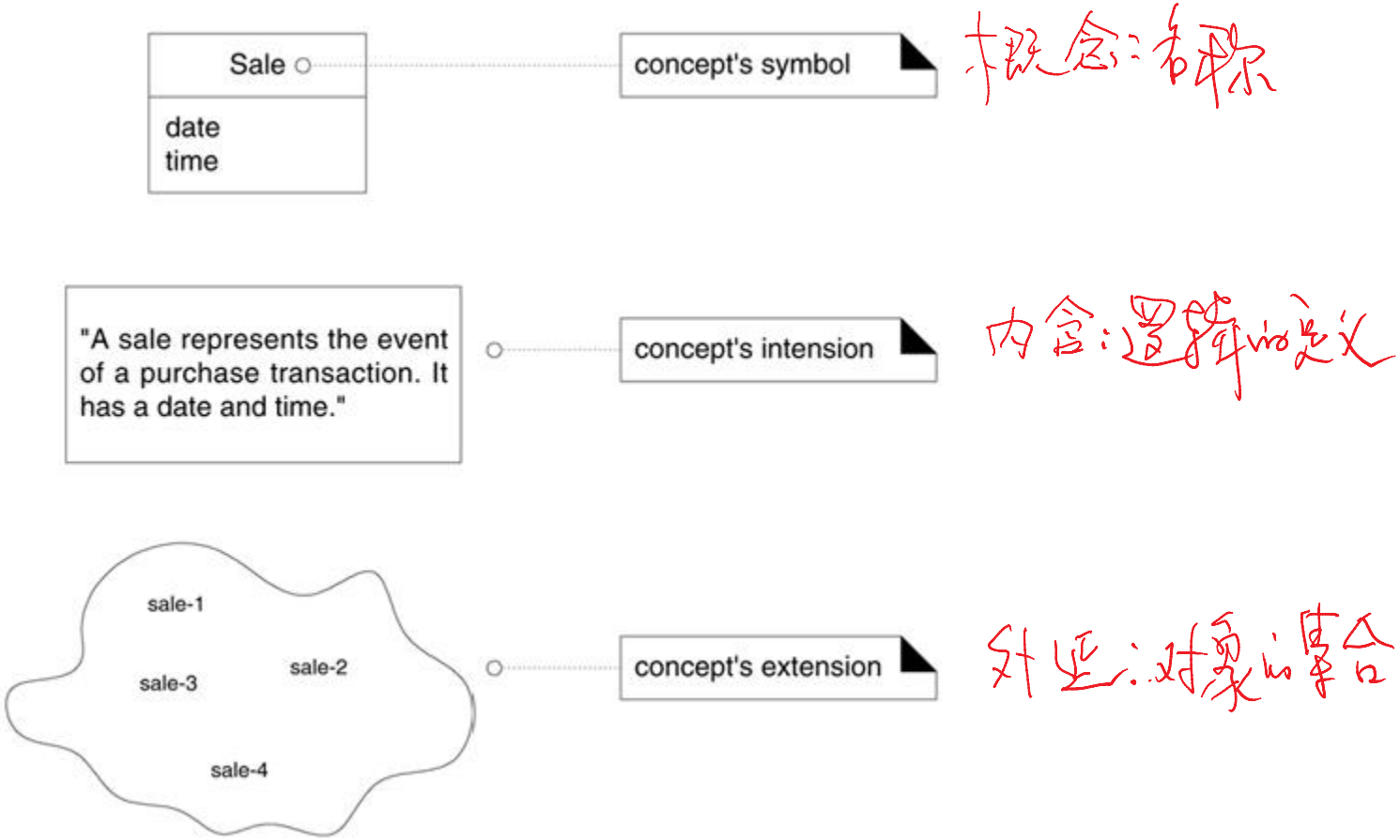
System Analysis and Design

L10. Domain Models II

Topics

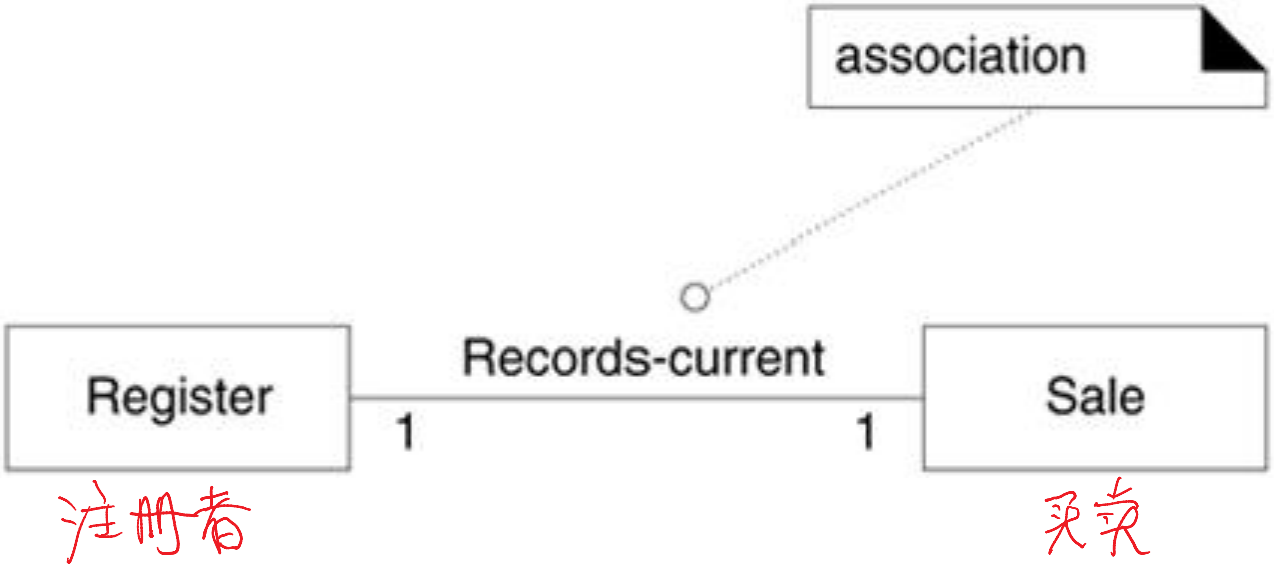
- Domain Model
- How to Create a Domain Model
- Associations
- Attributes
- Iterative and Evolutionary Domain Modeling

A Conceptual Class Has a Symbol, Intension, and Extension



Associations

- Relationship between (instances of) classes that indicates some interesting and meaningful connection 类之间的联结关系.



Associations

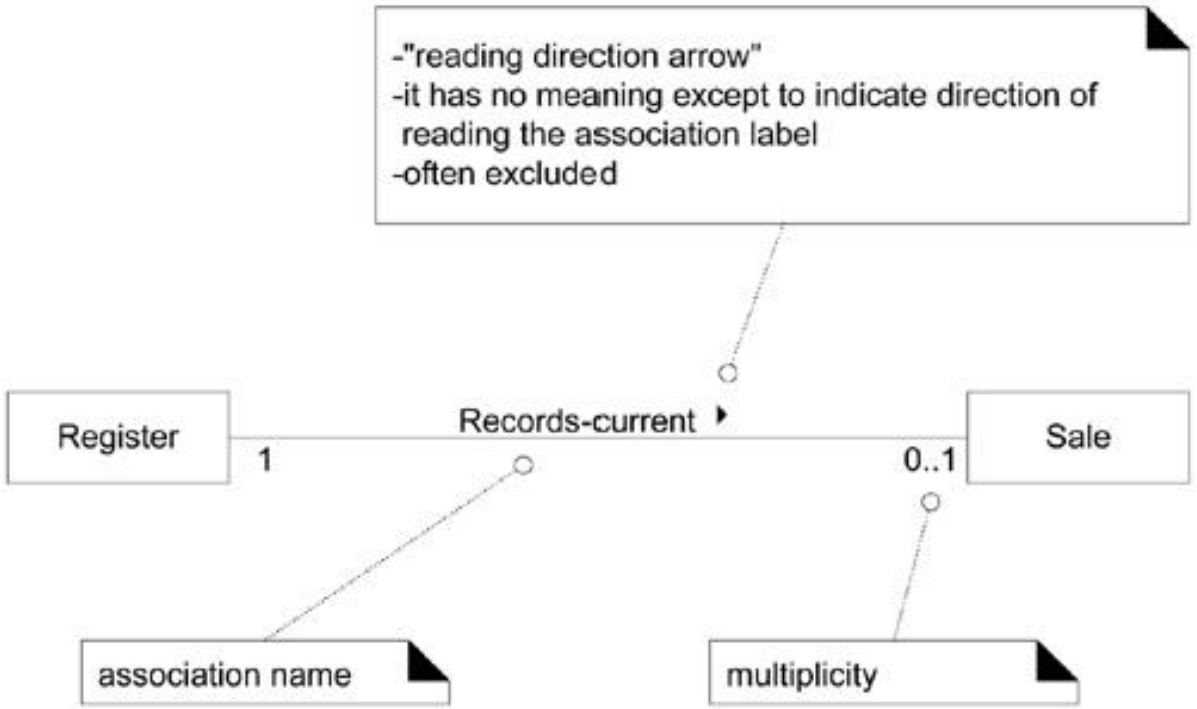
- Include "need-to-remember" associations in domain model
  - Associations imply knowledge of a relationship that needs to be preserved for some duration.
  - In Monopoly, where each piece is and which player owns which token
  - No need to remember numbers on the dice
- Avoid having too many; this adds communication paths and complexity

Will the Associations Be Implemented in Software?

- During domain modeling, an association relationship is meaningful in a purely conceptual perspective in the real domain.
- Many of these relationships *will* be implemented in software as paths of navigation and visibility (both in the Design Model and Data Model).
- But an association is *not* a statement about data flows, database foreign key relationships, instance variables, or object connections in a software solution;
- Associations are added to highlight our rough understanding of noteworthy relationships, not to document object or data structures.

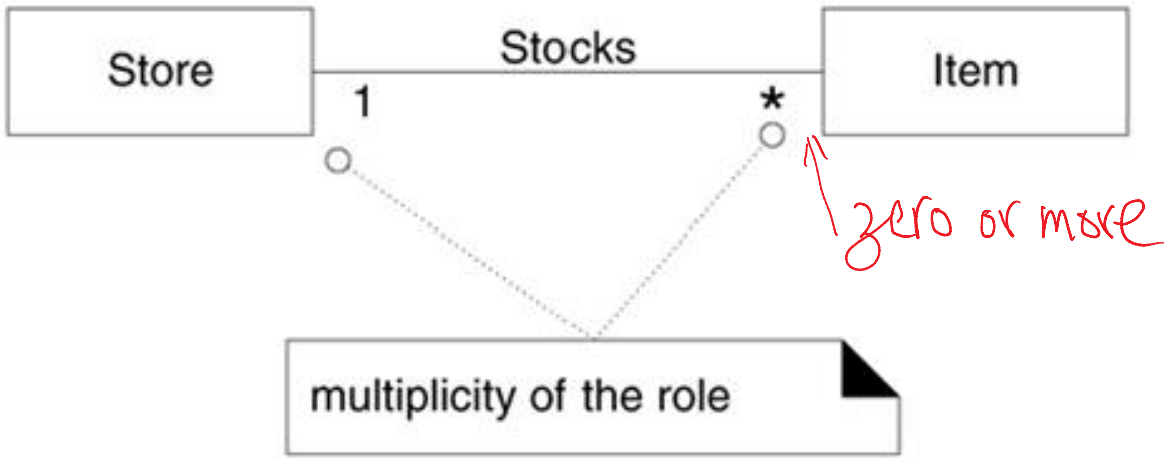
The UML Notation for Associations

- An association is represented as a line between classes.
- Name with *ClassName-VerbPhrase-ClassName* format.
- Each end of an association is called a role. Roles may optionally have multiplicity expression, name, navigability.
- The association is inherently bidirectional (logical traversal)
- An optional "reading direction arrow" indicates the direction to read the association name

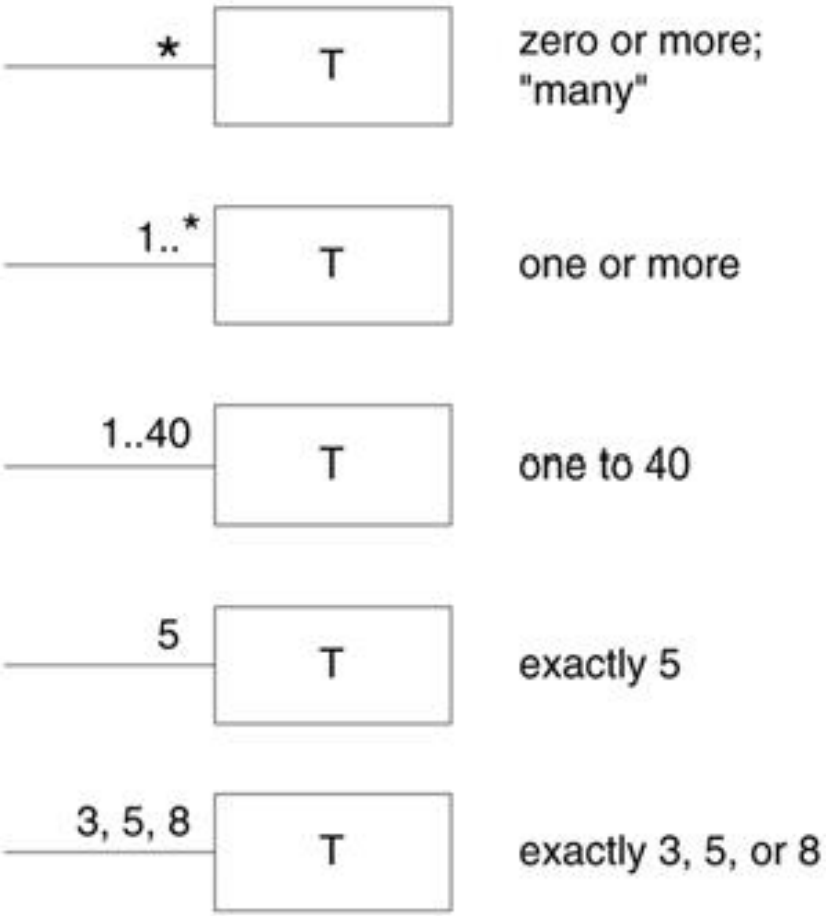


Multiplicity on an Association

- Multiplicity: how many instances of class A can be associated with an instance of class B.
  - E. g. an instance of *Store* can be associated with many (zero or more) *Items*
- This is at a particular moment, not over time



Multiplicity Values



- The multiplicity value communicates how many instances can be validly associated with another, at a particular moment, rather than over a span of time.
- A person can be *Married-to* only one other person at any particular moment, even though over a span of time, that same person may be married to many persons.

Multiplicity is Context Dependent

Multiplicity should "1" or "0..1"?

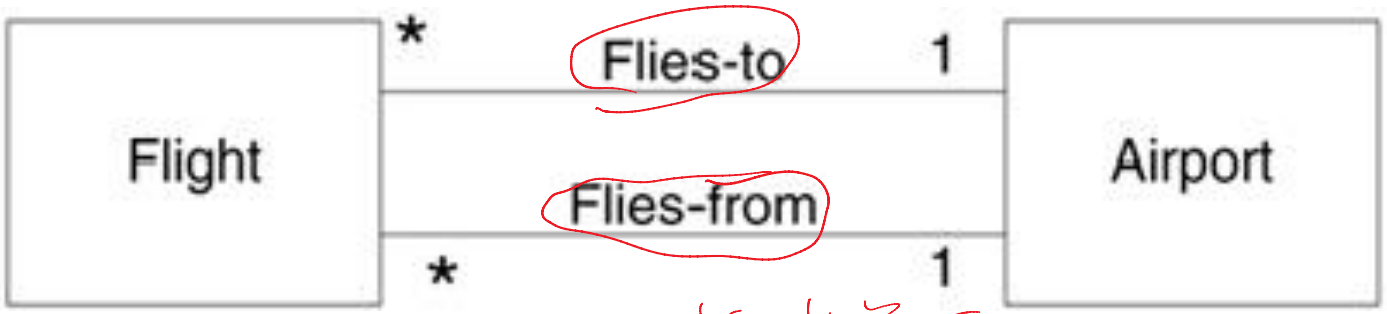
The answer depends on our interest in using the model. Typically and practically, the muliplicity communicates a domain constraint that we care about being able to check in software, if this relationship was implemented or reflected in software objects or a database. For example, a particular item may become sold or discarded, and thus no longer stocked in the store. From this viewpoint, "0..1" is logical, but ...

Do we care about that viewpoint? If this relationship was implemented in software, we would probably want to ensure that an *Item* software instance would always be related to 1 particular *Store* instance, otherwise it indicates a fault or corruption in the software elements or data.

This partial domain model does not represent software objects, but the multiplicities record constraints whose practical value is usually related to our interest in building software or databases (that reflect our real-world domain) with validity checks. From this viewpoint, "1" may be the desired value.

Multiple Associations

- The flying-to and flying-from associations are distinctly different relationships, which should be shown separately



通常可能的关联关系(总结到表)

Common Associations List

- Find Associations with a Common Associations List

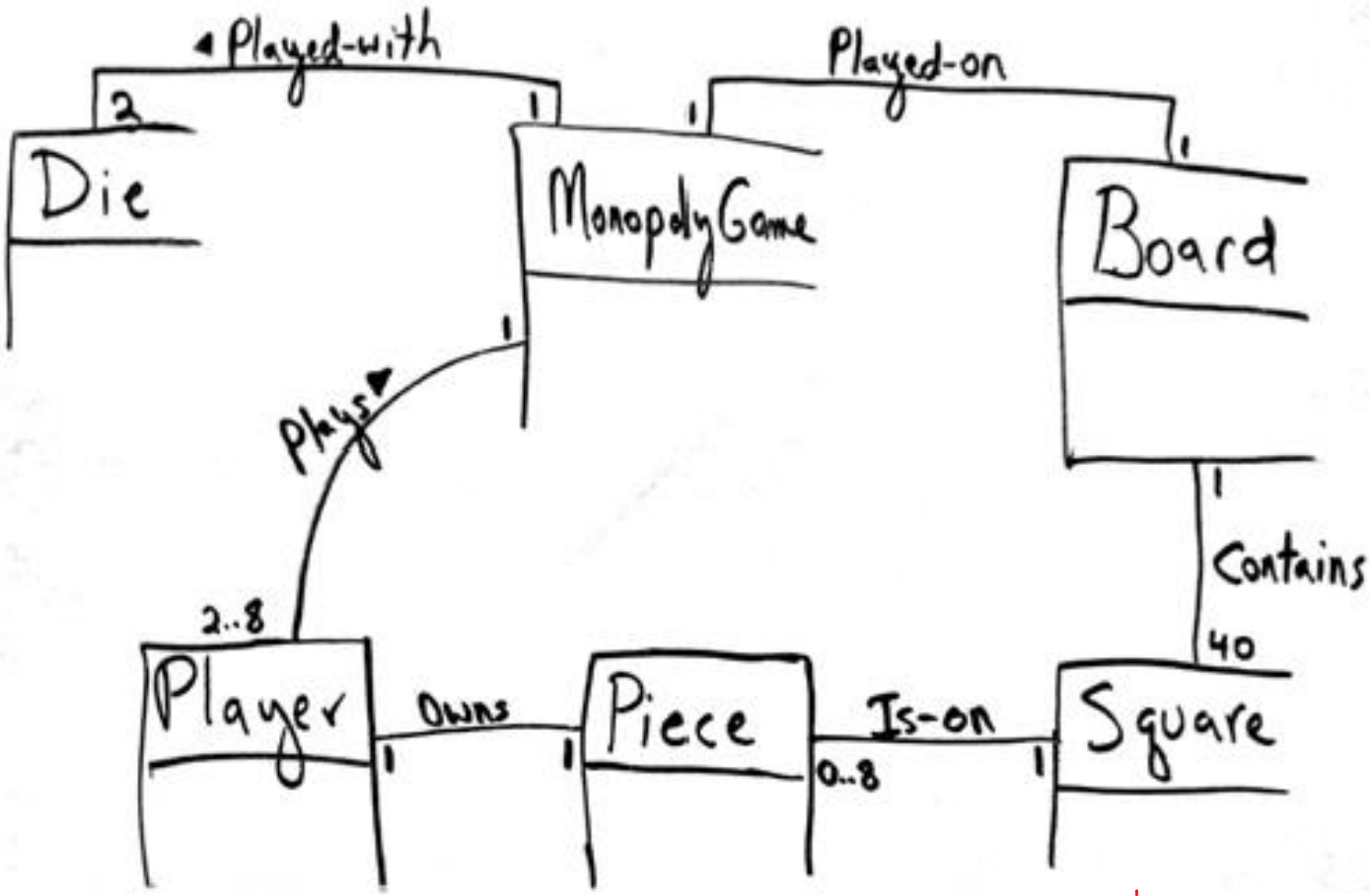
依据此表，可以让我们找到更多的关联关系。

Category 种类	Examples 例子
A is a transaction related to another transaction B	CashPaymentSale CancellationReservation
A is a line item of a transaction B	SalesLineItemSale
A is a product or service for a transaction (or line item) B	ItemSalesLineItem (or Sale), FlightReservation
A is a role related to a transaction B	CustomerPayment PassengerTicket

Common Associations List

Category	Examples
A is a physical or logical part of B	DrawerRegister, SquareBoard, SeatAirplane
A is physically or logically contained in/on B	RegisterStore, ItemShelf, SquareBoard, PassengerAirplane
A is a description for B	ProductDescriptionItem, FlightDescriptionFlight
A is known/logged/recorded/ reported/captured in B	SaleRegister, PieceSquare, ReservationFlightManifest
A is a member of B	CashierStore, PlayerMonopolyGame, PilotAirline
A is an organizational subunit of B	DepartmentStore, MaintenanceAirline
A uses or manages or owns B	CashierRegister, PlayerPiece, PilotAirplane
A is next to B	SalesLineItemSalesLineItem, SquareSquare, CityCity

Monopoly Partial Domain Model with Associations

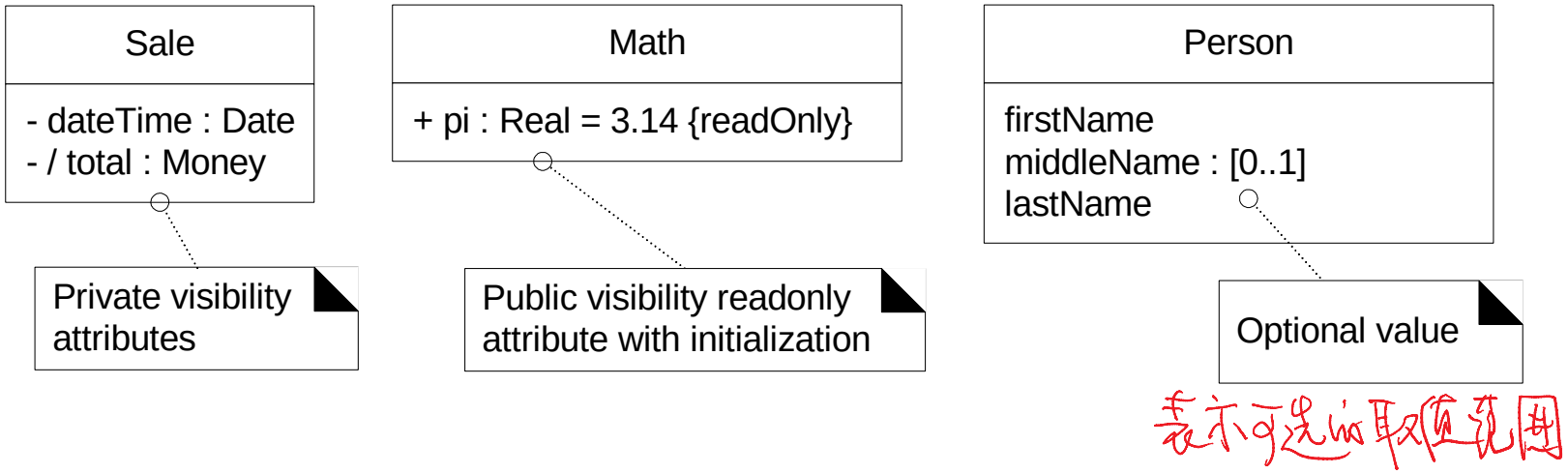


领域模型例(非完整模型)

第二个概念：属性

Attributes

- An Attribute is a logical data value of an object
- Full syntax:
  visibility Name : *type multiplicity = default {property string}*
- +pi : Real=3.14159 {read-only}
- Attribute notation in UML



在领域模型中何时需要引入属性？

When to Show Attributes?

- Include attributes that the requirements (e.g., use cases) suggest or imply a need to remember information.
- A receipt (which reports the information of a sale) in the *Process Sale* use case normally includes a date and time, the store name and address, and the cashier ID, among many other things.
  - *Sale* needs a *dateTime* attribute.
  - *Store* needs a *name* and *address*.
  - *Cashier* needs an *ID*.

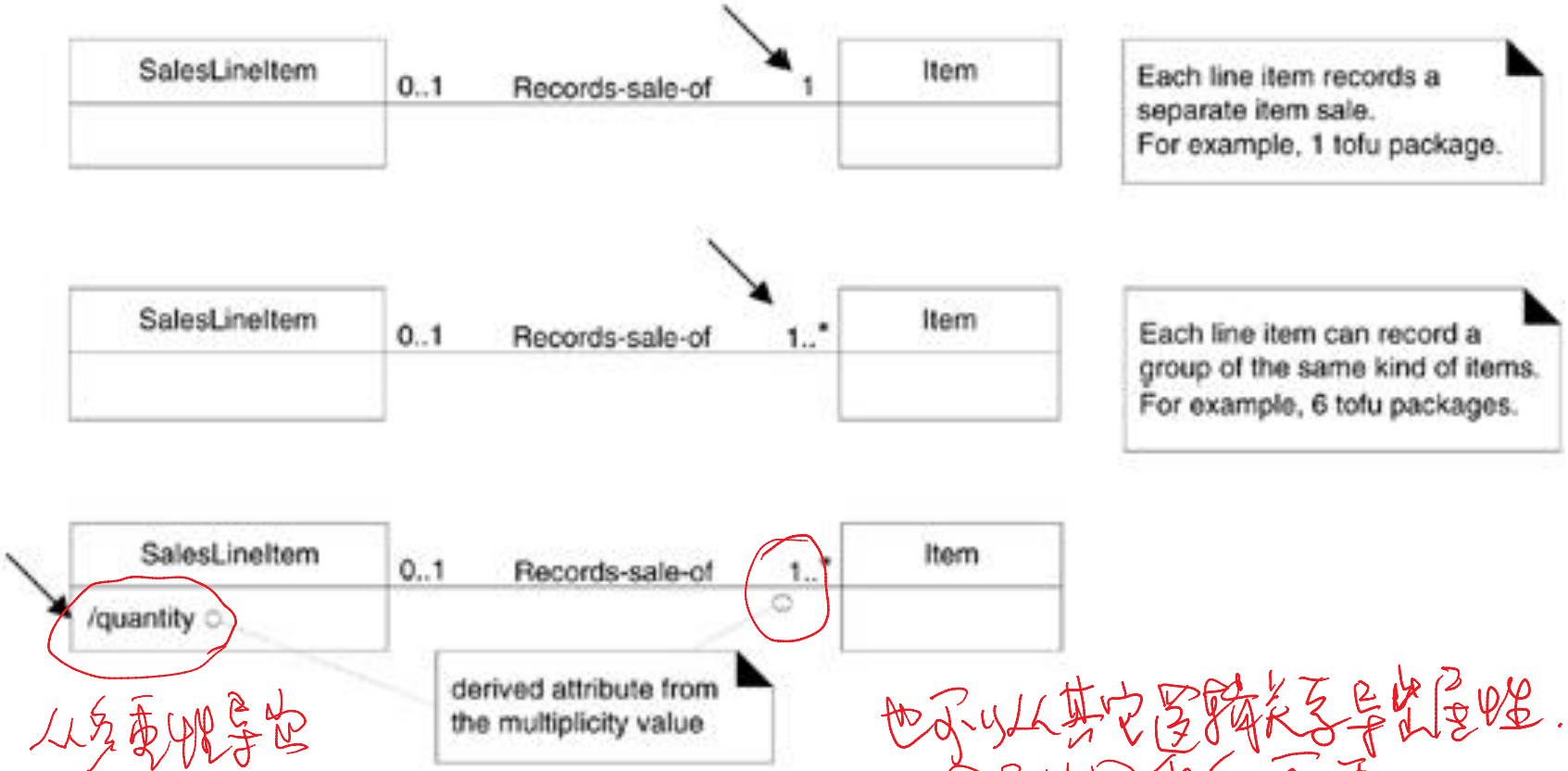
} 需求的要求

Where to Record Attribute Requirements?

- Notice that, subtly, *middleName : [0..1]* is a requirement or domain rule, embedded in the domain model.
- Although this is just a conceptual-perspective domain model, it probably implies that the software perspective should allow a missing value for *middleName* in the UI, the objects, and the database.
- I suggest placing all such attribute requirements in the UP Glossary, which serves as a data dictionary.
- Another alternative is to use a tool that integrates UML models with a data dictionary; then all attributes will automatically show up as dictionary elements.

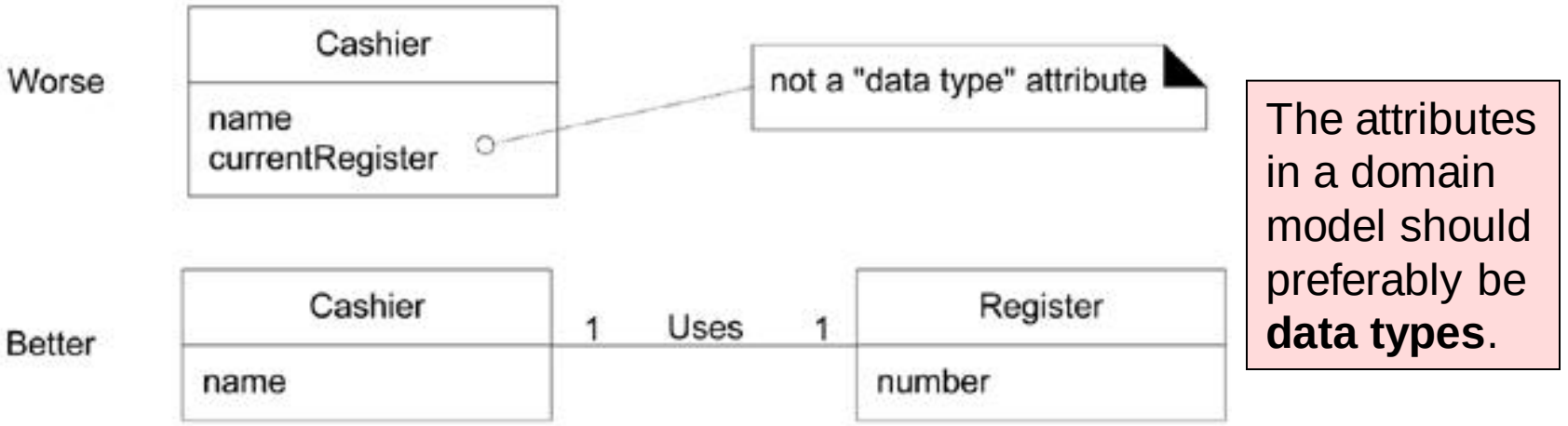
Derived Attributes

- You can have derived attributes denoted by / before the name



Attribute Types

- Usually primitive data types, as well as things like Color, DateTime, Zip code, etc.
- It should not normally be a class, such as Sale or Airport
- Relate conceptual classes with an association, not an attribute. No Attributes Representing Foreign Keys
- Attributes in code are okay



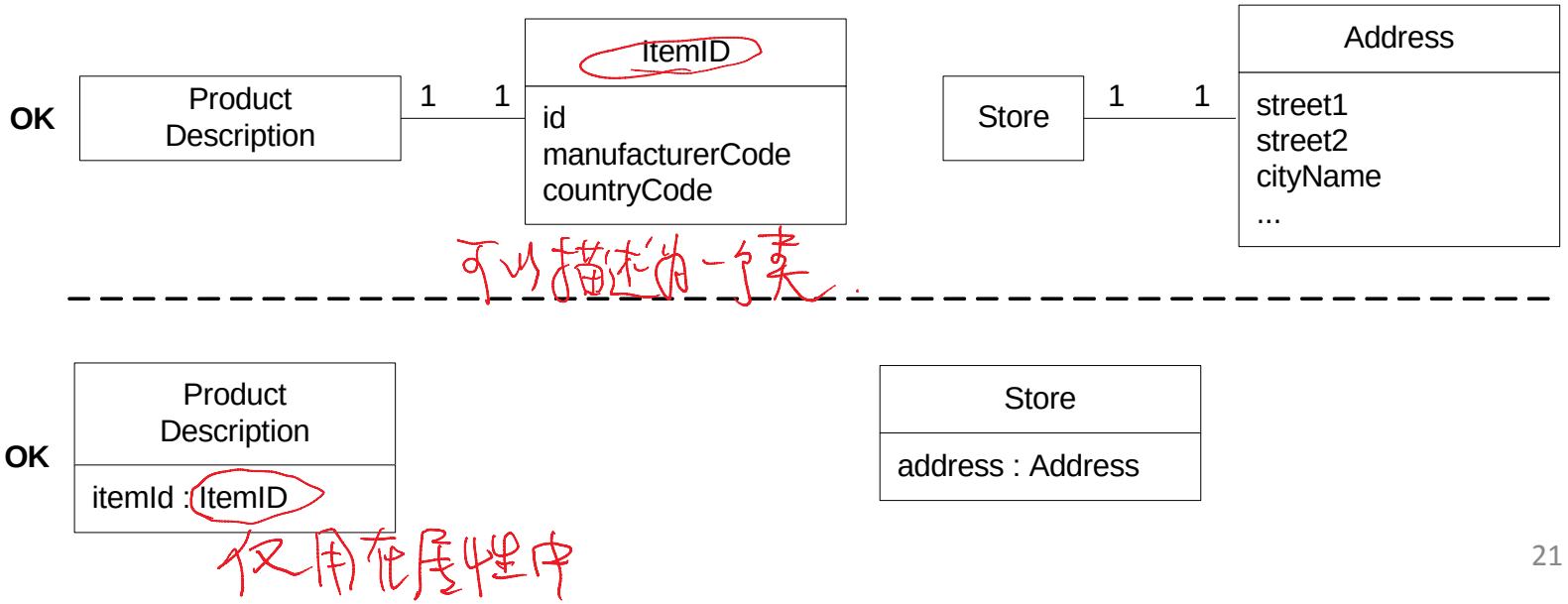
什么时候 Attributes 可以是类

When to Define New Data Type Classes?

- Things like ItemID or ClaimNumber are not always simple data types, even though they look like them
- It can be its own class:
  - If the attribute contains separable pieces. 分部
    E. g., a Claim Number has the state, year, and day encoded in it
  - If it has operations associated with it 有操作
  - If it has other attributes, such as a sale price 有属性
  - If it has units, such as currency 有单位
  - If it is an abstraction of one or more types with these properties 有多种类型

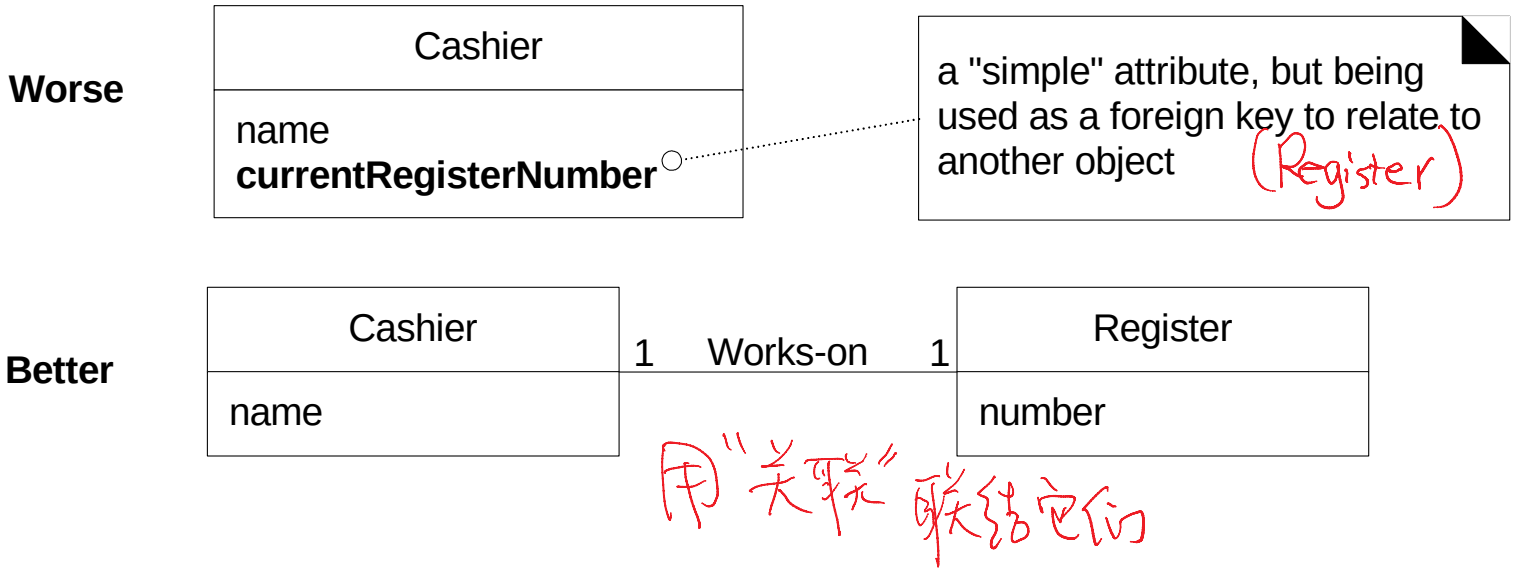
Where to Illustrate These Data Type Classes? (in domain model)

- Should the *ItemID* class be shown as a separate class in a domain model? It depends on what you want to emphasize in the diagram.
- Two ways to indicate a data type property of an object.



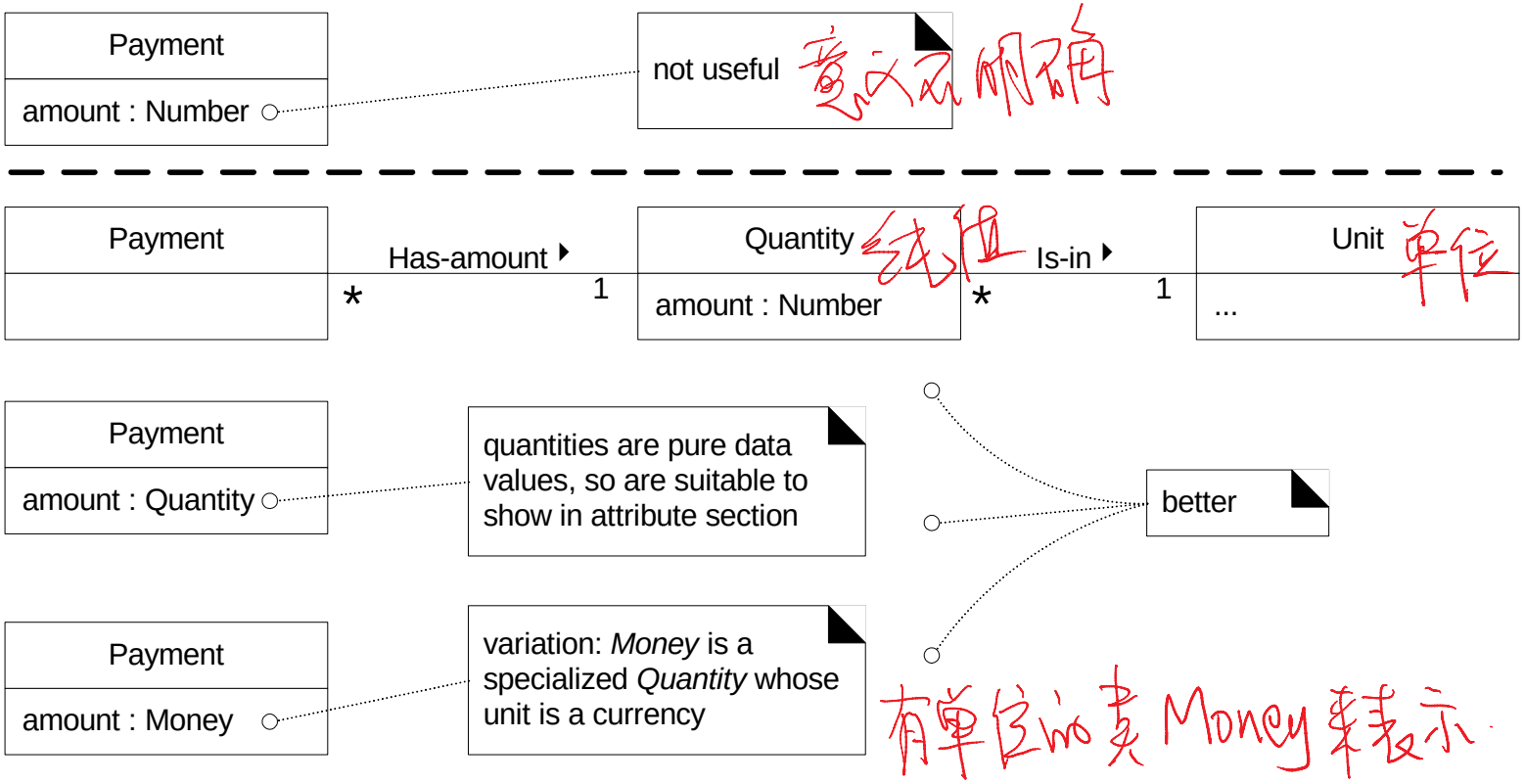
No Attributes Representing Foreign Keys

- Attributes should *not* be used to relate conceptual classes in the domain model.
- The most common violation of this principle is to add a kind of **foreign key attribute**, as is typically done in relational database designs, in order to associate two types.
- **Relate types with an association, not with an attribute**.

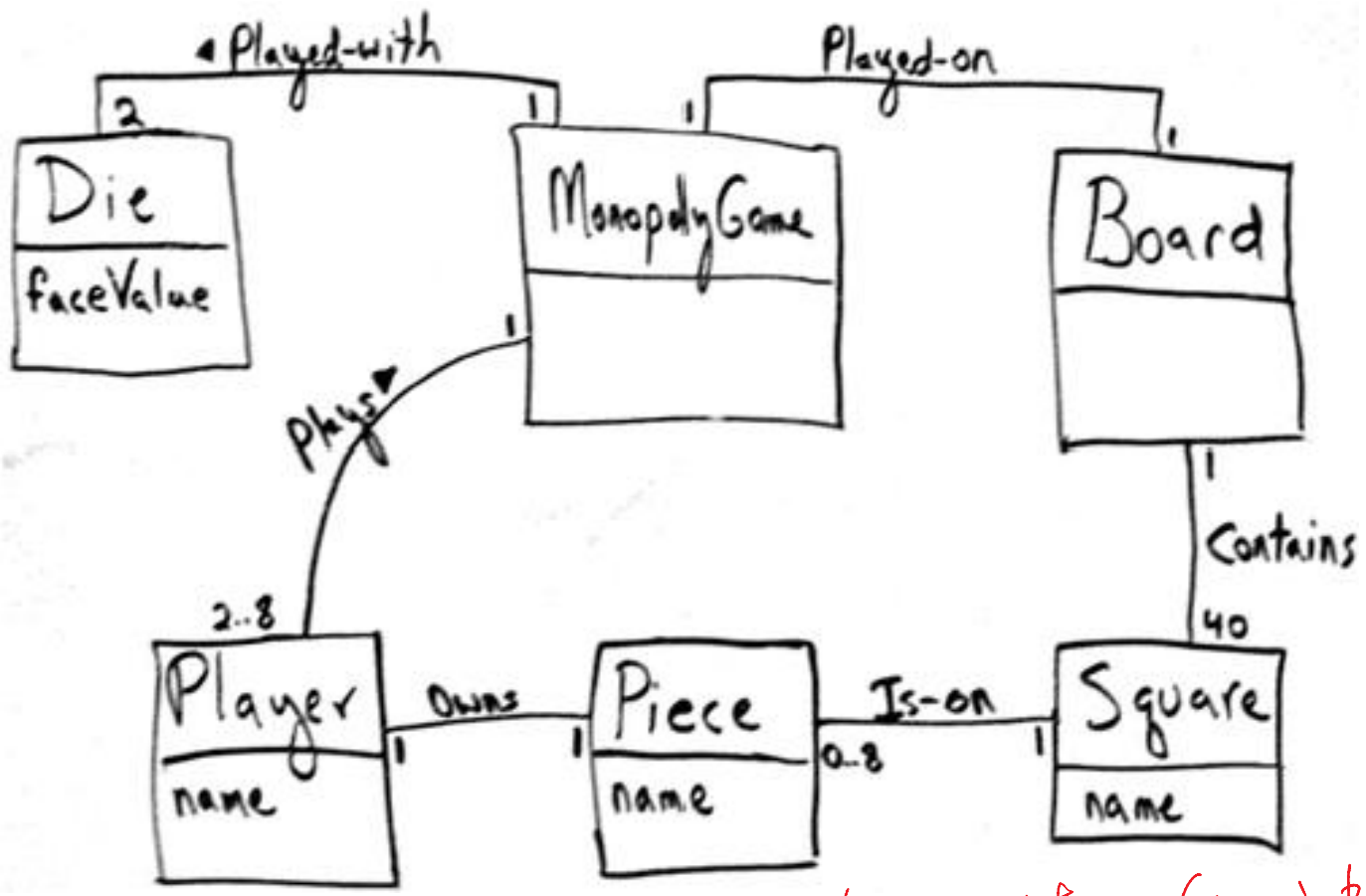


Quantities with Units

- Most numeric quantities should not be represented as plain numbers.
- Represent them with distinct classes with an associated unit
- To show Quantity specializations. Money is a kind of quantity whose units are currencies. Weight is a quantity with units such as kilograms or pounds.



Monopoly Partial Domain Model



本次选用中的领域模型(非完整)

Is the Domain Model Correct?

- No, but it’s a good approximation
- It gets better with each iteration; don’t try to get it all at once
- Domain model usually done in the elaboration phase

Iterative and Evolutionary Domain Modeling

Inception

- Domain models are not strongly motivated in inception, since inception's purpose is not to do a serious investigation, but rather to decide if the project is worth deeper investigation in an elaboration phase.

Elaboration

- The Domain Model is primarily created during elaboration iterations, when the need is highest to understand the noteworthy concepts and map some to software classes during design work.

Iterative and Evolutionary Domain Modeling

- In iterative development, we incrementally evolve a domain model over several iterations.
- In eachiteration, the domain model is limited to the prior and current scenarios under consideration,
  - rather than expanding to a "big bang" waterfall-style model that early on attempts to capture all possible conceptual classes and relationships.

Iterative and Evolutionary Domain Modeling

Domain Models Within the UP

Discipline	Artifact	Incep.	Elab.	Const.	Trans.
	Iteration	I1	E1..En	C1..Cn	T1..T2
Business Modeling	**Domain Model**		s		
Requirements	Use-Case Model (SSDs)	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		
	Data Model		s	r	