Com S 362
Object-Oriented Analysis & Design

Domain Model

Domain Model

- Illustrates noteworthy concepts of the domain
- Inspiration for designing software objects

Recap

Responsibility-Driven Design

- First described by Rebecca Wrifs-Brock and Brian Wilkerson (1989)
- The dominant approach to Object-Oriented Design
- Describes objects in terms of their responsibilities and their interactions with other objects.
- Full process is most applicable to clean slate design (new product, new major feature, etc)
- Component skills and mind-set are applicable in all projects

How to find
Conceptual Classes & Responsibilities

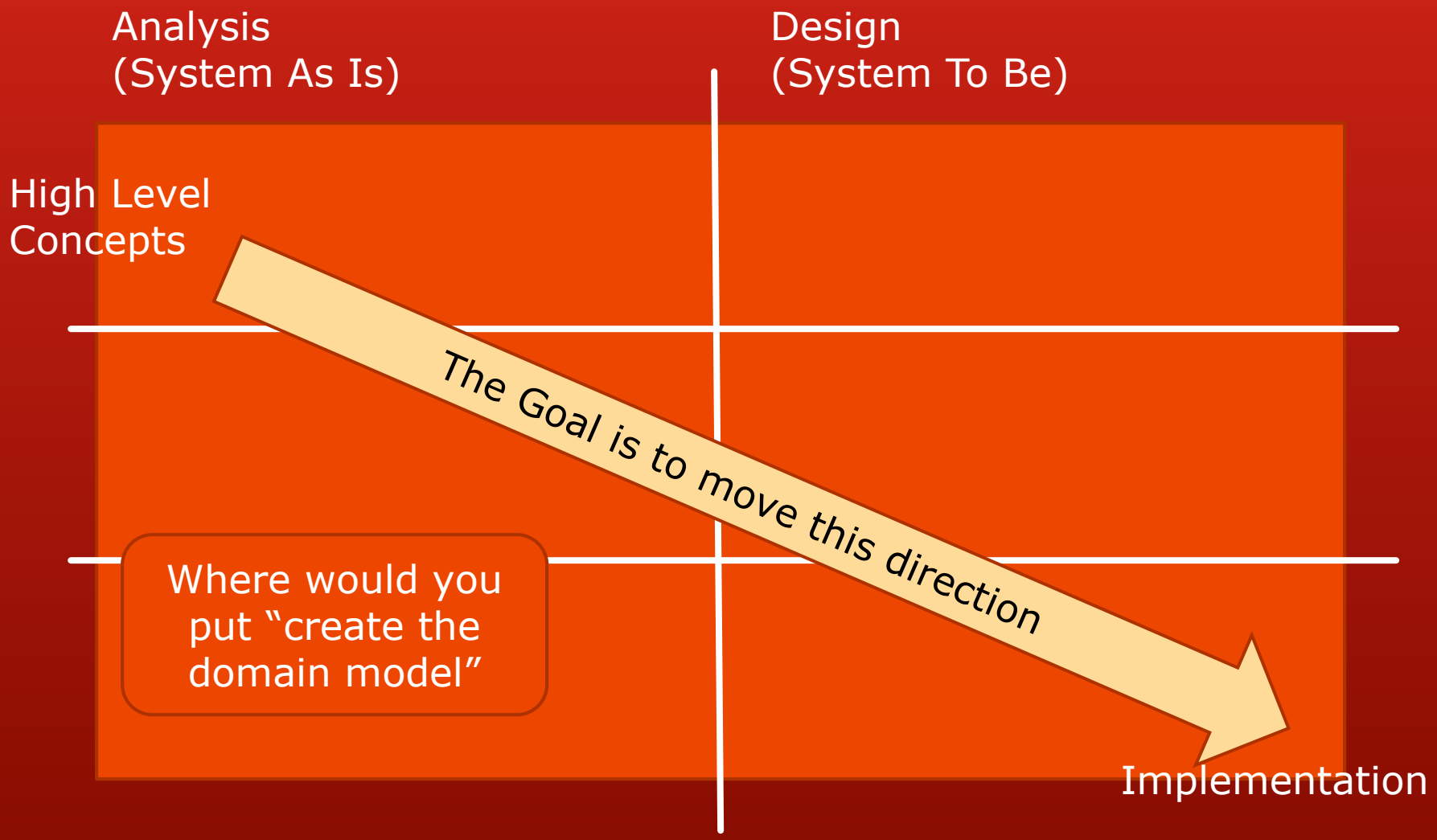
- Noun-verb analysis
  - Any text describing the system to be or the problem space can be a resource.
  - Nouns are possibly classes
  - Verbs may describe interactions between classes, and thus suggest responsibilities
- Other Sources:
  - Existing domain models (including reverse engineering existing code)
  - Category Lists

Example Category List

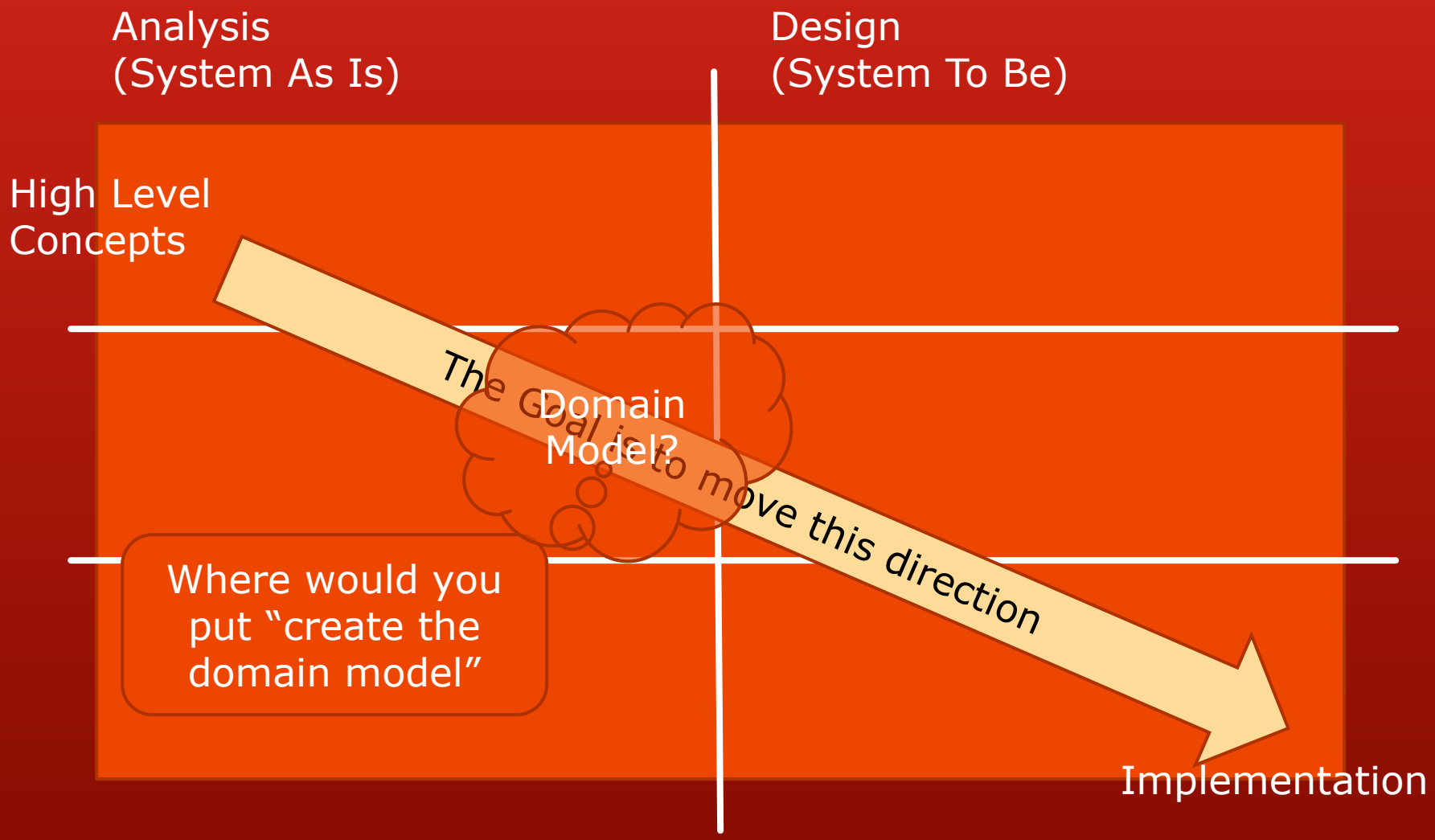
Conceptual Class Category	Examples
physical or tangible objects	**Register Airplane**
specifications, designs, or descriptions of things	*ProductSpecification FlightDescription*
places	*Store Airport*
transactions	*Sale, Payment Reservation*
transaction line items	*SalesLineItem*
roles of people	*Cashier Pilot*
containers of other things	*Store, Bin Airplane*
things in a container	*Item Passenger*
other computer or electro-mechanical systems external to the system	*CreditPaymentAuthorizationSystem AirTrafficControl*
abstract noun concepts	*Hunger* *Acrophobia*
organizations	*SalesDepartment ObjectAirline*
events	*Sale, Payment, Meeting Flight, Crash, Landing*
processes (often not represented as a concept, but may be)	**SellingAProduct BookingASeat**
rules and policies	*RefundPolicy CancellationPolicy*
catalogs	*ProductCatalog PartsCatalog*
records of finance, work, contracts, legal matters	*Receipt, Ledger, EmploymentContract Main tenanceLog*
financial instruments and services	*LineOfCredit Stock*
manuals, documents, reference papers, books	*DailyPriceChangeList* `RepairManual`

from: http://csis.pace.edu/~marchese/CS616/Lec5/se_l5a.htm Based on Larman pp 140-141.

Analysis vs Design



Analysis vs Design



Introducing:

Domain Models

The Domain Model is the analysis artifact that most directly translates to the design domain.

"A Domain Model is a *visual* representation of **conceptual classes** or real-situation objects in a domain." [1]

"The Domain Model is a *visual dictionary* of the noteworthy abstractions, domain vocabulary, and information content of the domain"

Note:

The domain model is **not** software classes or objects.

Based on course notes by Dr. Hridish Rajan.

Domain Model

Scope of Domain Models

- A Domain Model can include:
  - Domain classes - representation of types of conceptual objects
  - Attributes - description of a named slot of domain classes which holds a separate value for different class instances
  - Associations - description of relationship between domain classes
  - Multiplicity - how many instances of class A can be associated with an instance of class B.

Elaborating the Domain Model

- Start with candidate classes

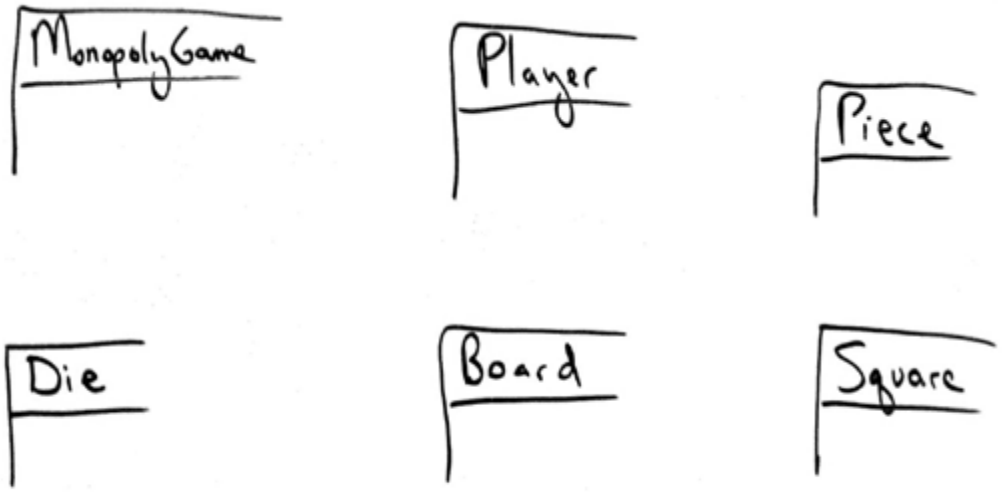


Figure 9.8. Initial Monopoly domain model.

Elaborating the Domain Model

- Capture associations (static relationships)
- Responsibilities and Collaborations (CRC cards) help identify associations

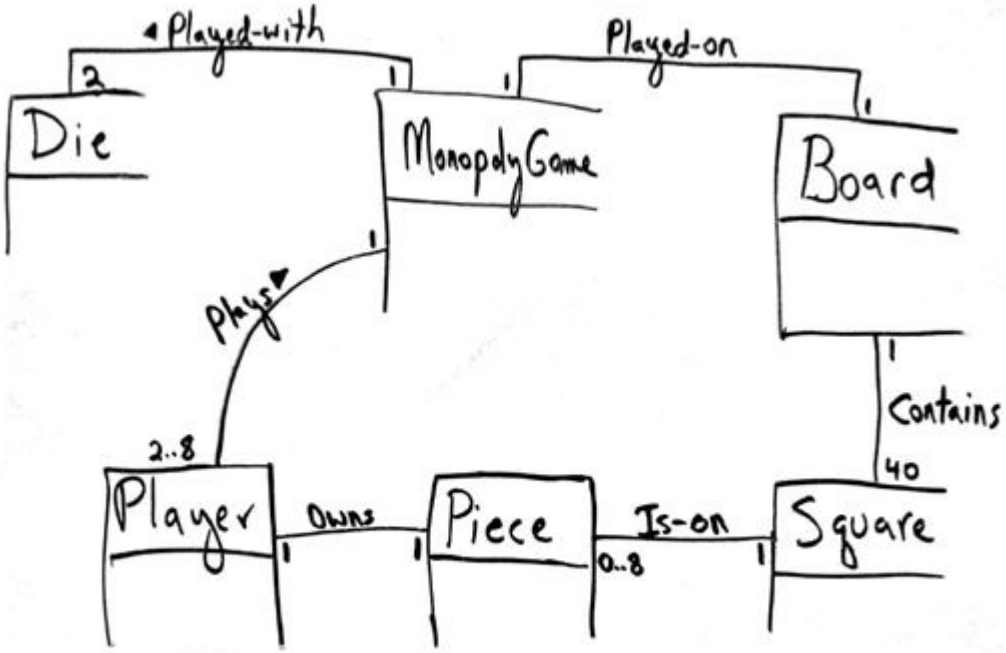


Figure 9.18. Monopoly partial domain model.

Elaborating the Domain Model

- capture important attributes

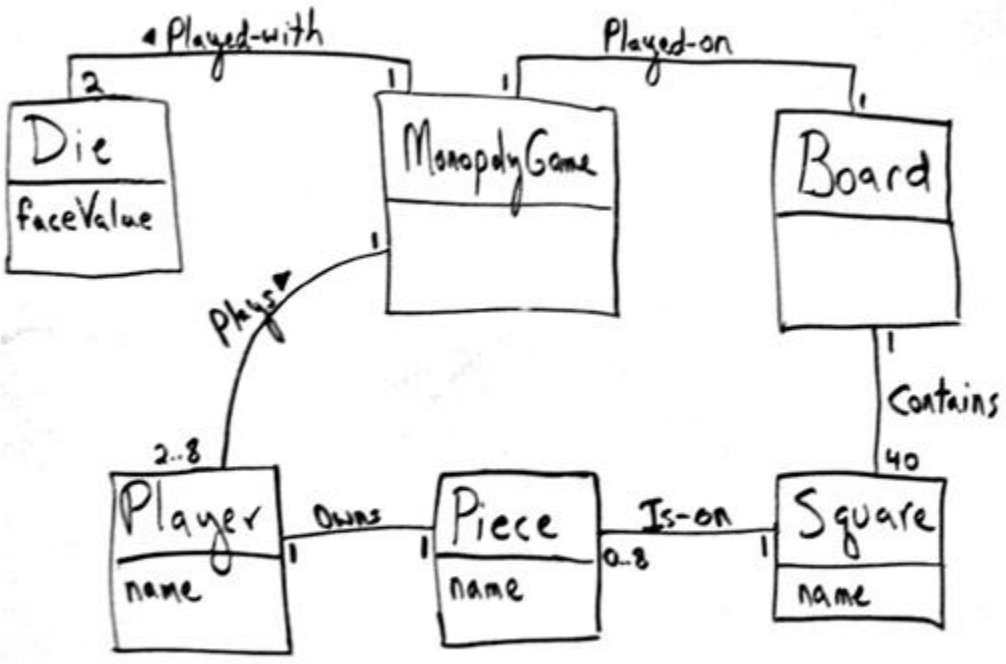
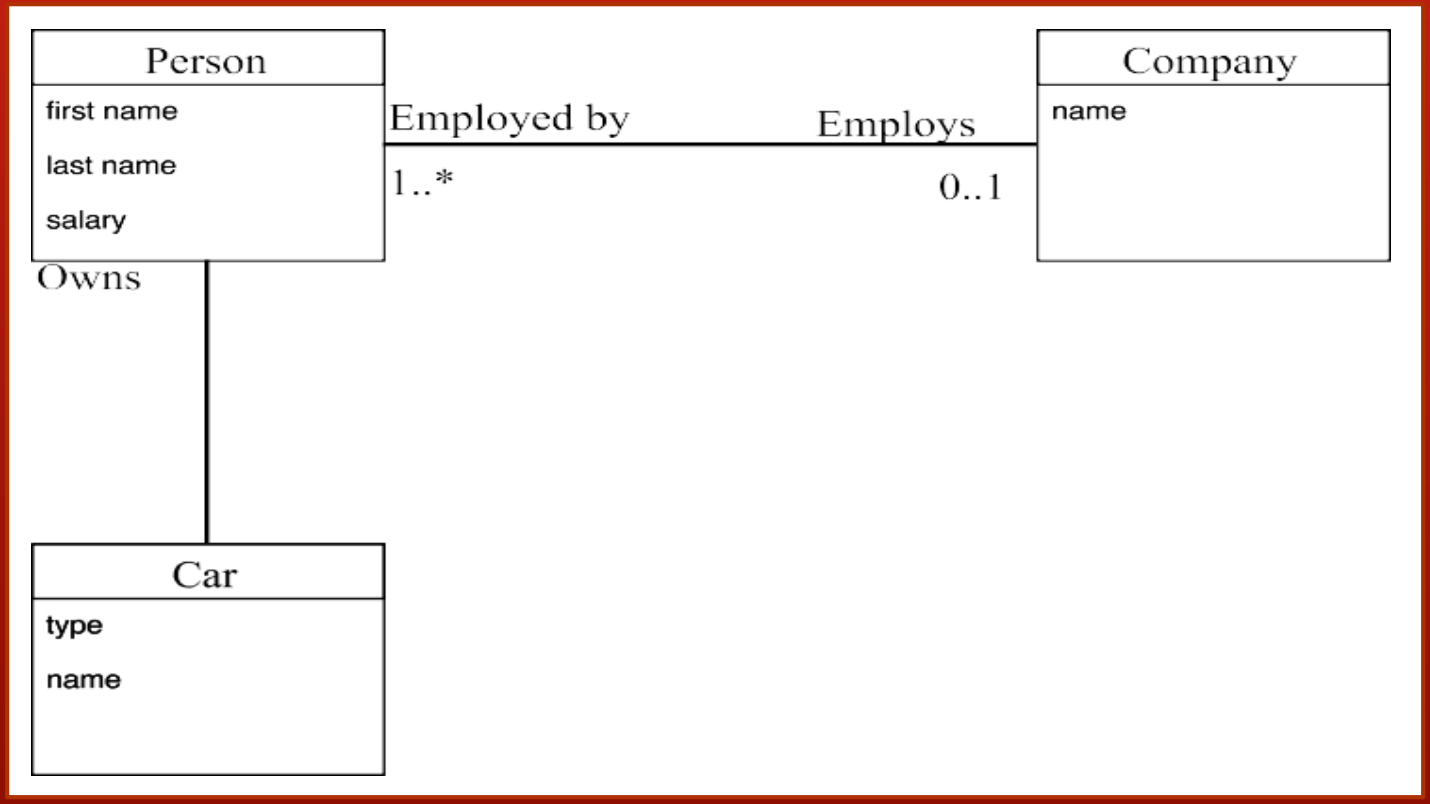


Figure 9.28. Monopoly partial domain model.

Notice there are no methods. During this phase, we will capture interaction information in a communication diagram.

Domain Model

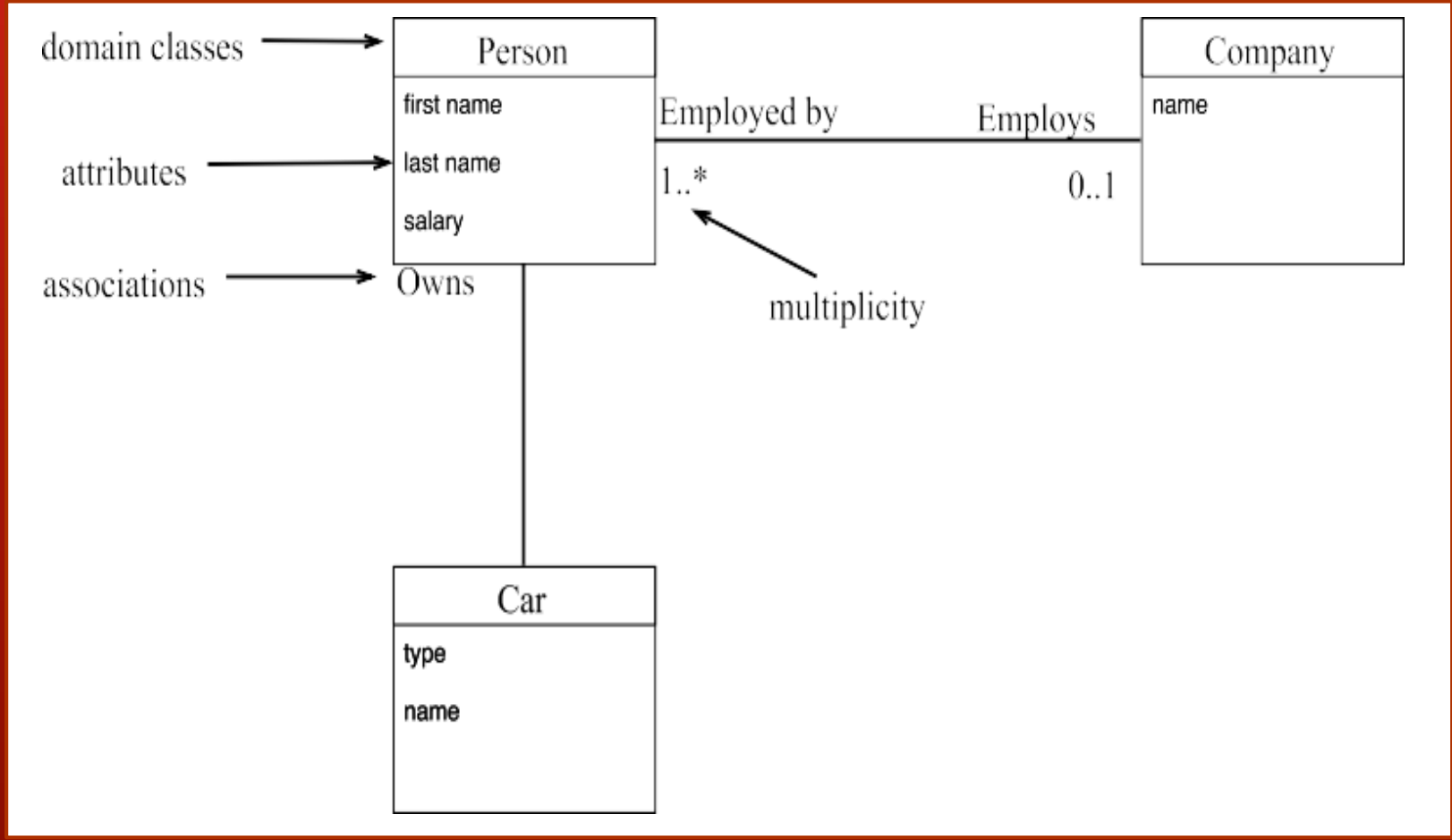
An Example Domain Model



Based on course notes by Dr. Hridish Rajan.

Domain Model

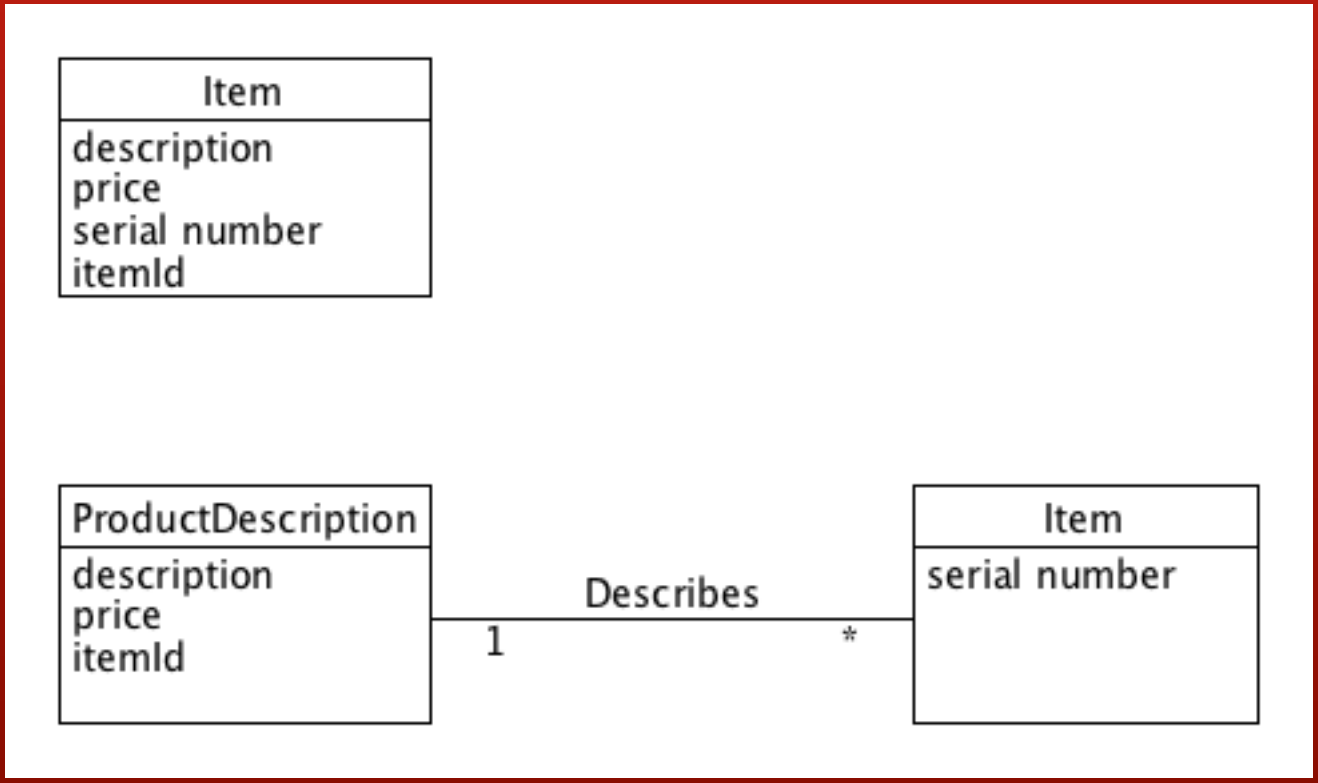
Annotated Example



Based on course notes by Dr. Hridish Rajan.

Description Classes

- Common for multiple entities to share a common description



Domain Model

Some points to keep in mind

- Domain models are static representations, NOT dynamic information, such as entities sending messages to another
- The objects represent real world concepts, NOT software entities
- We are not writing programs
- Domain models aids designers and developers with visualizing the domain of the problem, and helps with thinking by using the terms in the model

Based on course notes by Dr. Hridish Rajan.

Additional Guidelines

- Avoid assumptions about the specifics of the user interface.
- Be consistent in your representation for attributes with complex data type. In smaller models, opt for an association between the owning object and the complex type. (fig 9.24)
- Don’t confuse with data base schema; avoid properties that function as foreign keys - instead show the association.
- With quantities, use unit types, not primitive language types. (GBP instead of float).
- Specifying multiplicity helps identify role and avoid associations like “listof”.
- Associations are static relationships - not method calls.