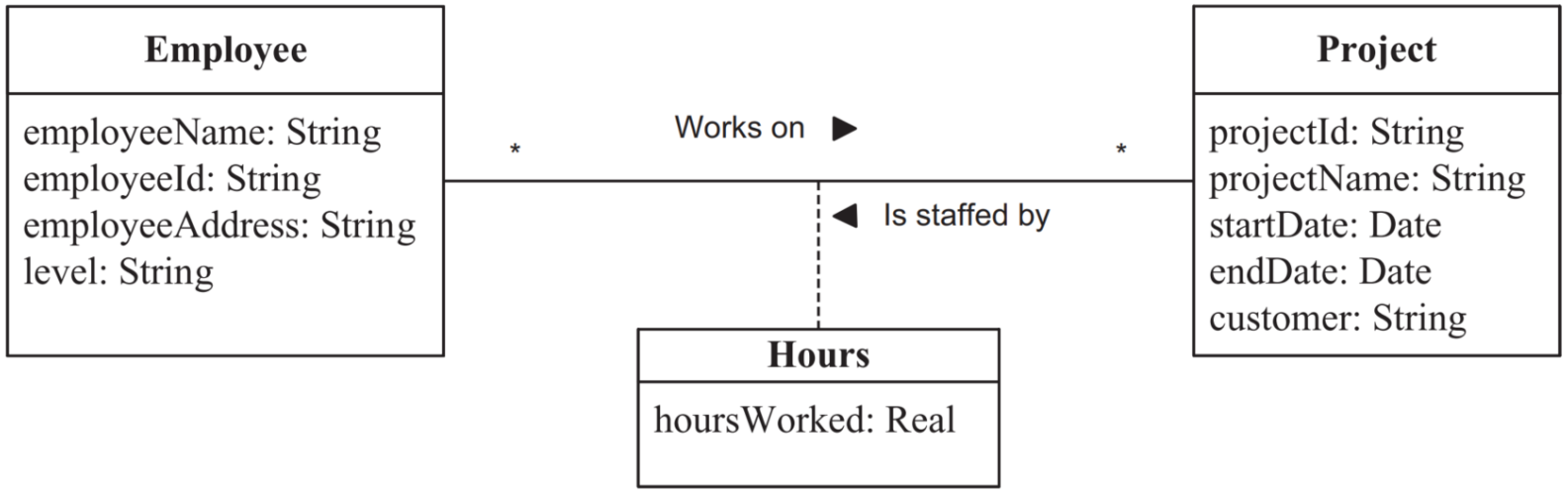


Figure 15.17. Mapping association class to association table

Project (projectId, projectName)
Employee (employeeId, employeeName, employeeAddress)
Hours (*projectId*, *employeeId*, hoursWorked)

从静态模型到关系数据库设计

- **整体/部分关系**是**组合**或**聚合**关系，包含代表组合/聚合类的一个实体类与代表部分类的两个或多个实体类

- **整体类和每个部分类都要被设计为一个关系表**
- 整体关系表的主键作为部分关系表的以下某项:
  - 1) **部分表的主键**: 当整体类与部分类之间存在**一对一关联**
  - 2) **部分表的复合主键的一部分**: 当整体类和部分类之间存在**一对多关联**
  - 3) **部分表的外键**: 当部分表不需要用复合主键来唯一确定表中某一行，且整体类与部分类之间存在**一对多关联**

不准确，如何修改？

从静态模型到关系数据库设计

- **泛化/特化关系**映射到关系数据库有3种可选方法
  - 1) **每个父类和子类分别映射为一张关系表**
  - 2) **只将子类映射为关系表**
  - 3) **只将父类映射为关系表**

从静态模型到关系数据库设计

■ 每个父类和子类分别映射为一张关系表

- 父类和每个子类的关系表**共享主键**
- 要在父类中显式地定义子类的**区分属性**, 以确定导航到哪个子类表
- **优点**: 简洁、可扩展
- **缺点**: 父类/子类之间的导航较慢

Account (accountNumber, accountType, balance)
Checking Account (accountNumber, lastDepositAmount)
Savings Account (accountNumber, interest)

此方案假设:
accountNumber是唯一的

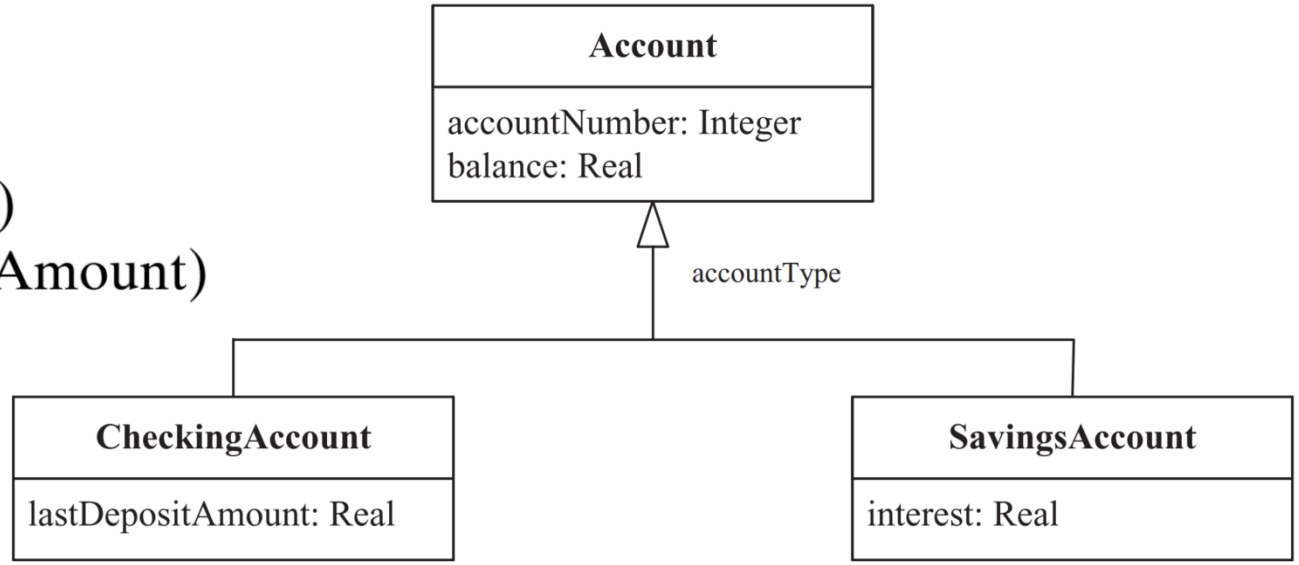


Figure 15.19. Mapping generalization/specialization relationship to relational tables

从静态模型到关系数据库设计

只将子类映射为关系表

- 每个子类会映射为一张关系表, 但没有父类对应的关系表
- 父类的属性在每个子类表中重复出现
- **最佳使用情形**: 子类属性比较多, 父类属性比较少
- **应用程序需要知道查询哪个子类表**

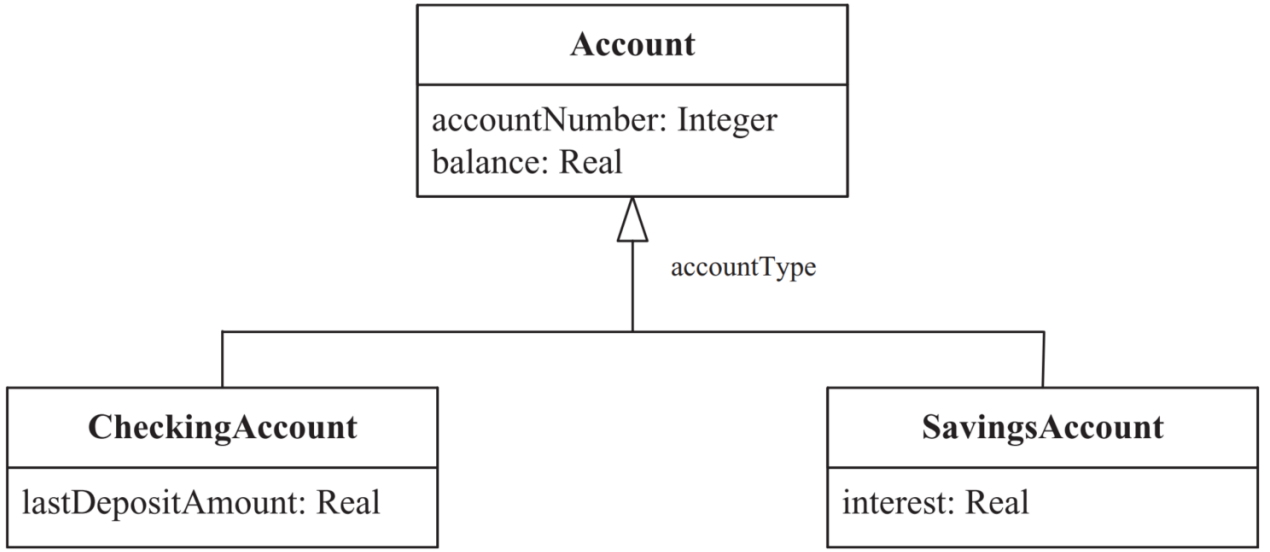


Figure 15.19. Mapping generalization/specialization relationship to relational tables

此方法常用于提高数据库的访问速度 → 避免了父类与子类之间的导航

Checking Account (accountNumber, balance, lastDepositAmount)
Savings Account (accountNumber, balance, interest)

从静态模型到关系数据库设计

■ 只将父类映射为关系表

- 只有一张父类的关系表, 不存在子类表
- 所有子类的属性都放到父类表中
- 父类表中需要添加子类的**区分属性**
- 父类表的每一行描述某个子类的属性, 与该子类无关的属性设为空值
- **最佳使用情形**: 父类属性非常多, 子类数量及其属性很少

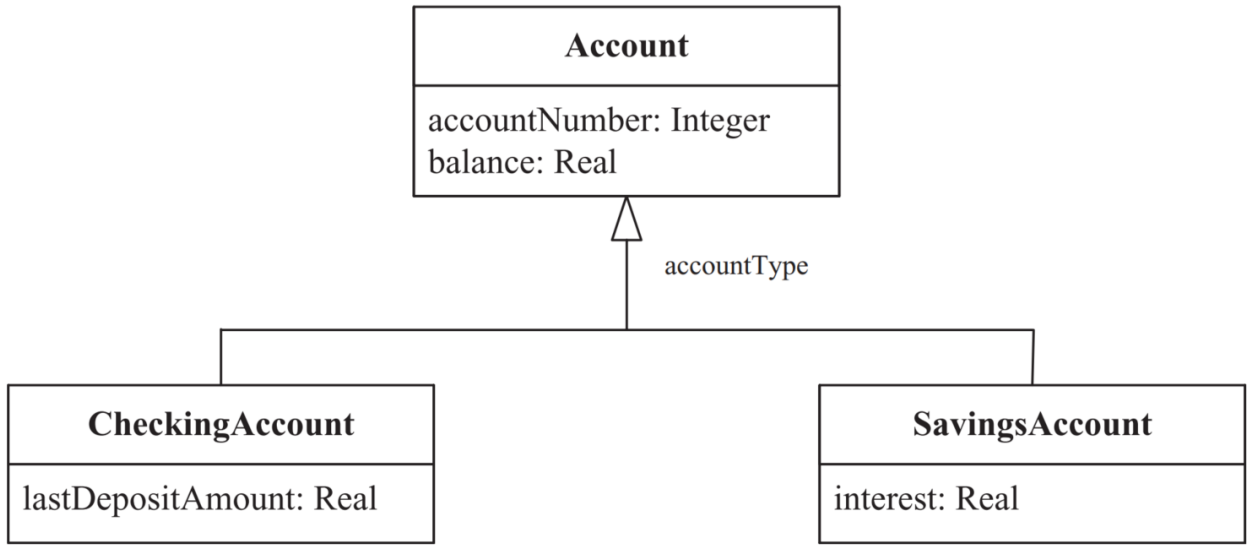


Figure 15.19. Mapping generalization/specialization relationship to relational tables

Account (accountNumber, accountType, balance, lastDepositAmount, interest)

SSE210 作业问题

SSE210 作业问题讲解

SSE212课程实践: 接口设计+关系数据库映射

SSE212课程实践系统--
图书自助借还系统

SSE212课程实践: 接口设计+关系数据库映射

■ 实验产出

- **平时作业4**: 各小组内每个成员提交一个类的接口设计: 其它对象与该类的对象之间的消息交互(即并发通信图), 以及该类的完整类图(属性+操作) //参考P12、P17
- **平时作业5**: 各小组共同提交一份全部实体类的关系数据库映射表: 表名、属性、主键、外键等要表示清楚