

CS6110

OBJECT ORIENTED ANALYSIS & DESIGN

Elaboration – Domain Model

Dr.S.Neelavathy Pari

Assistant Professor (Sr. Grade)
Department of Computer Technology
Anna University, MIT Campus

- Elaboration
- Domain Model – Finding Conceptual Classes
- Description Classes
- Association and Attributes

What is an elaboration?

Elaboration is used to build the
core architecture,
resolve the high-risk elements,
define most requirements,
& estimate the overall schedule and resources.

Tasks performed in elaboration

1. The core, risky software architecture is programmed and tested
2. The majority of requirements are discovered and stabilized
3. The major risks are mitigated or retired

key ideas & best practices that will manifest in elaboration

1. Do a short time boxed risk-driven iterations
2. Start programming early
3. Adaptively design, implement, and test the core and risky parts of the architecture
4. Test early, often, realistically
5. Adapt based on feedback from tests, users, developers
6. Write most of the use cases and other requirements in detail, through a series of workshops, once per elaboration iteration

- Elaboration is not a Design Phase
- During this phase, concentrate on code and design and production quality portion of the final system.
- It is called executable architecture of architectural baseline

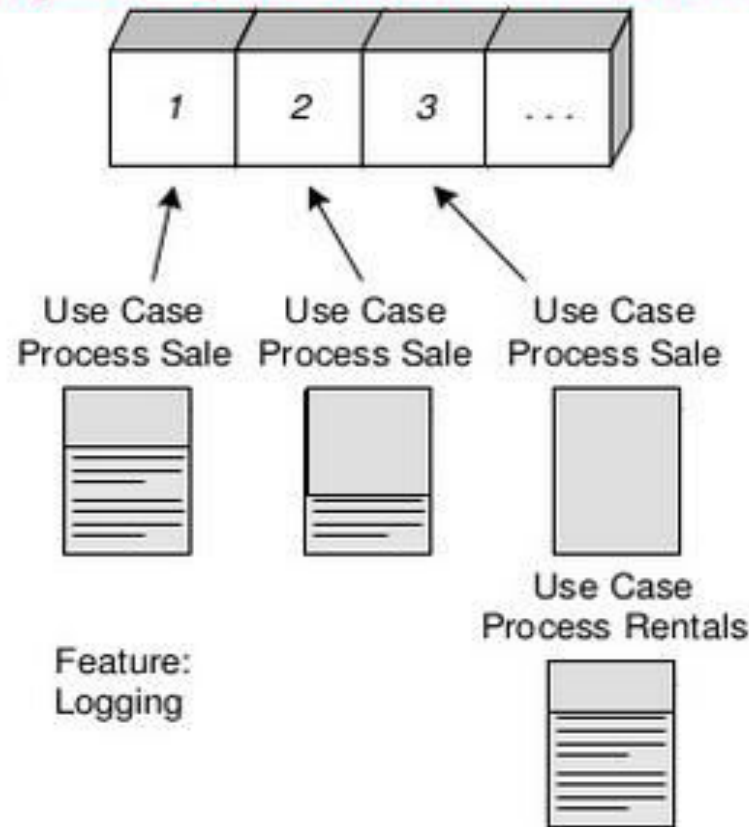
Artifacts

Artifact	Description
Domain Model	Visual representation of the domain model
Design Model	Logical Design Diagrams
Software Architecture Document	Summary of key architectural issues and their resolution
Data Model	Database schemas, mapping strategies between object and non-object representation
Use-Case Storyboards, UI Prototypes	User interface description, usability models...

Iterations in Elaboration

- In Iterative Development
 - Don't Implement All the Requirements at once
 - **Incremental Development for the Same Use Case Across Iterations**

Use case
implementation
may be spread
across iterations



A use case or feature is often too complex to complete in one short iteration.

Therefore, different parts or scenarios must be allocated to different iterations.

Iteration 1 of Elaboration Phase

- Emphasizes a range of fundamental and core OOA/D skills used in building object systems,
 - Such as assigning responsibilities to objects.
- Iteration 1 is NOT architecture-centric and risk-driven

Iteration 1 of NextGen POS Application

- **Requirements for iteration 1 of the POS application**

- Implement a basic, key scenario of the Process Sale use case:

Entering items and receiving a cash payment.

- **Implement a Start Up use case** as necessary

To support the initialization needs of the iteration.

- **Nothing fancy or complex is handled,**

A simple happy path scenario, and the design and implementation to support it.

- No collaboration with external services, such as a tax calculator or product DB.

- **No complex pricing rules are applied.**

- **Design and implementation of supporting UI, DB, and so forth, would also be done**

Planning the Next Iteration

- Organize requirements and iterations by risk, coverage, and criticality
 - **Risk** includes both technical complexity and other factors, such as uncertainty of effort or usability.
 - **Coverage** implies that all major parts of the system are at least touched on in early iterations perhaps a "wide and shallow" implementation across many components.
 - **Criticality** refers to functions the client considers of high business value.

Planning the Next Iteration

- Use cases or use case scenarios are ranked for implementation.
 - Early iterations implement high ranking scenarios.

Rank	Requirement (Use Case or Feature)	Comment
High	Process Sale Logging ...	Scores high on all rankings. Pervasive. Hard to add late. ...
Medium	Maintain Users ...	Affects security subdomain. ...
Low

DOMAIN MODEL

- **What is domain model?**

A domain model is a visual representation of conceptual classes or real-world objects in a domain of interest.

They have also been called conceptual models, domain object models, and analysis object models.

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

Steps involved in creating a domain model

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

Conceptual Classes : The conceptual class is defined as an idea, thing, or object. More formally, a conceptual class may be considered in terms of its symbol, intension, and extension .

Description Class A description class is defined as information that describes something else. For example, a Product Description that records the price, picture, and text description of an Item.

*Good designs require
deep application
knowledge.”*

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**.

Visualizing Domain Models

Case Study : CMS

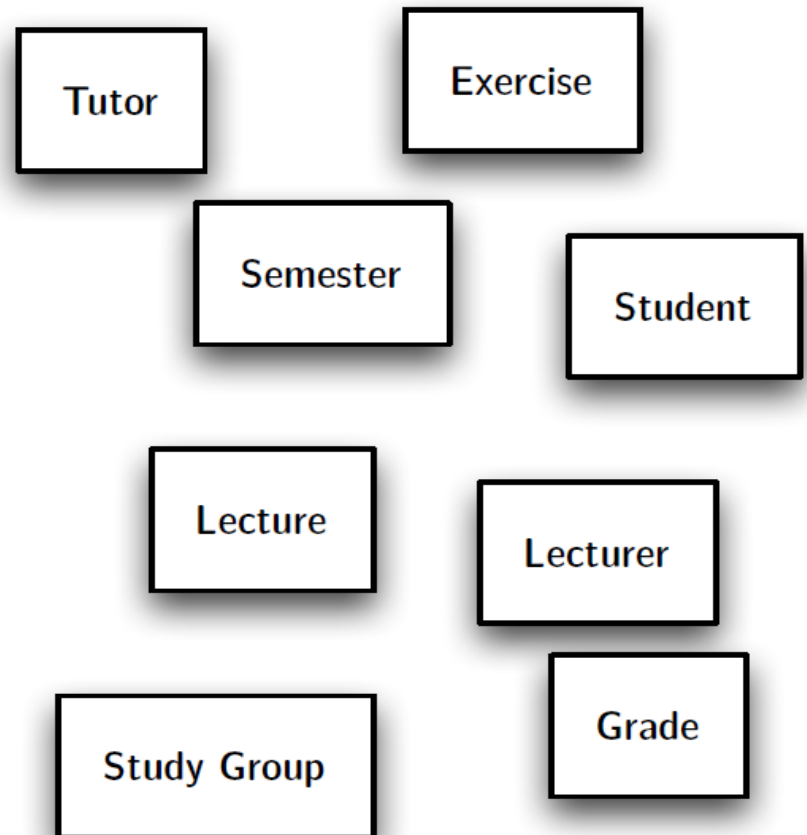
Statements about a **Course Management System**

- During a semester a Professor handles one or more lectures.
- Sometimes the professor is on leave to focus on doing research, in this case (s)he does not give a lecture
- A student usually attends one or more Course
- 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**



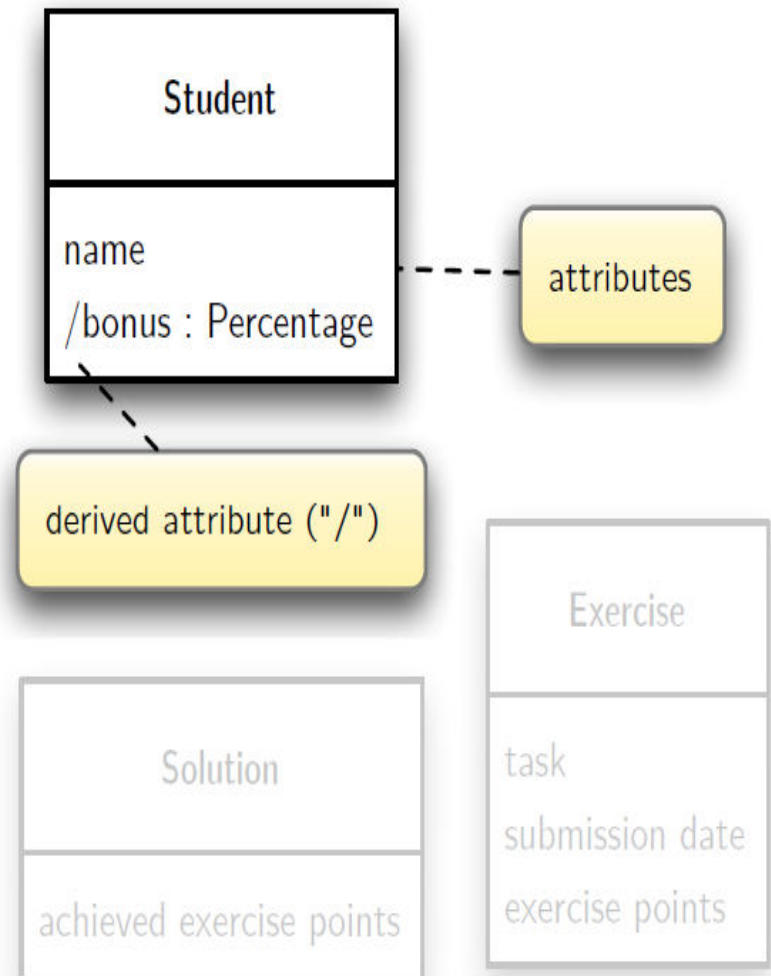
Attributes

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.

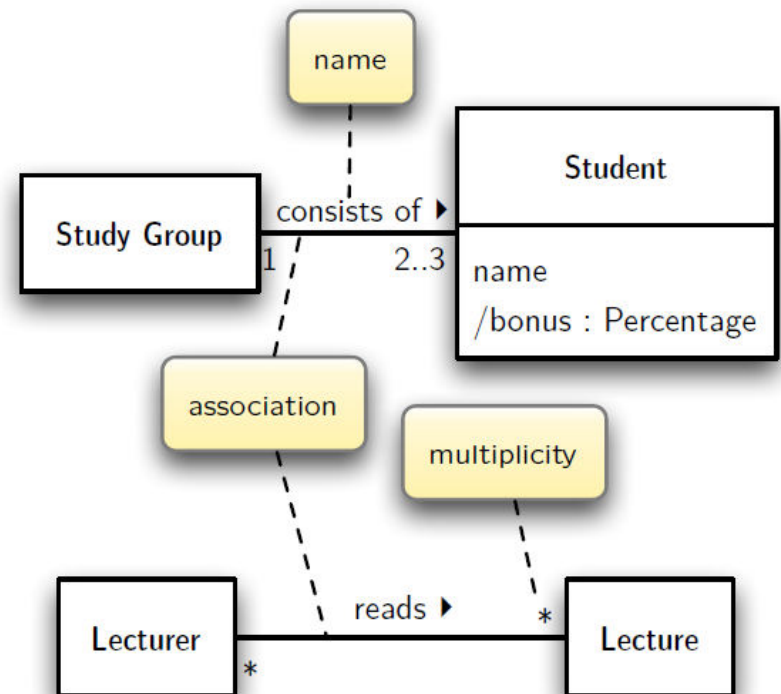


Association

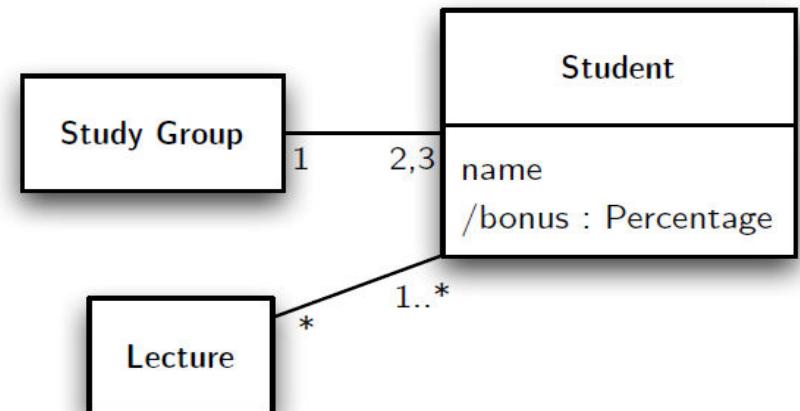
- **An association is a relationship between classes.**
- The ends of an association are called roles.
- Roles optionally have a multiplicity, name and navigability.

Association

- 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
- ...

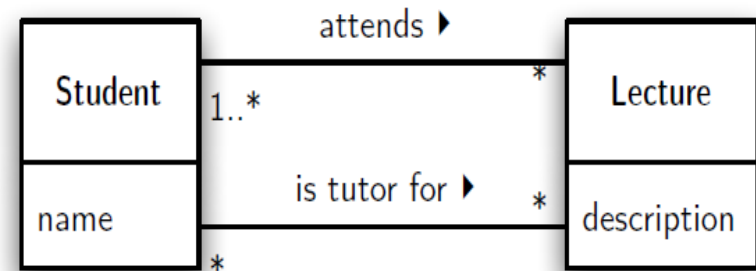


- The **multiplicity** defines how many instances of a class A can be associated with one instance of a class B at any particular moment
- (e.g., * \triangleq zero or more; 1..10 between 1 and 10; 1,2 one or two)
- A student usually attends **one or more lectures**, unless (s)he has something better to do
- A study group consists of **two to three** students...
- ...

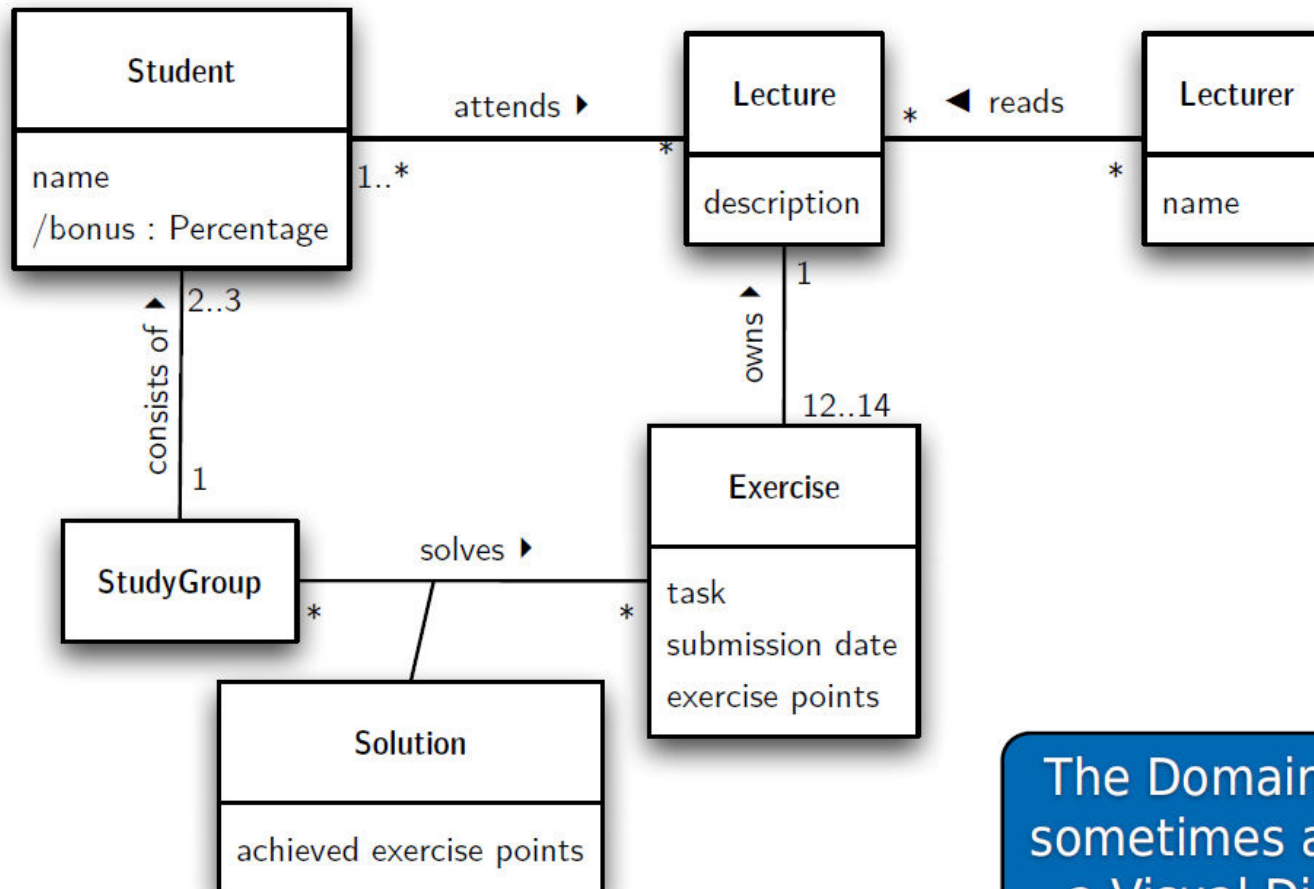


Two classes can have multiple associations

- 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.

- **Why domain model is called as a "Visual Dictionary"?**

The information domain model illustrates could alternatively have been expressed in plain text.

But it's easy to understand the terms and especially their relationships in a visual language, since our brains are good at understanding visual elements and line connections.

Therefore, the domain model is a visual dictionary of the noteworthy abstractions, domain vocabulary, and information content of the domain.

Domain Modeling

Domain Model For the POS System

How to create the domain model?

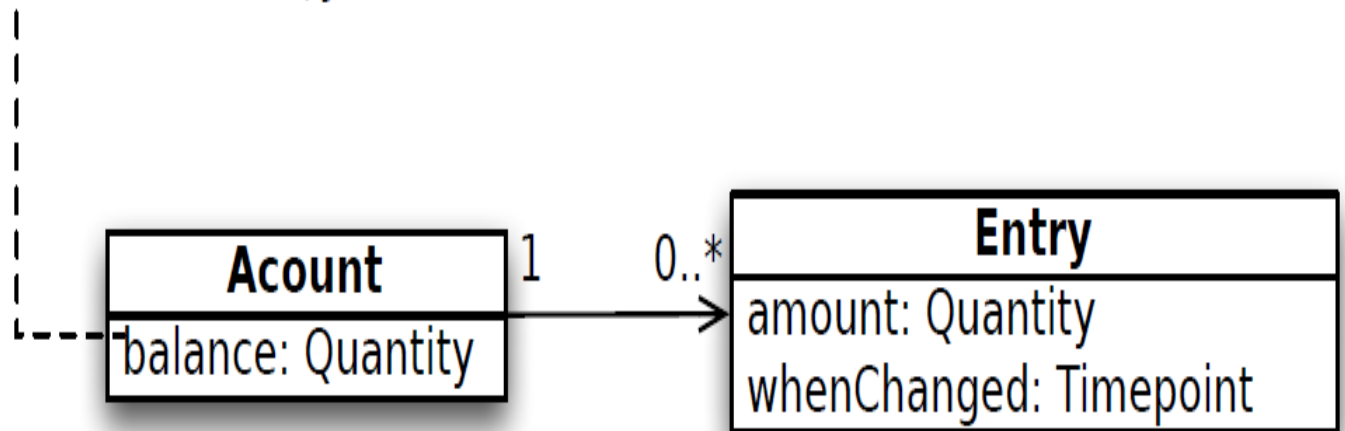
1. Find the conceptual classes Strategies:
 - a. Reuse or modify an existing model*
 - b. Use a category list*
 - c. Identify noun phrases*
2. Draw them as classes in a UML class diagram
3. Add associations and attributes

Note : Use the domain vocabulary; e.g. a model for a library should use names like “*Borrower*” instead of *customer*.

- a) To find the conceptual classes reuse or modify an existing model.

Example: existing model for bank accounts and associated entries

`{balance = sum(entries.amount)}`



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
...	...

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

Note : “Use Cases” are also an excellent source for identifying conceptual classes.

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
- 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.

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
- 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.

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



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

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...

Should Receipt be in 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

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

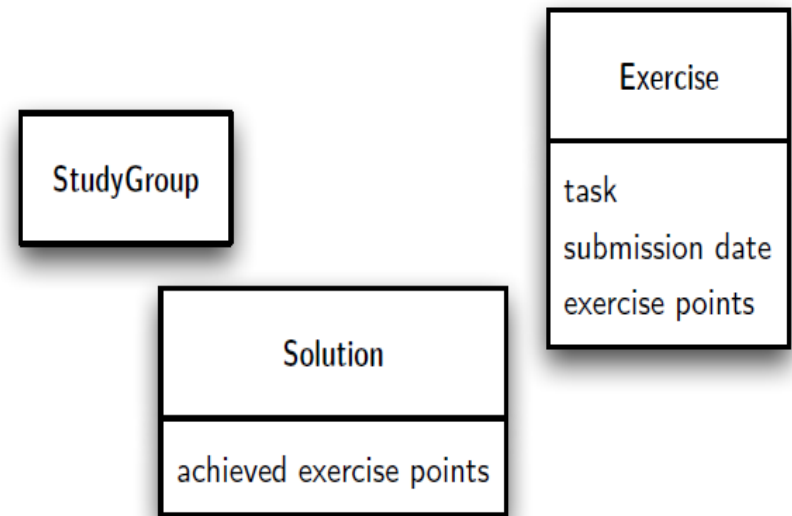
Guidelines when to include a description class into the domain model

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

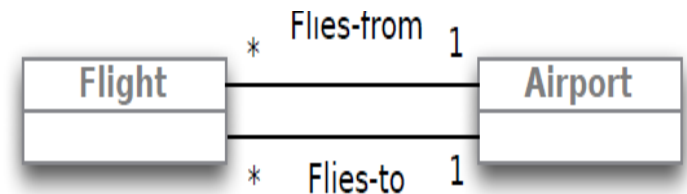
When should I model something as an attribute or a class?

- 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.



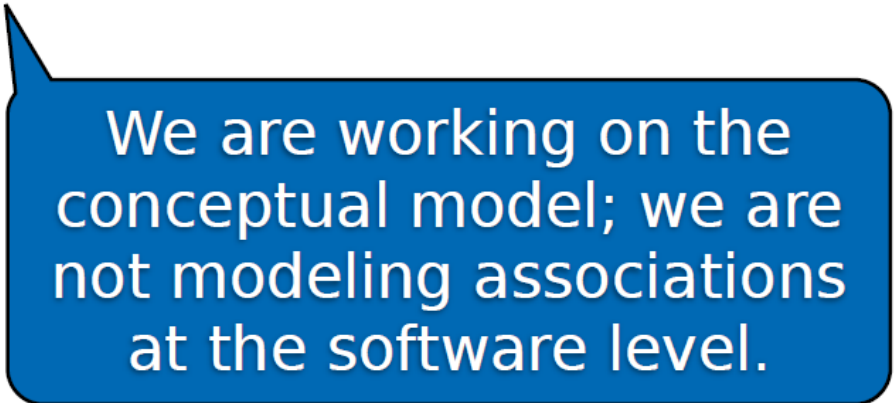
When should I model something as an attribute or a class?
Let's assume that we develop an airline reservation system

- 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 SalesLine Item 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- Verb Phrase-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!).

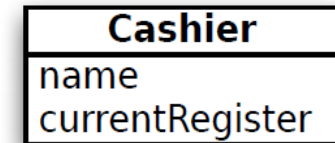
- Very common data types include: Boolean, Date, Number, Character, String, Address, Color, Phone Number,...
- **Consider modeling quantities as classes to be able to associate units**
e.g. the data type of the amount attribute of a payment should indicate the currency

Cashier
name
currentRegister



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

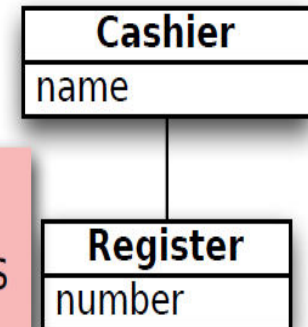
- Very common data types include: Boolean, Date, Number, Character, String, Address, Color, Phone Number,...
- Consider modeling quantities as classes to be able to associate units
e.g. the data type of the amount attribute of a payment should indicate the currency



recommended:

it

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



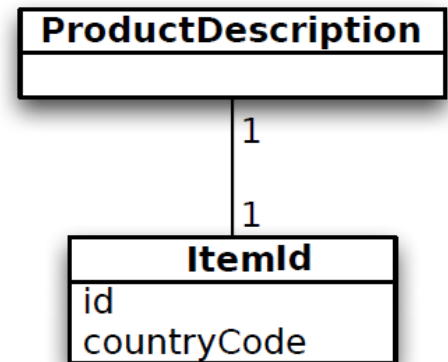
Consider defining a new data type class for something that is initially considered a string

- 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:

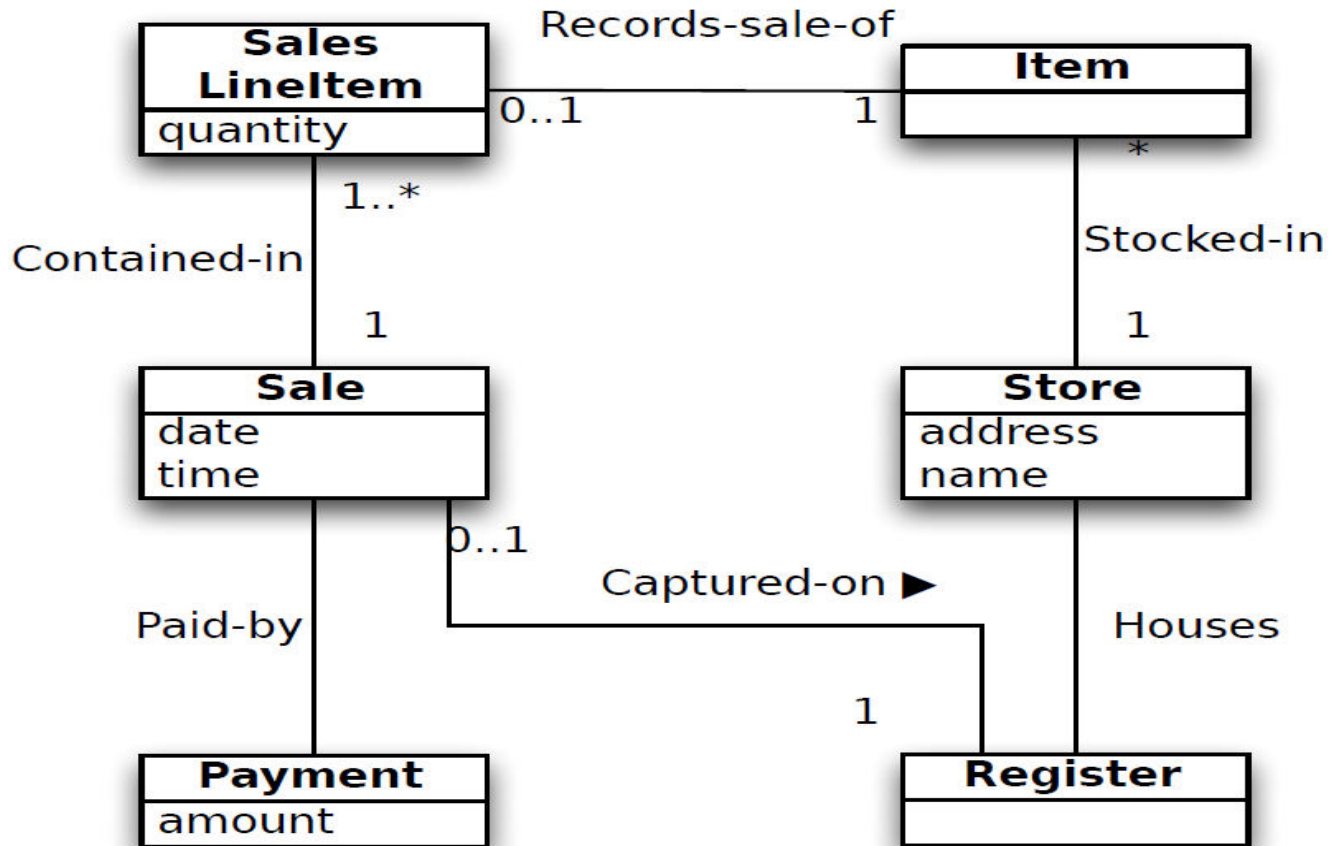


Domain Model For the POS System

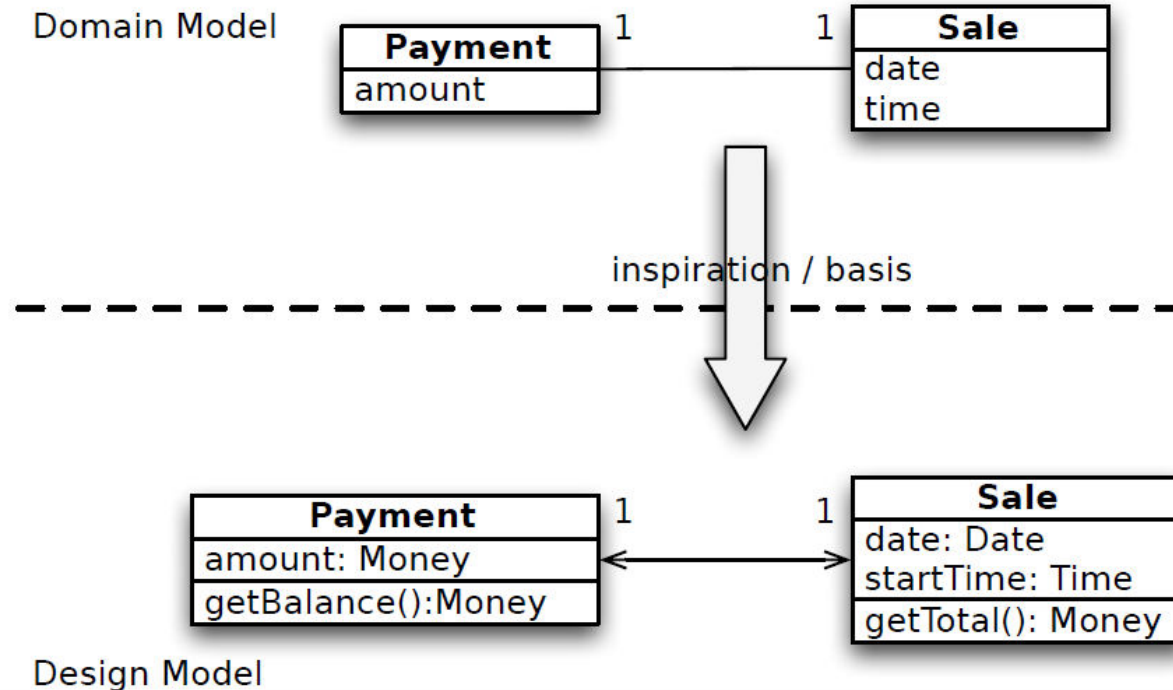


Which are noteworthy domain concepts /
domain objects ?

Domain Model For the POS System



Domain Model and Domain Modeling



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.

Domain Model and Domain Modeling

Summary

- 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

model-driven development

A *domain model* is called *conceptual model* in database modeling, while a *design model* is called *logical model*.

- (solution-independent) ***domain models*** resulting from domain/requirements engineering in the system analysis, or inception, phase of a development project;
- (platform-independent) ***design models*** resulting from the system design activities in the elaboration phase and typically based on a *domain model*;
- (platform-specific) ***implementation models***, which may be (e.g., JavaScript or Java EE) ***class models***, SQL ***table models*** or other types of ***data models*** derived from an information design model.