Domain Model

- Why:
  Domain modeling helps us to *identify the relevant concepts and ideas of a domain*
- When:
  Domain modeling is done during *object-oriented analysis*
- Guideline:
  Only create a domain model for the tasks at hand

The Domain Model illustrates noteworthy concepts in a domain.

- The domain model is created during object-oriented analysis to decompose the domain into concepts or objects in the real world
- The model should identify the set of conceptual classes (The domain model is iteratively completed.)
- It is the basis for the design of the software

The domain model is also called conceptual model, domain object model or analysis object model.

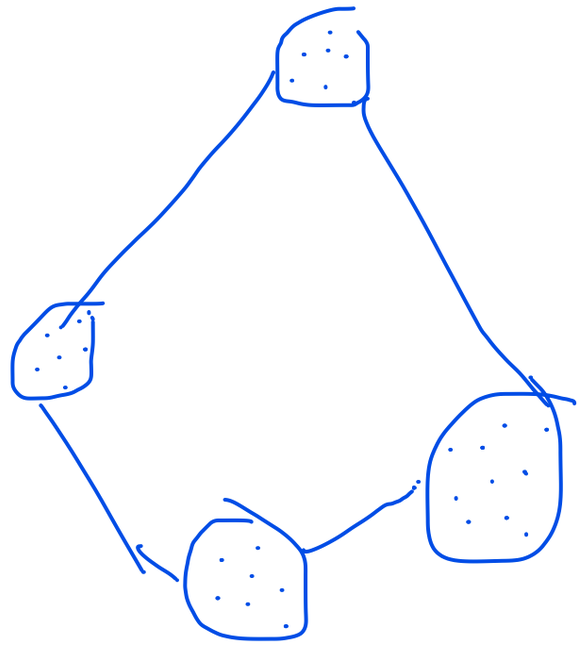
- Conceptual classes are ideas, things or objects in the domain

  A conceptual class has a symbol representing the class, an intension and an extension that defines the set of examples to which the conceptual class applies.

- Domain concepts / conceptual classes are not necessarily software objects as, e.g., in Java, C#, Ruby, ...!

To visualize domain models the UML class diagram notation is used.

- However, **no operations** are defined in domain models
- Only ...
  - domain objects and conceptual classes
  - associations between them
  - attributes of conceptual classes



The result is a conceptual perspective model.

Visualizing Domain Models

- Exemplified

  (By means of a basic course management system.)

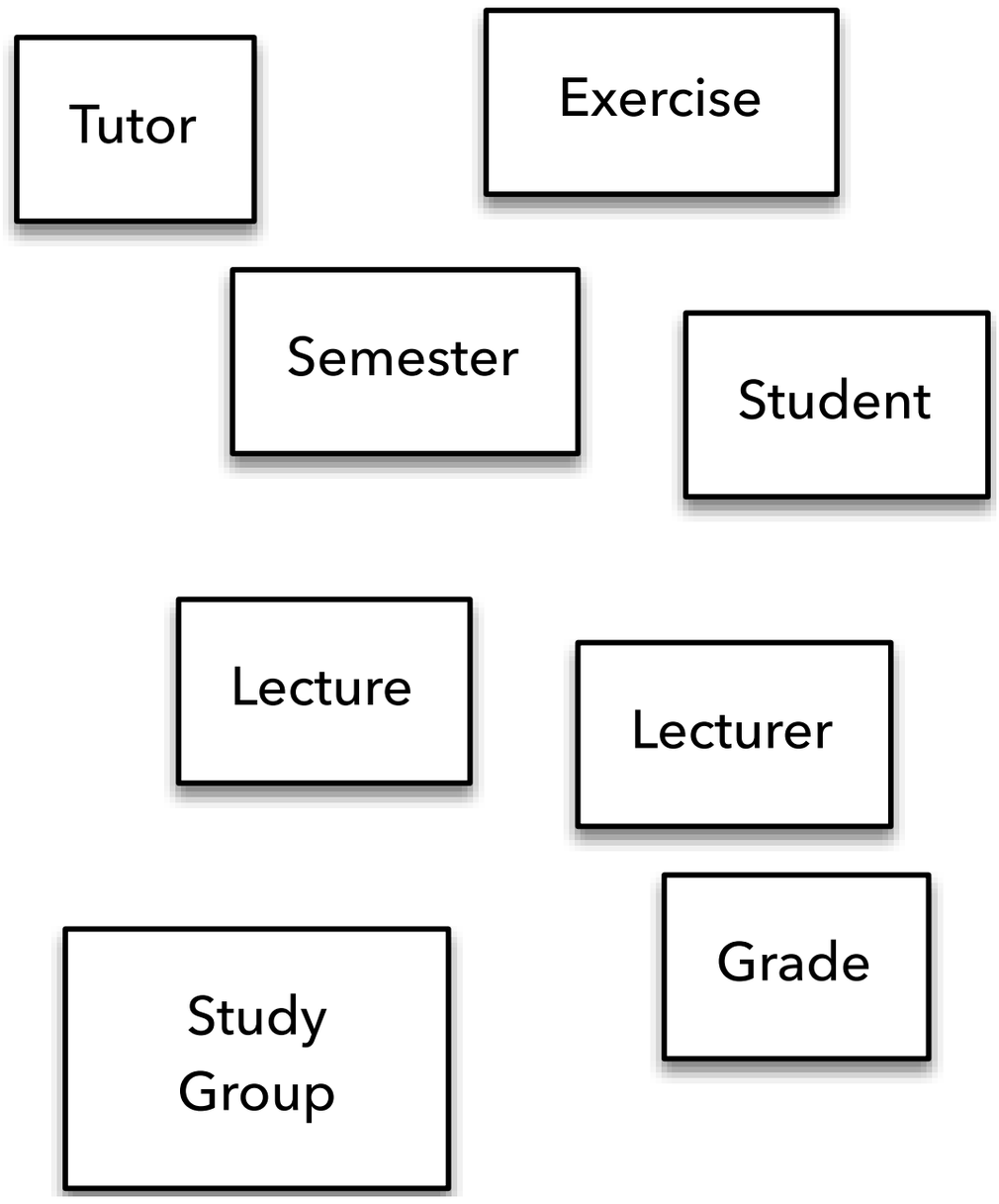
Statements about a Course Management System

- During a semester a lecturer reads one or more lectures
- Sometimes the lecturer is on leave to do research, in this case (s)he does not give a lecture
- A student usually attends one or more lectures, unless (s)he has something better to do
- During the semester there will be several exercises which are meant to be solved by small study groups
- Each student is assigned to one particular study group for the whole semester
- A study group consists of two to three students
- After submission of a solution by a study group it is graded by a tutor
- ...

A class describes a set of objects with the same semantics, properties and behavior.

When used for domain modeling, it is a visualization of a real world concept.

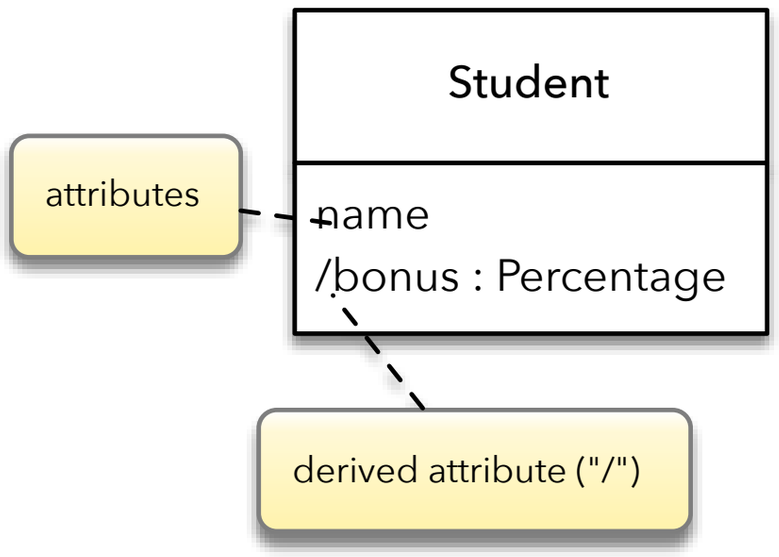
- During a semester a lecturer reads one or more lectures
- A student usually attends one or more lectures, ...
- During the semester there will be several exercises...
- Each student is assigned to one particular study group for the whole semester
- ... it is graded by a tutor

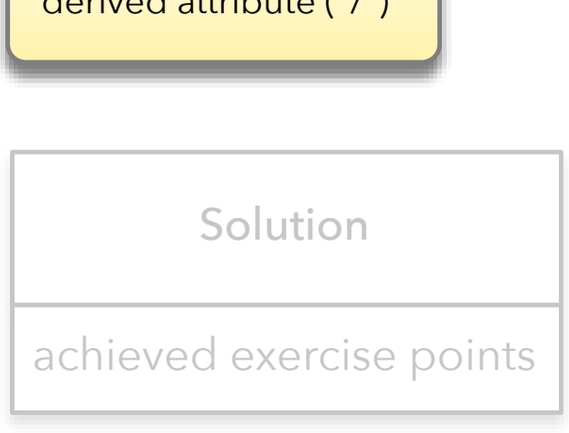


Attributes are logical data values of an object.

It is useful to identify those attributes of conceptual classes that are needed to satisfy the information requirements of the current scenarios under development.

- ... after submitting a solution it is graded by a tutor
- The bonus is a relative bonus that reflects the relative number of exercise points gained during the semester
- The bonus is derived.



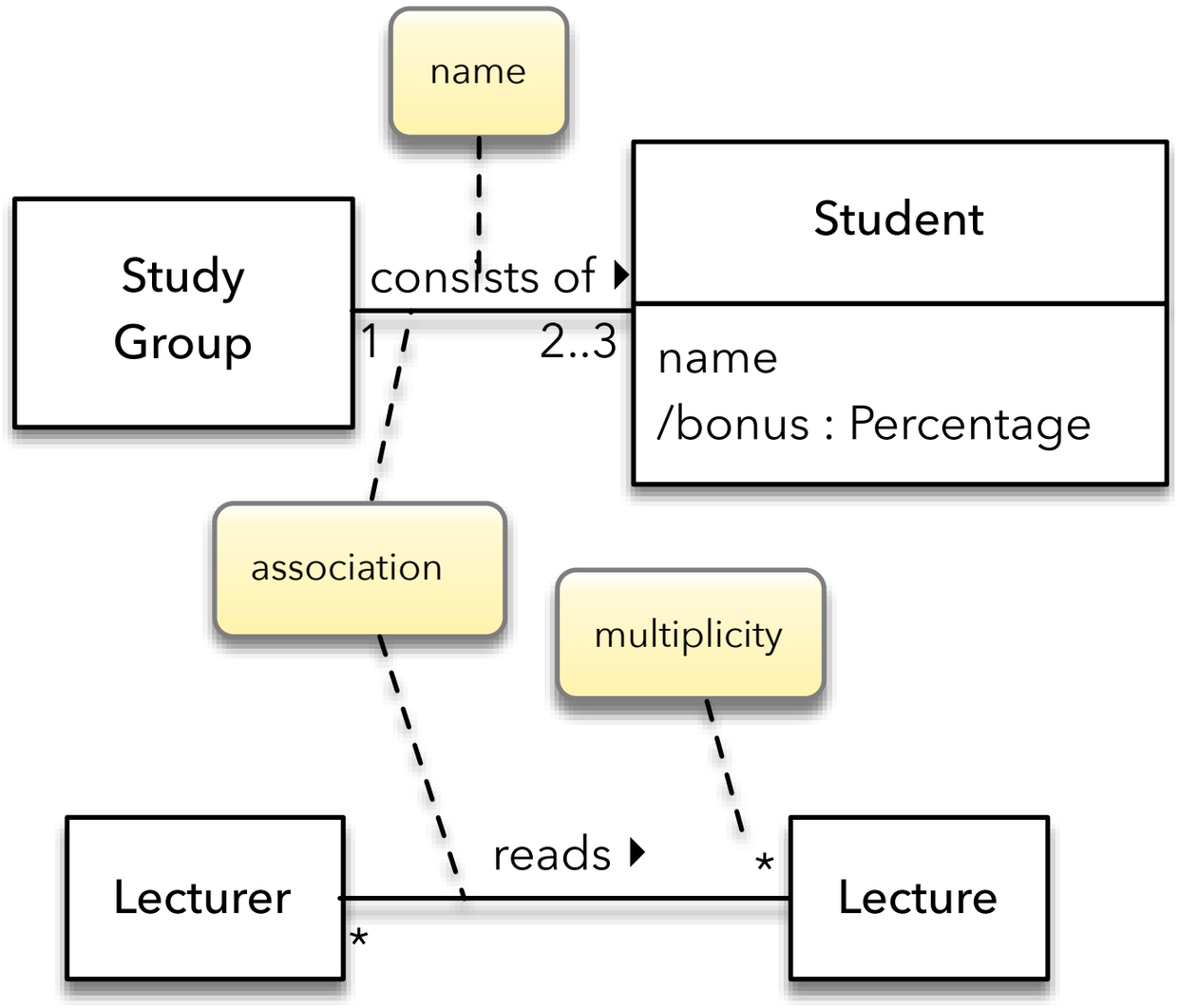


An association is a relationship between classes.

The ends of an association are called roles.

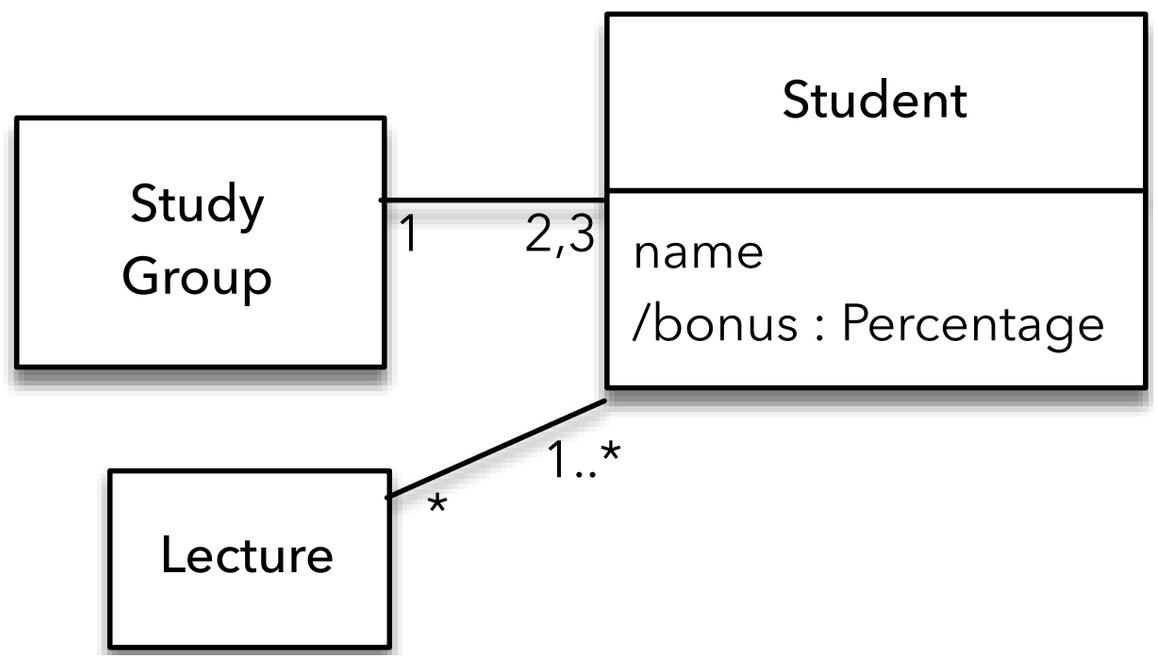
Roles optionally have a multiplicity, name and navigability.

- A lecturer **reads** one or more lectures...
- A student usually attends one or more lectures...
- A study group **consists of** two to three students...
- During the semester there will be several exercises
- ...



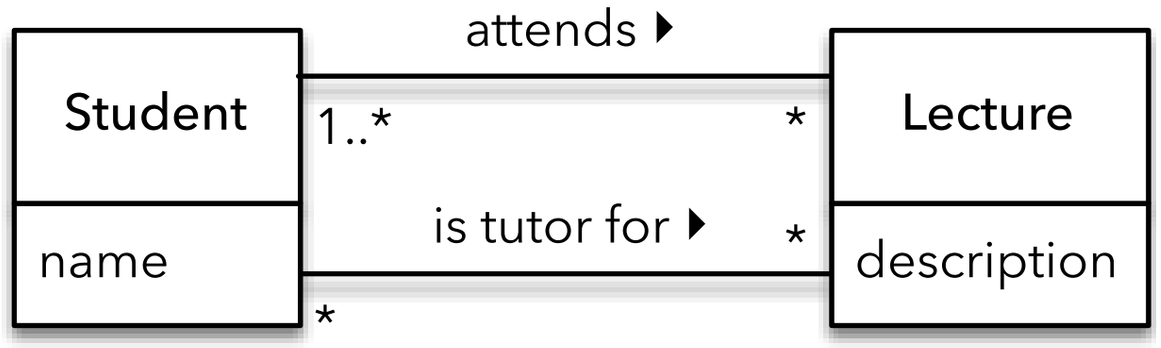
The multiplicity defines how many instances of a class A can be associated with one instance of a class B at any particular moment.
(e.g., * ≙ zero or more; 1..10 between 1 and 10; 1,2 one or two)

- A student usually attends **one or more** lectures, unless (s)he has something better to do
- A study group consists of **two to three** students...
- ...

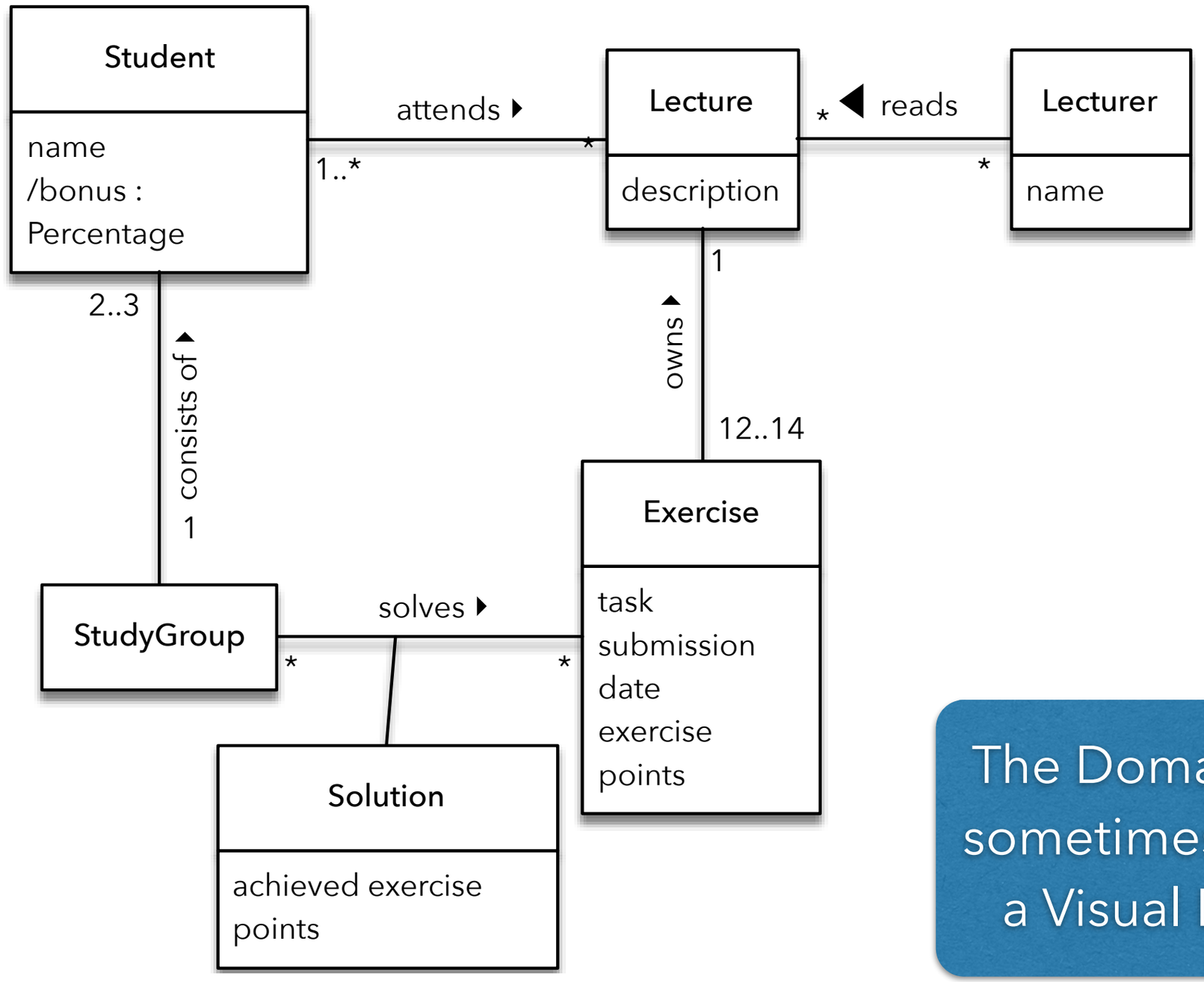


Two classes can have multiple associations.

- A student usually attends one or more lectures, unless the student has something better to do
- A study group consists of two to three students; after submitting a solution it is graded by a tutor who is also a student



A preliminary domain model for a course management system.



The Domain Model is sometimes also called a Visual Dictionary.

Domain Modeling

- Overview

Excerpt Of the Domain Model For the POS System

Which are noteworthy
domain concepts / domain objects?

How to create the domain model?

(If we are not readily familiar with the domain.)

1. Find the conceptual classes

   Strategies:

   a. *Reuse or modify an existing model*

   b. *Use a category list*

   c. *Identify noun phrases*

   } explained in the following

2. Draw them as classes in a UML class diagram
3. Add associations and attributes

Use the domain vocabulary; e.g. a model for a library should use names like "Borrower" instead of customer.

How to create the domain model?

Use a category list to find the conceptual classes.

Conceptual Class Category	Classes (for the POS system)
Business transactions...	Sale, Payment
Transaction line items...	SalesLineItem
Product or service related to a transaction or transaction line item.	Item
Where is the transaction recorded?	Register
Roles of people or organizations related to the transaction; actors in use cases.	Cashier, Customer, Store
Place of transactions.	Store
Noteworthy events, often with a time or place that needs to be remembered.	Sale, Payment
...	...

How to create the domain model?
Identify noun phrases to find the conceptual classes(linguistic analysis).

- Identify the nouns and noun phrases in textual descriptions of a domain and consider them as candidate conceptual classes or attributes
- A mechanical noun-to-class mapping isn't possible; words in natural languages are ambiguous; i.e. the same noun can mean multiple things and multiple nouns can actually mean the same thing

"Use Cases" are also an excellent source for identifying conceptual classes.

How to create the domain model?
Identify noun phrases to find the conceptual classes.

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates the inventory. The customer receives a receipt from the system and then leaves the store with the items.

How to create the domain model?
Identify noun phrases to find the conceptual classes.

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates the inventory. The customer receives a receipt from the system and then leaves the store with the items.

How to create the domain model?

Identify noun phrases to find the conceptual classes.

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each item. [illegible] e-item details [illegible] ch the system [illegible] es the inventory. The customer receives a receipt from the system and then leaves the store with the items.

Identified candidate conceptual classes:

Customer, **Item**, **Cashier**, **Store**, **Payment**, **Sales Line Item**, **Inventory**, **Receipt**, **Sale**.

How to create the domain model?

Identify noun phrases to find the conceptual classes.

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each item. [illegible] e-item details [illegible] ch the system [illegible] es the inventory. The customer receives a receipt from the system and then leaves the store [illegible]

Identified candidate conceptual classes:
Customer, **Item**, **Cashier**, **Store**, **Payment**, **Sales Line Item**, **Inventory**, **Receipt**, **Sale**.

Should Receipt be in the domain model?

Guidelines when to include a candidate conceptual class that reports Information into the domain model

- In general, it is not useful since all information is derived or duplicated from other sources
- If it has a specific semantics w.r.t. the business, then it should be included

Identified candidate conceptual classes:

Customer, **Item**, **Cashier**, **Store**, **Payment**, **Sales Line Item**, **Inventory**, **Receipt**, **Sale**.

Should Receipt be in the domain model?

Guidelines when to include a candidate conceptual class that reports Information into the domain model

- In general, it is not useful since all information is derived or duplicated from other sources
- If it has a specific semantics w.r.t. the business, then it should be included

Identified candidate conceptual classes:

Customer, **Item**, **Cashier**, **Store**, **Payment**, **Sales Line Item**, **Inventory**, **Receipt**, **Sale**.

A receipt is just a report of a sale and a payment...

Guidelines when to include a candidate conceptual class that reports Information into the domain model

- "Facts":
  - Identified candidate conceptual classes: ... Receipt, ...
  - A receipt is just a report of a sale and a payment...
- Should Receipt be in the domain model?
- Well, it depends....
  - ... if we just consider the current scenario then receipt should not be part of the domain model; a receipt is just a report
  - ... if we also consider how to handle returns then a receipt represents an important concept on its own
- How about legal restrictions...?

Domain Requirement

Guidelines when to include a description class into the domain model

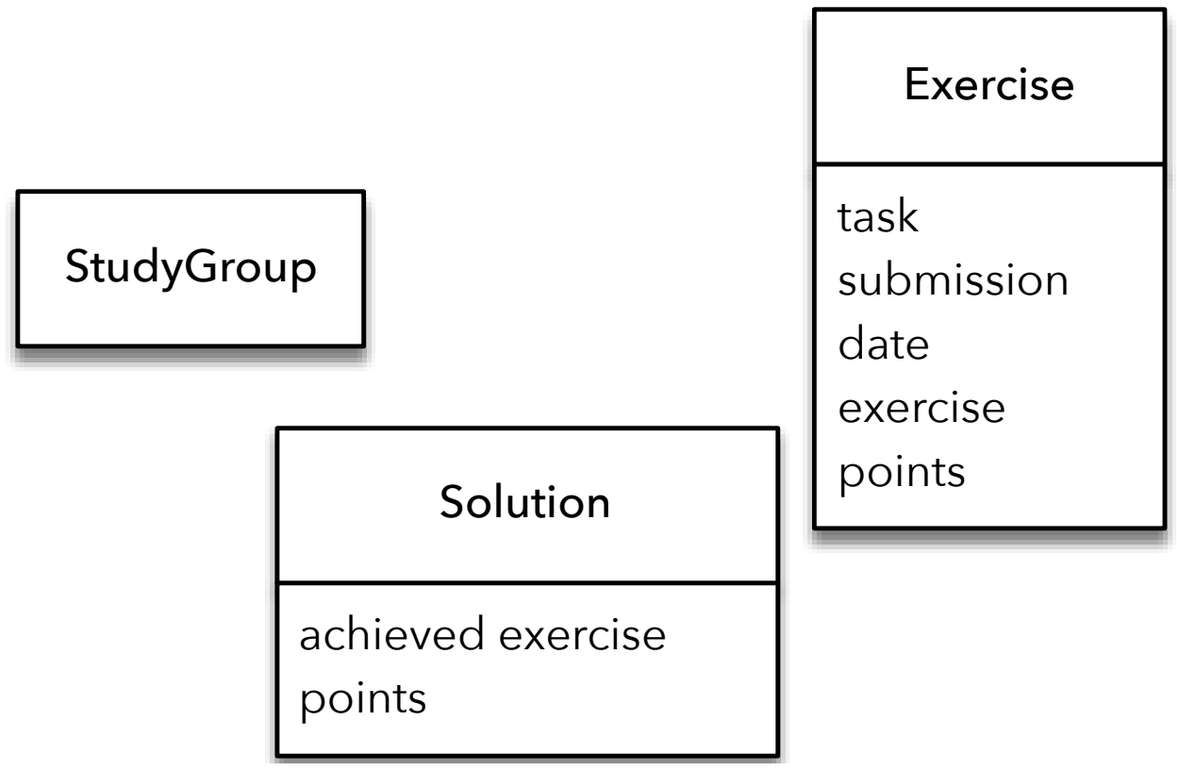
- A description class contains information that describes something else
  Examples:
  - a product description records the price, picture and text of an item
  - a flight description contains information about the flight (number) and the source and target destinations

Guidelines when to include a description class into the domain model

- A description class should be added to the domain model when:
  - There needs to be a description about an item or service, independent of the current existence of any examples of those items or services
  - Deleting instances of things they describe results in a loss of information that needs to be maintained, but was incorrectly associated with the deleted thing
  - It reduces redundant or duplicated information

When should I model something as an attribute or a class?

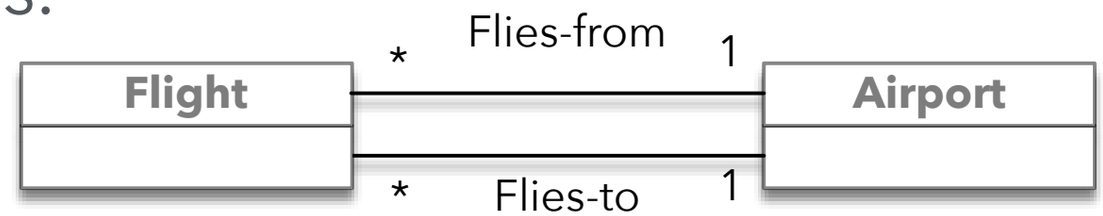
- Rule of Thumb:
  If we do not think of some conceptual class X as a number, date or text in the real world, X is probably a conceptual class, not an attribute.



When should I model something as an attribute or a class?

Let’s assume that we develop an airline reservation system.

- Rule of Thumb:
  If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attribute.
- Should **destination** be an attribute of flight, or a conceptual class airport?
  - A destination airport is a building at a specific place, it is not just a number or some text.
  - Hence, it should be a conceptual class.
- How about the name of the airport?



When should I add an association to the domain model?

- Rule of Thumb:
  Include associations in the domain model for which knowledge of the relationship needs to be preserved for some duration.

We are working on the conceptual model; we are not modeling associations at the software level.

When should I add an association to the domain model?

- When an association is among the common associations list:
  - A is a transaction related to another transaction B
  - A is a line item of a transaction B
  - A is a product or service for a transaction B
  - A is a role related to a transaction B
  - A is physical or logical part of B
  - ...

When should I add an association to the domain model?

- E.g. the relation between a **Sale** and a **SalesLineItem** needs to be remembered
- However, it is not necessary to store the relation between a **Cashier** and a **ProductDescription** that he looks up

Name an association based on a ClassName-VerbPhrase-ClassName format.

The verb phrase creates a sequence that is readable and meaningful.

- Good examples:
  - Player Is-on Square
  - Sale Paid-by CashPayment
- Bad examples:
  - Sale Uses CashPayment
    (Uses is usually generic and doesn't tell us anything.)
  - Player Has Square
    (Has is usually generic and doesn't tell us anything.)

The attributes in a domain model should preferably be “primitive” data types (w.r.t. the domain!).

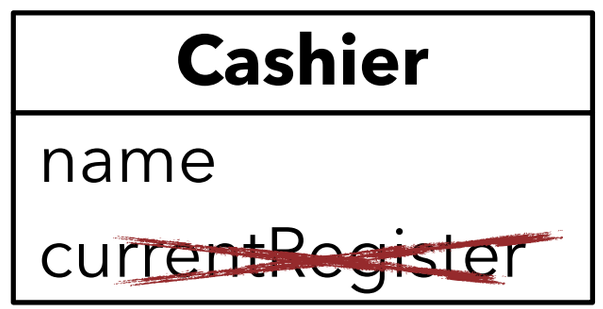
- Very common data types include: Boolean, Date, Number, Character, String, Address, Color, Phone Number, …
- Consider modeling quantities as classes to be able to associate units
  e.g. the data type of the amount attribute of a payment should indicate the currency

Cashier
name
currentRegister

The attributes in a domain model should preferably be “primitive” data types.

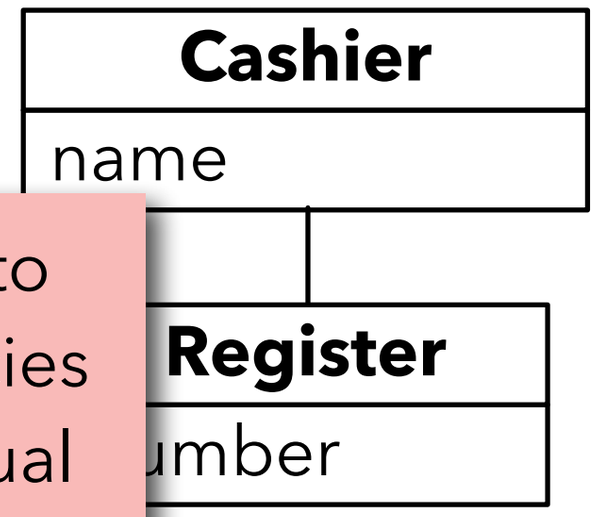
- Very common data types include: Boolean, Date, Number, Character, String, Address, Color, Phone Number, …
- Consider modeling quantities as classes to be able to associate units
  e.g. the data type of the amount attribute of a payment should indicate the currency

avoid:



recommended:

Use associations to model dependencies between conceptual classes; do not use attributes.

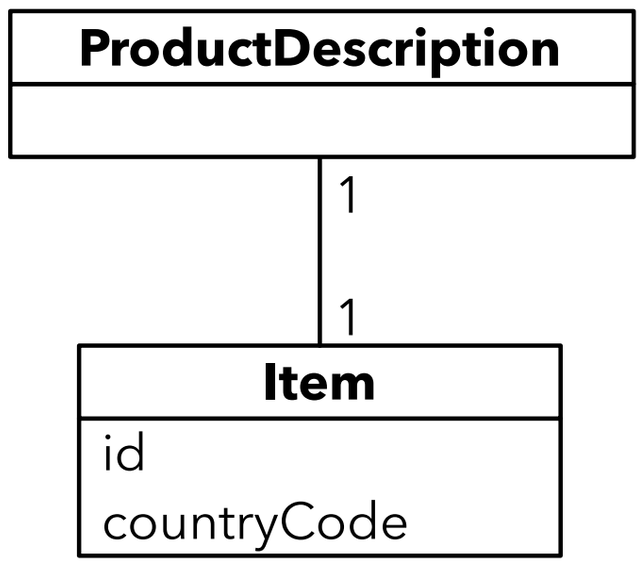


Consider defining a new data type class for something that is initially considered a string.

- If the string is composed of separate sections
  e.g., phone number, name of person,...
- If different operations are associated with the string
  e.g., social security number
- If the string has other attributes
- If the string is a quantity with a unit
  e.g., money has a unit for currency

avoid:

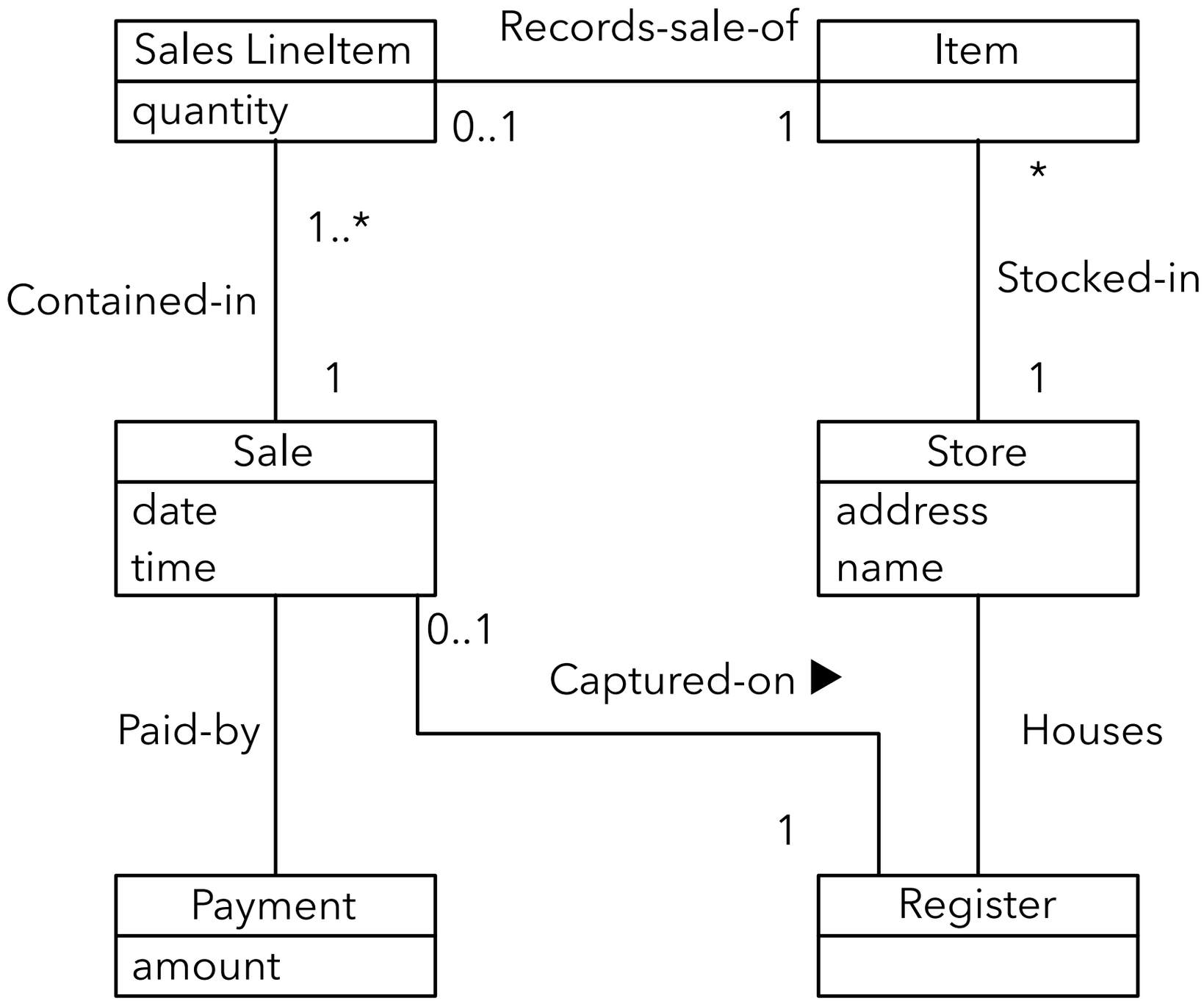
recommended:



Excerpt Of the Domain Model For the POS System

Which are noteworthy
domain concepts / domain objects?

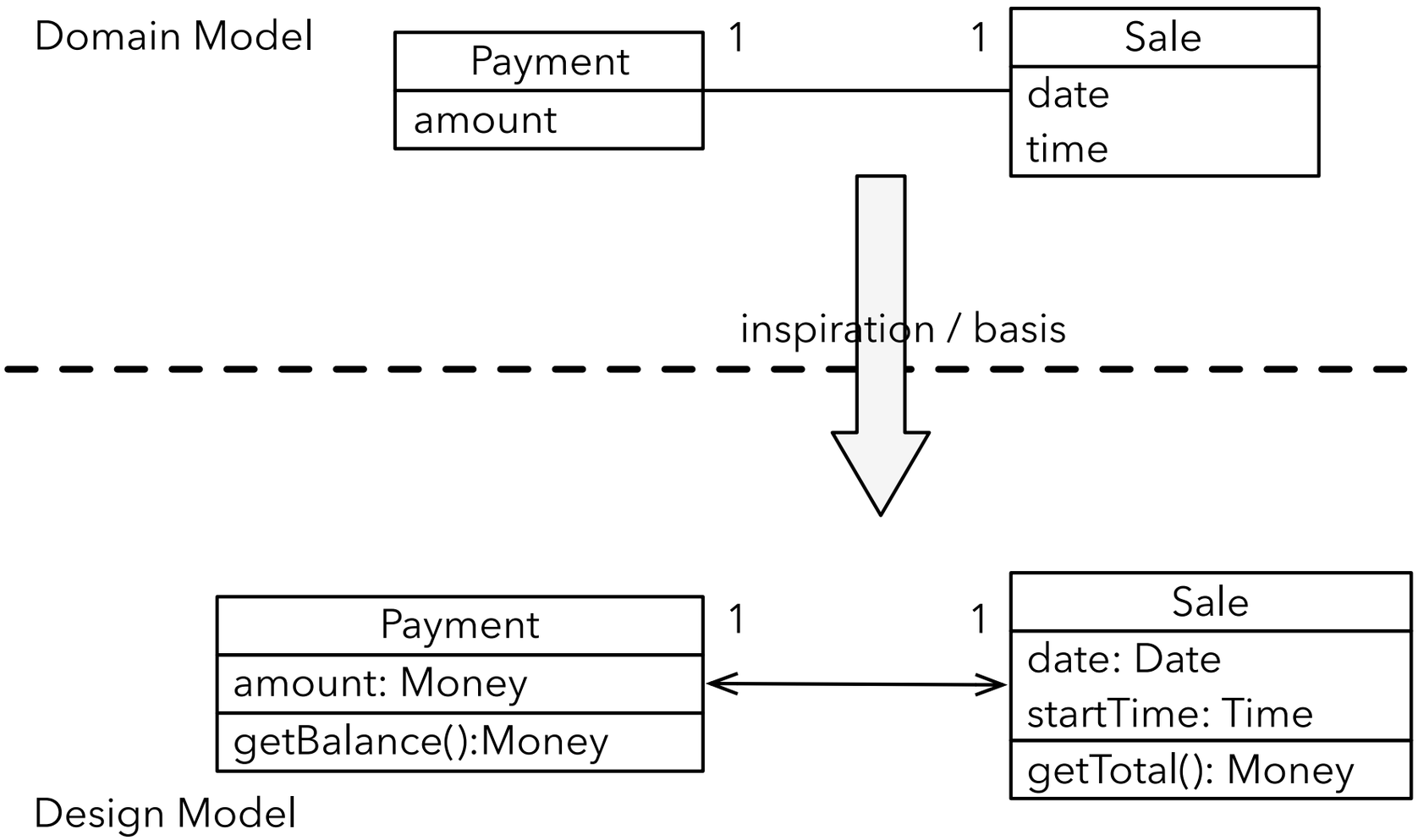
Excerpt Of the Domain Model For the POS System



Domain Model and Domain Modeling

- Summary

The domain model serves as a source of inspiration for the design model which will be discussed later on.

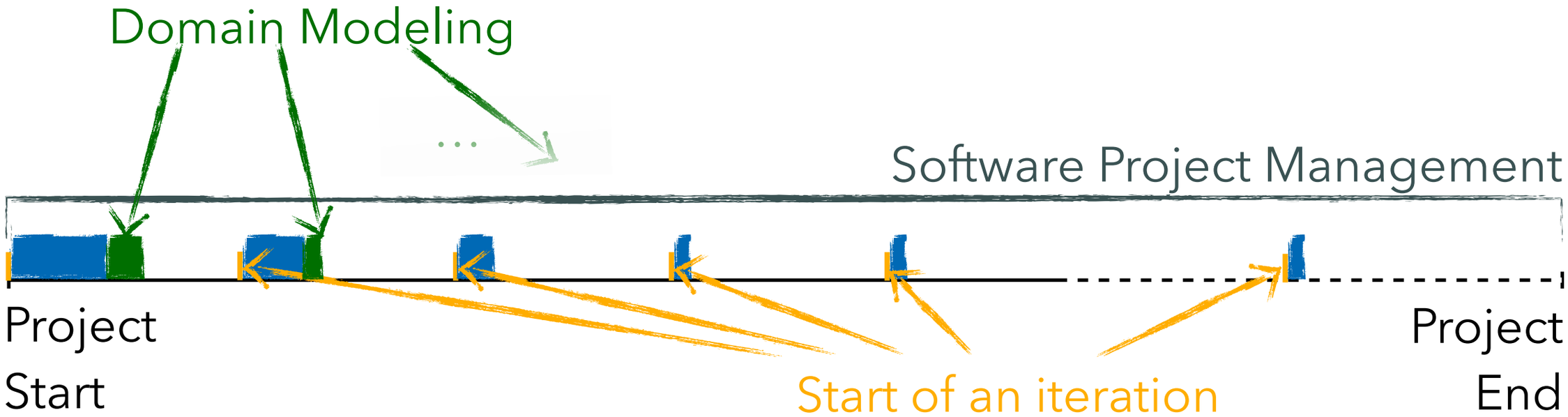


By using the domain model as a direct inspiration for software classes the representational gap between the domain concepts and the program is (relatively) small.

The goal of this lecture is to enable you to systematically carry out small(er) software projects that produce well-designed software.

- **Domain Modeling** is useful to understand the ideas and concepts of the domain and their interrelationships
- A **Domain Model** is usually created at the beginning of a project and is a basis for the design model
- A **Domain Model** is created using a subset of the UML class diagram notation

- The goal of this lecture is to enable you to systematically carry out small(er) commercial or open-source projects.



Requirements Management
Domain Modeling