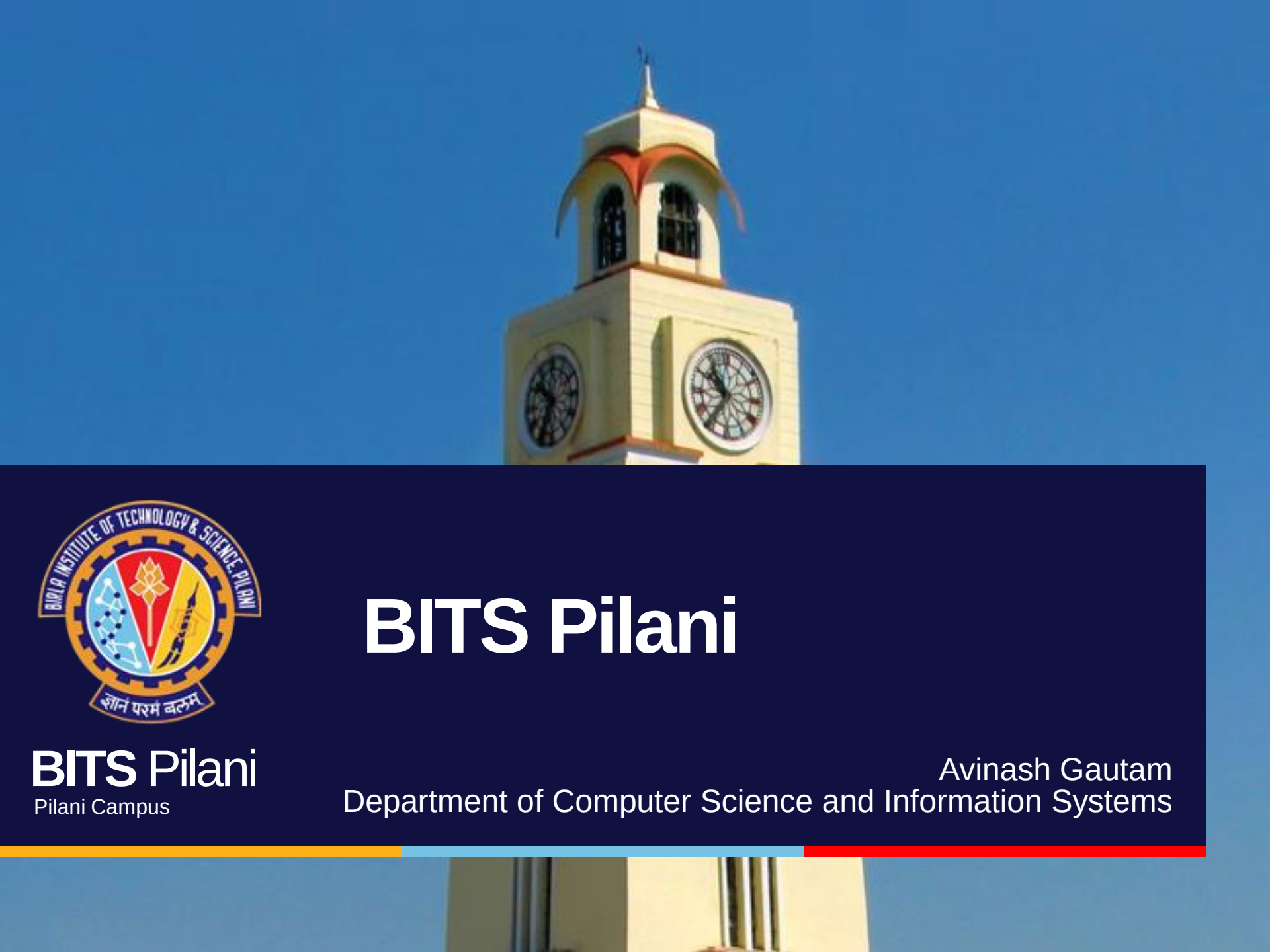


BITS Pilani

BITS Pilani
Pilani Campus

Avinash Gautam
Department of Computer Science and Information Systems

Drawing System Sequence Diagrams

System Behavior

- Objective
  - identify system events and system operations
  - create system sequence diagrams for use cases

System Behaviour and UML Sequence Diagrams

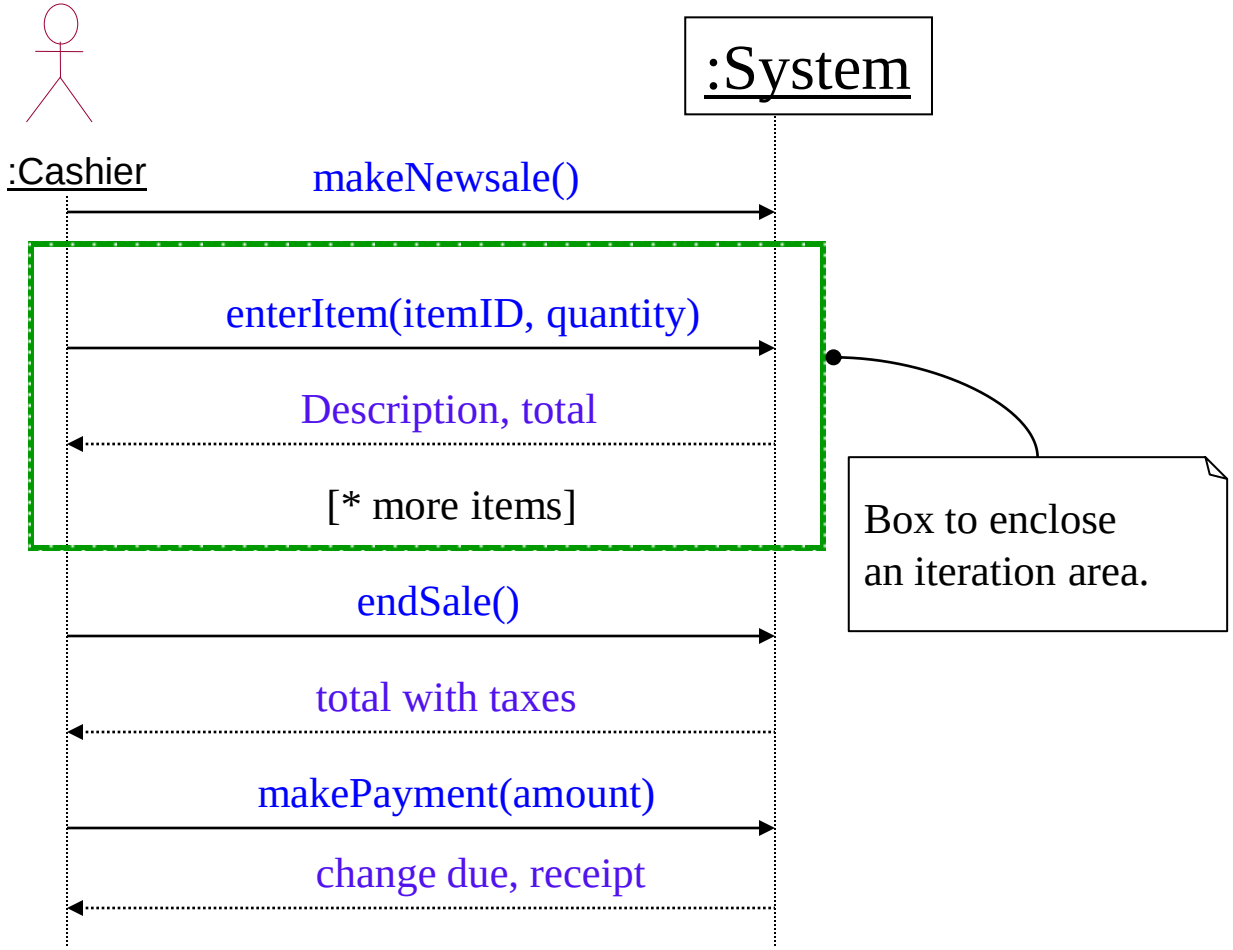
- It is useful to investigate and define the behaviour of the software as a “black box”.
- System behaviour is a description of *what the system does* (without an explanation of how it does it).
- Use cases describe how external actors interact with the software system. During this interaction, an actor generates events.
- A request event initiates an operation upon the system.

System Behaviour and System Sequence Diagrams (SSDs)

- A system sequence diagram is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and possible inter-system events.

- All systems are treated as a black box; the diagram places emphasis on events that cross the system boundary from actors to systems.

System sequence diagram - Example: *Process Sale* scenario

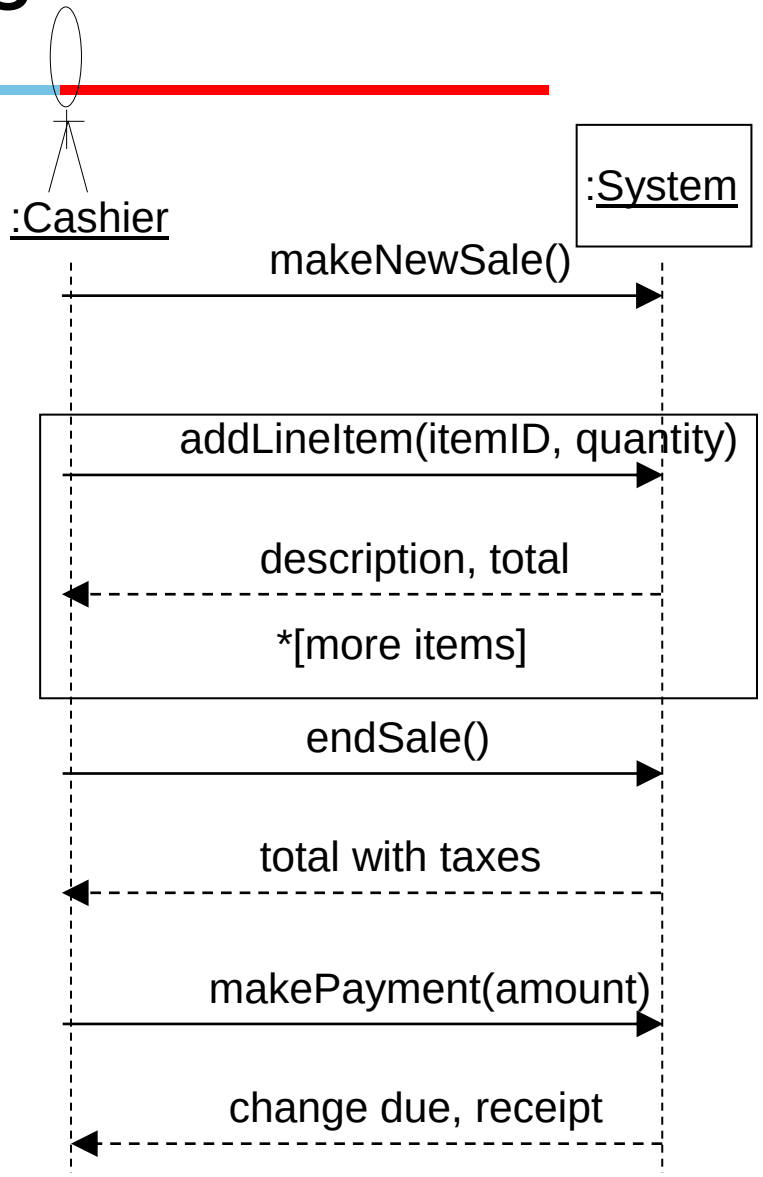


SSD and Use Cases

Simple cash-only Process Sale Scenario

1. Customer arrives at a POS checkout with goods to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item, and presents item description, price and running total.
   cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.

…



System sequence diagrams [1]

- A system sequence diagram illustrates *events* from *actors* to *systems* and the external response of the system

- SSD drawing occurs during the analysis phase of a development cycle; dependent on the creation of the use cases and identification of concepts.

- UML notation - Sequence Diagram *not* System Sequence Diagram.

System sequence diagrams [2]

- One diagram depicts one scenario. This is the *main success* scenario.
- Frequent or complex alternate scenarios could also be illustrated.
- A system is treated as a *black box*.
- SSD is often accompanied by a textual description of the scenario to the left of the diagram.

System sequence diagram [3]

- Identify the system boundary…what is inside and what is outside.
- System event: An external event that directly stimulates the (software) system.
- Events are initiated by actors.
- Name an event at the level of *intent* and *not* using their physical input medium or interface widgets.
  - *enterItem()* is better than scan().
- Keep the system response at an abstract level.
  - *description, total* is preferred over *display description and total on the POS screen.*

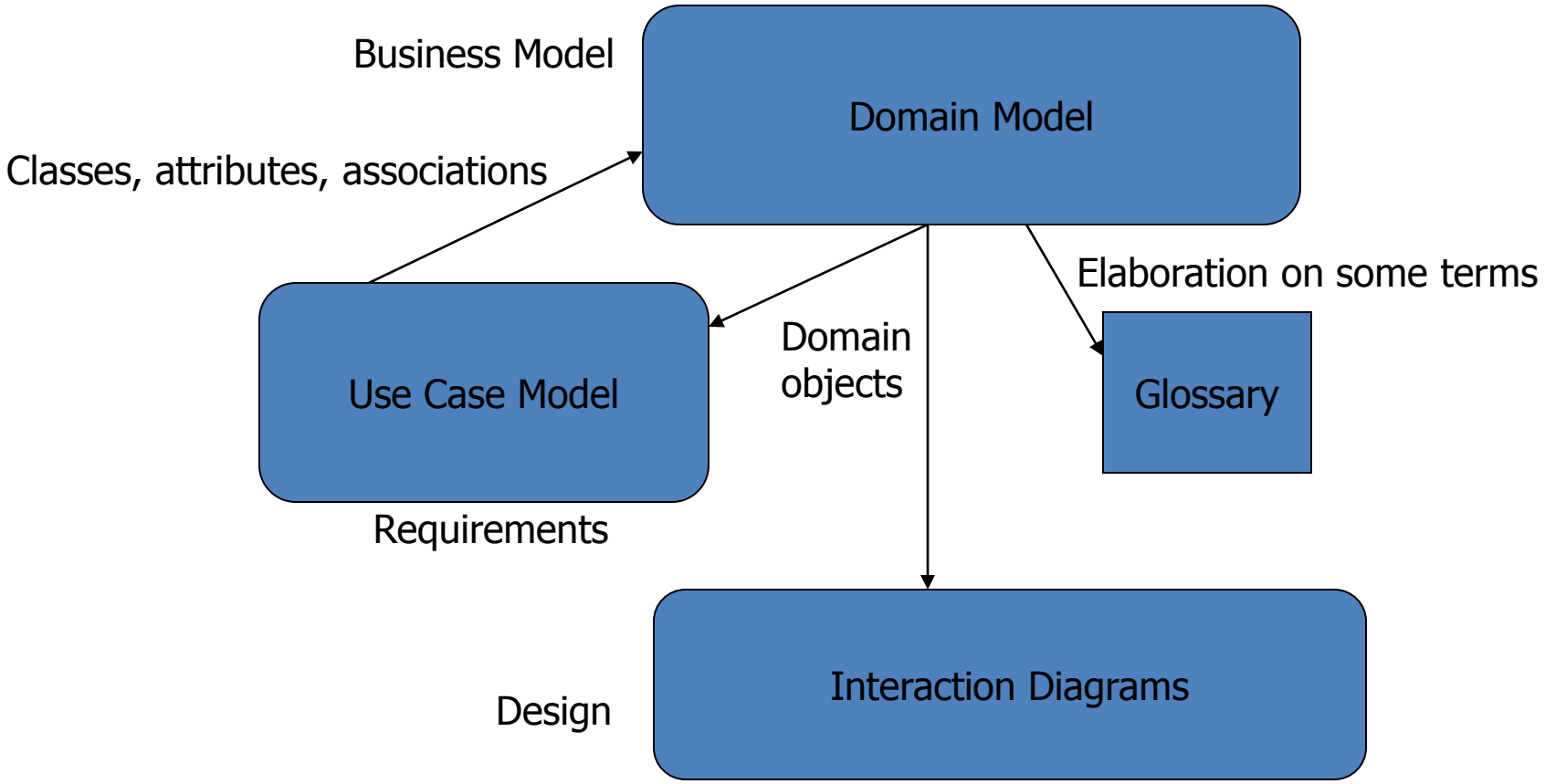
Domain Model

Domain model: What is it?

innovate achieve lead

- Illustrates meaningful *concepts* in the *problem domain.*
- Usually expressed in the form of *static* diagrams (in Rational Rose this implies a high-level class diagram).
- Is a representation of *real-world* things; not software components (of the system under development).
- No operations are defined or specified in the domain model.
- The model shows concepts, associations between concepts, and attributes of concepts.
- Serves as a source of software objects.

Domain Model Relationships



A Domain Model is the most important OO artifact

- Its development entails identifying a rich set of conceptual classes, and is at the heart of object oriented analysis.
- It is a visual representation of the decomposition of a domain into individual conceptual classes or objects.
- It is a visual dictionary of noteworthy abstractions.

Decomposition

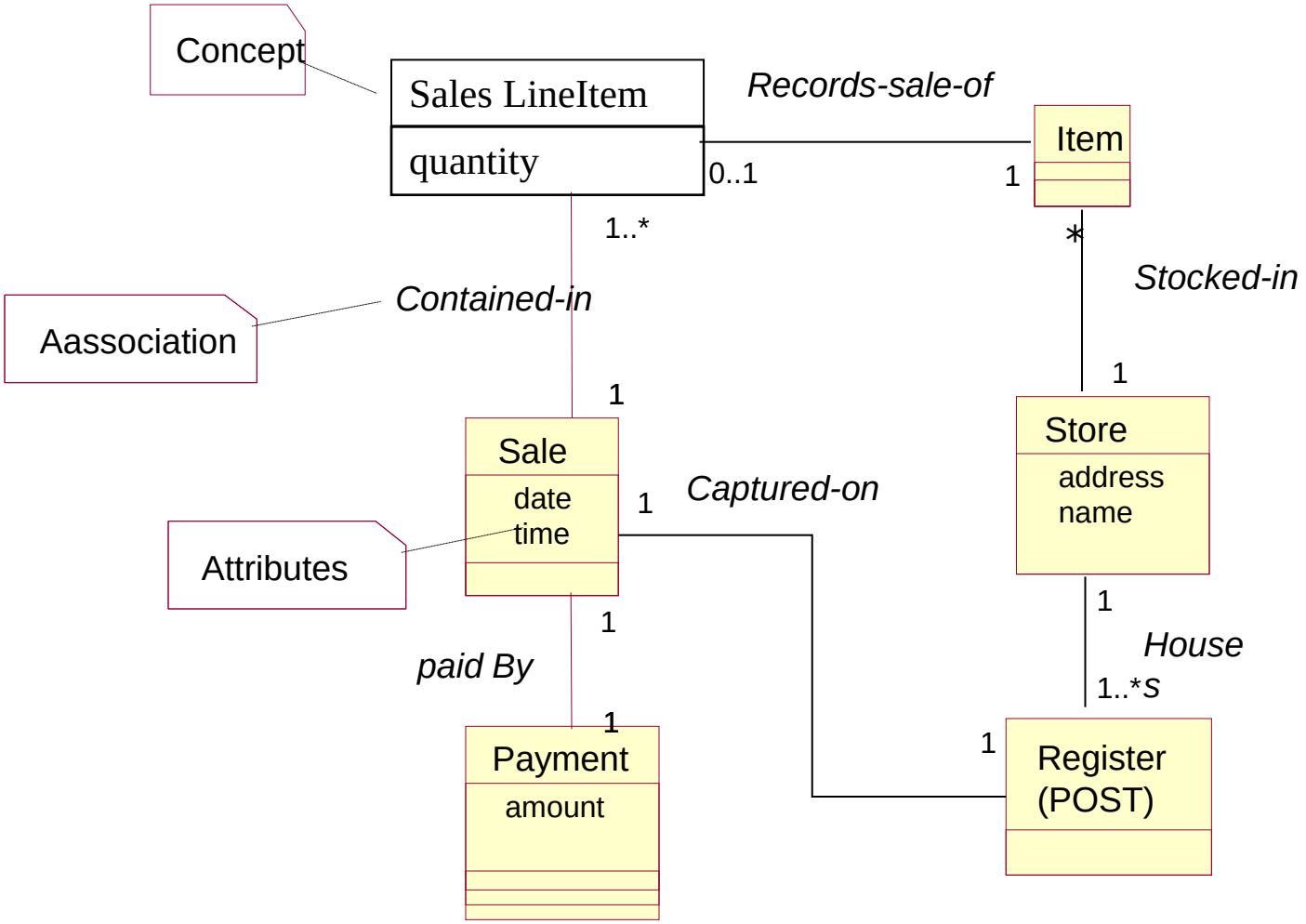
- A central distinction between Object-oriented analysis and structured analysis is the division by objects rather than by functions during decomposition.
- During each iteration, only objects in current scenarios are considered for addition to the domain model.

Domain model: How to construct?

innovate achieve lead

- Objectives
  - identify concepts related to current development cycle requirements
- create initial conceptual model
- Identify attributes
- add specification concepts

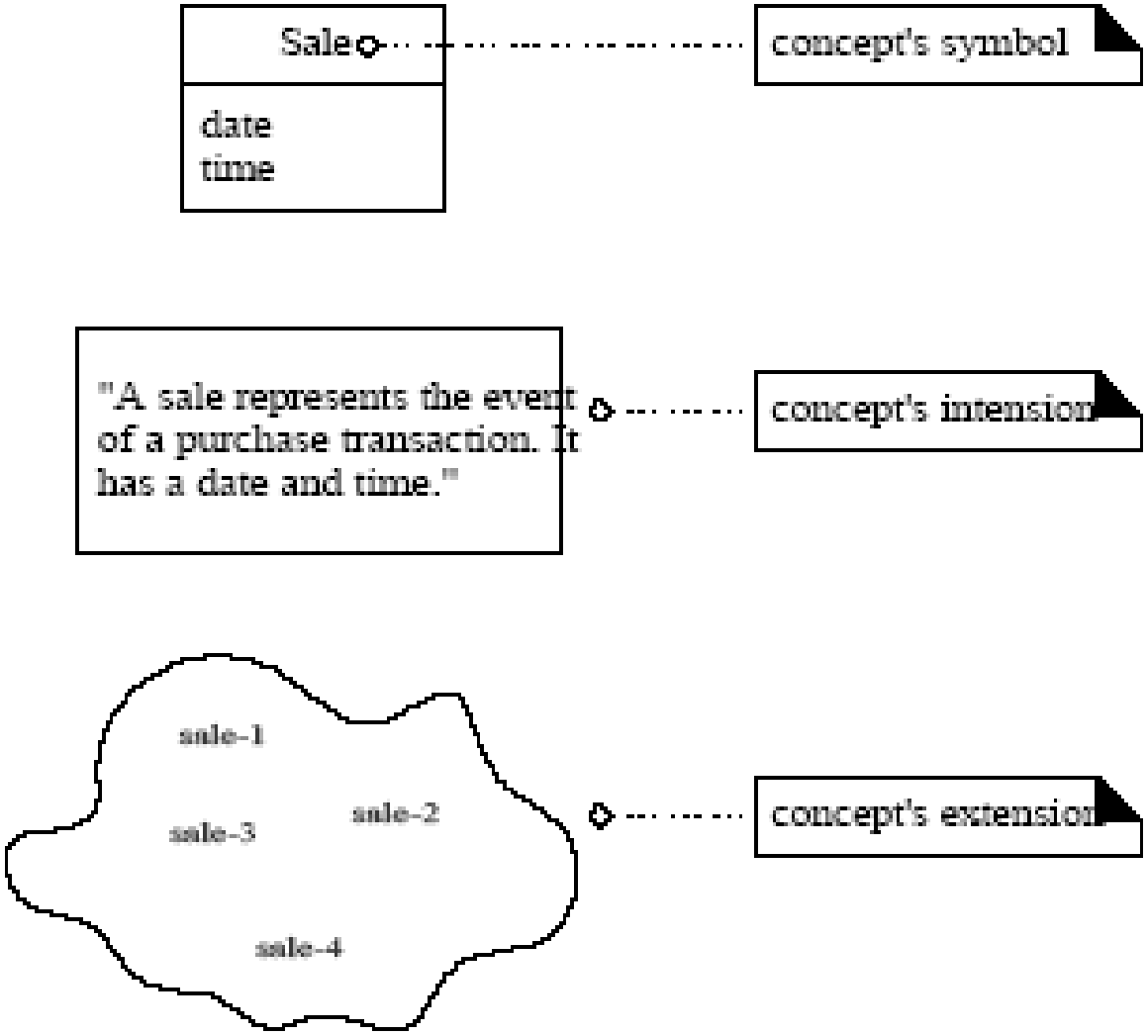
Partial domain model of POST



Think of Conceptual Classes in terms of

- Symbols – words or images
- Intensions – its definition
- Extensions – the set of examples to which it applies
- Symbols and Intensions are the practical considerations when creating a domain model.

Symbol - Intension - Extension



Conceptual Class Identification

- It is better to overspecify a domain with lots of fine-grained conceptual classes than it is to underspecify it.
- Discover classes up front rather than later.
- Unlike data modeling, it is valid to include concepts for which there are no attributes, or which have a purely behavioral role rather than an informational role.

Finding concepts: Use noun phrases

- Finding concepts using Noun Phrase identification in the textual description of the domain.
- Finding concepts using the concept category list (refer to page p140 in Larman)
  - Physical objects: register, airplane, blood pressure monitor
  - Places: airport, hospital
  - Catalogs: Product Catalog

Identify Conceptual Classes by Category List:

Common Candidates for classes include:

Tangible objects, Descriptions, Roles, Places, Transactions, Containers, Systems, Abstract nouns, Rules, Organizations, Events, Processes, Written Materials, Catalogs, Records, Financial Instruments and Services

Finding concepts: refer to use cases

- Examine use case descriptions.
- Example: *Process Sale* use case:
  - Main success scenario:
    - *Customer* arrives at a *POS* checkout counter.
    - *Cashier* starts a new *sale*.
    - *Cashier* enters an *item* ID.
    - System records *sale line item*. It then presents a description of the *item*, its price, and a running total.
- Possible source of confusion: Is it an *attribute* or a *concept*?
    - If X is not a number or a text then it probably is a *conceptual class*.

Finding concepts: Examples

- Are these concepts or attributes?
  - Store
  - Flight
  - Price
- Use terms familiar to those in the problem domain.
  - POST or register?
- Concepts from “Unreal” world ?
- Example - Telecommunications
    - Message
    - Connection
    - Port

What about Sales Receipt?

innovate achieve lead

- Should it be included in the model?
  - Its common in the real world system
  - It's a sales report. Reports not explicitly stated in the use cases have little value in this model.
  - However, when a Customer wishes to return an item, it is an important object in the domain.
  - Since this develop cycle doesn't include the Return Items use case, we'll leave it out of this CM

Concepts in POST domain

- POST
- Item
- Sale
- Store
- Payment
- SalesLineItem
- Product Specification
- ProductCatalog
- Customer
- Cashier
- Manager

Specification Concepts

- When are they needed?
- Add a specification concept when:
- deleting instances of things they describe (for example, Item) results in a loss of information that needs to be maintained.
- it reduces redundant or duplicated information

Specification Example

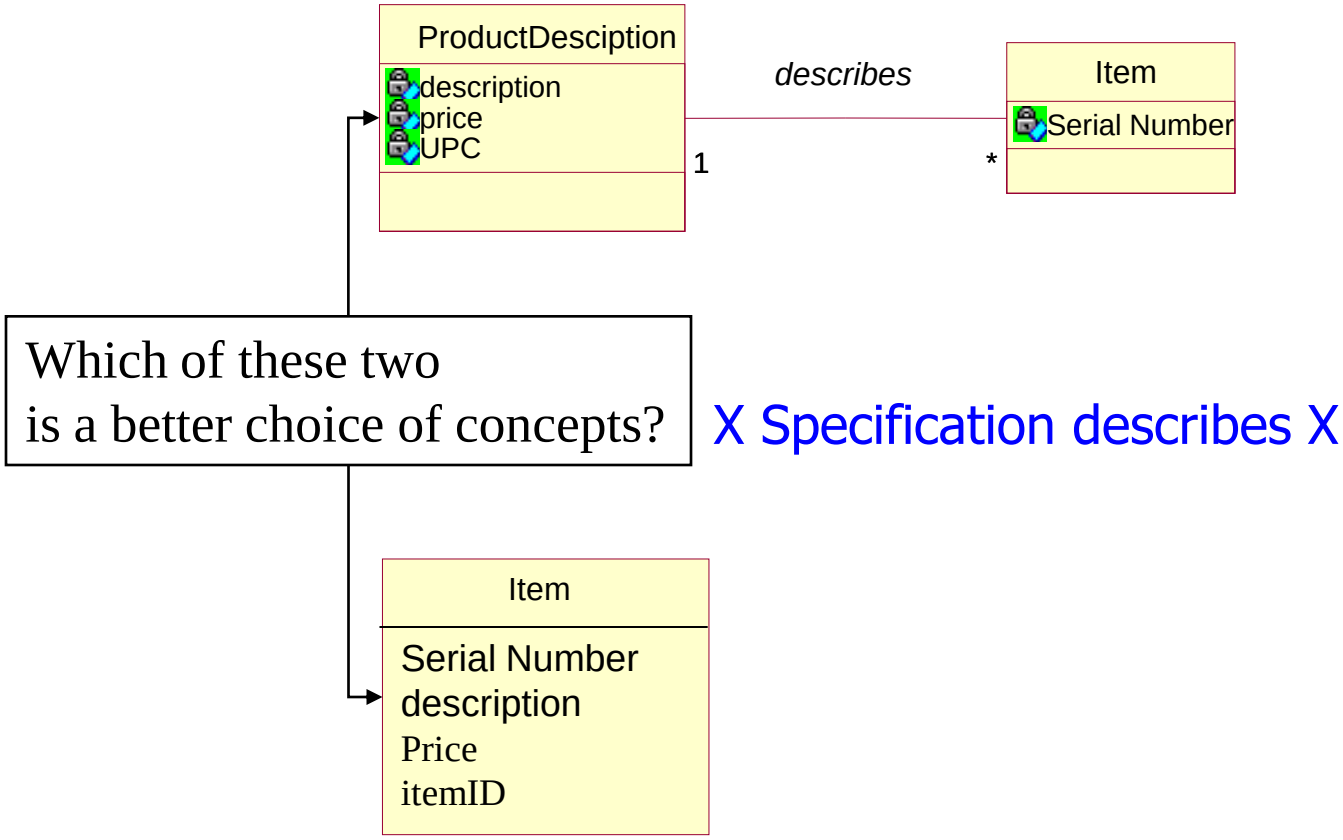
- Assume that
  - an *item* instance represents a physical *item* in the store; it has a serial number
  - an *item* has a description, price and itemID which are not recorded anywhere else.
  - every time a real physical *item* is sold, a corresponding software instance of *item* is deleted from the database

- With these assumptions, what happens when the store sells out of a specific *item* like "burgers"? How does one find out how much does the "burger" cost ?
- Notice that the price is stored with the inventoried instances

Specification Example – Contd.

- Also notice the data is duplicated many times with each instance of the item.
- This example illustrates the need for a concept of objects that are specifications or descriptions of other things (often called a *Proxy or Surrogate*)
- Description or specification objects are strongly related to the things they describe.

Specification - Example



Conceptual Models - Association

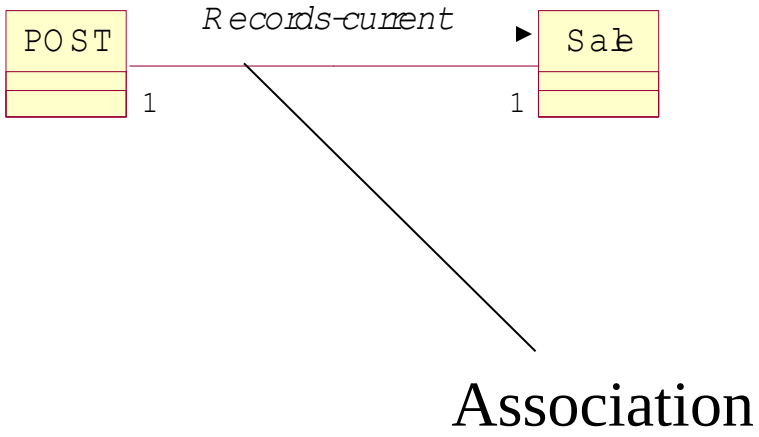
- Objective
  - Identify associations within a conceptual model
  - Distinguish between need-to-know associations from comprehension-only associations

Associations

- Associations imply knowledge of a relationship that needs to be preserved over time
  - could be milliseconds or years!
  - such associations are *need-to-know* assocs.
- Associations derived from the Common Associations List (Pg 155-156, Larman).
- Represented as a solid line between objects
  - the association is inherently bi-directional
  - may contain a *cardinality* or *multiplicity* value
  - optionally contains an arrow for easy reading

Associations

- Association - a relationship between concepts that indicates some meaningful and interesting connection



Finding Associations

- High priority associations
  - A is a physical or logical part of B
  - A is physically or logically contained in/on B
  - A is recorded in B

- Other associations
  - *A* uses or manages or controls *B (Pilot -airplane)*
  - *A* owns *B (Airline -airplane)*

Association Guidelines

- Focus on those associations for which knowledge of the relationship needs to be preserved for some duration (need-to-know associations)
- More important to identify concepts than associations
- Too many associations tend to confuse the conceptual model
- Avoid showing redundant or derivable associations

Roles in Associations

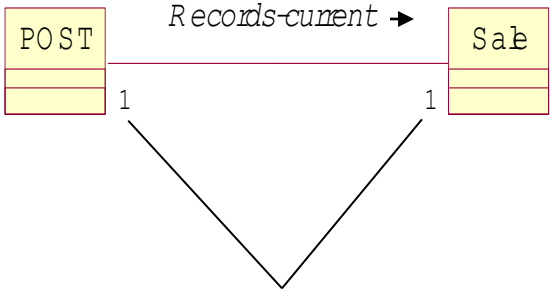
- Each of the two ends of an association is called a role. Roles have
  - name
  - multiplicity expression
  - navigability

Naming Associations

- Association names should start with a capital letter (same as concepts, objects)
- Noun phrases help identify objects/concepts
- Verb phrases help identify associations
- Use hyphens to separate words when a phrase is used to name something

Multiplicity

- **Multiplicity** defines how many instances of type A can be associated with one instance of type B at a particular moment in time



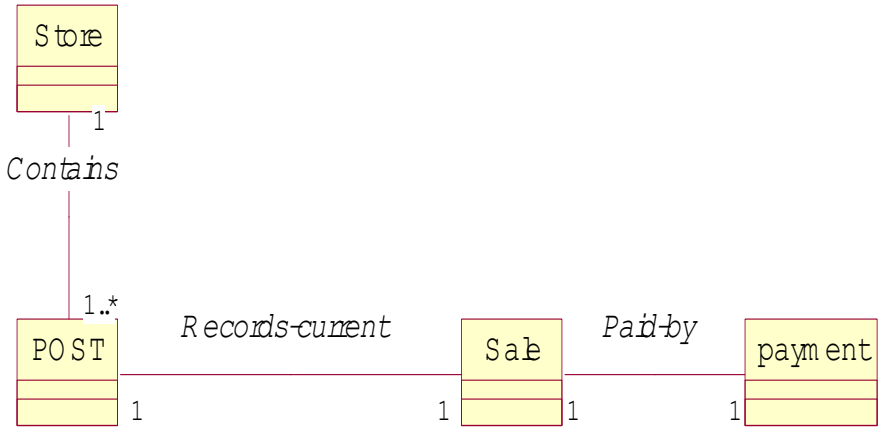
Multiplicity of the role

Association - Multiplicity

- Multiplicity: indicates the number of objects of one class that may be related to a single object of an associated class

- Can be one of the following types
  - 1 to 1, 1 to 0..*, 1 to 1..*, 1 to n, 1 to 1..n

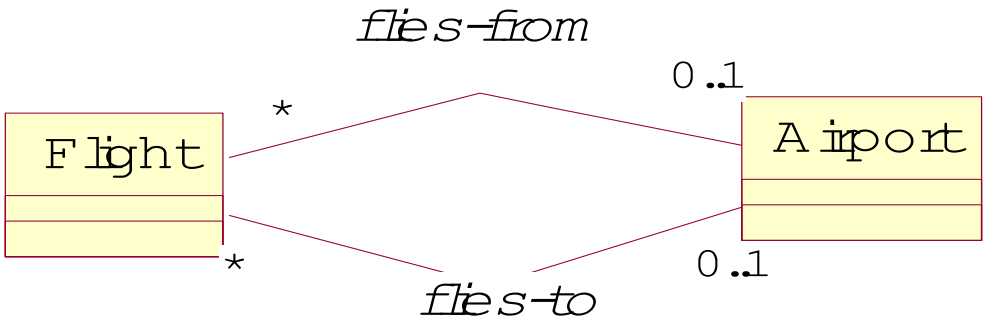
Associations - Contd.



Associations are generally read left to right, top to bottom

Associations - Contd.

Multiple associations between two types



During analysis phase, an association is *not* a statement about data flows, instance variables, or object connections in the software solution.

Regarding Associations

- Associations are real-world relationships, which may or may not be implemented in the final system.
- Conversely, we may discover associations that need to be implemented but were missed in analysis. The conceptual model should be updated!
- Usually, associations are implemented by placing an instance variable which “points to” (references) an instance of the associated class.
- Only add associations (or concepts) to the model that aid in the *comprehension* of the system by others.
- Remember, conceptual models are communication tools, used to express concepts in the problem domain.

Conceptual Model - Attributes

innovate achieve lead

- Objectives
  - Identify attributes in a conceptual model
  - Distinguish between correct and incorrect attributes

Attributes [1]

- *Attribute - is a logical data value of an object.*

  It is a named property of a class describing values held by each object of the class

- Attribute Type: A specification of the external behavior and/or the implementation of the attribute

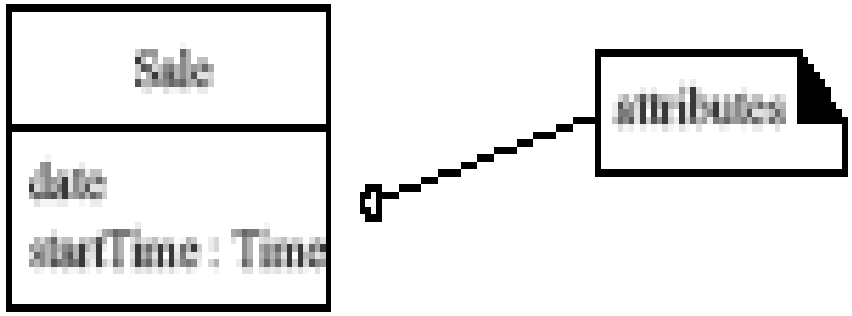
- Include the following attributes in a domain model
  - those for which the requirements suggest or imply a need to remember information

- For example, a *sale* receipt normally includes a date and time attribute

Attribute Name:attribute Type

UML Attribute Notation

- Attributes reside in the 2nd compartment of a concept box
- Format is name: type
- Attributes should be pure data values

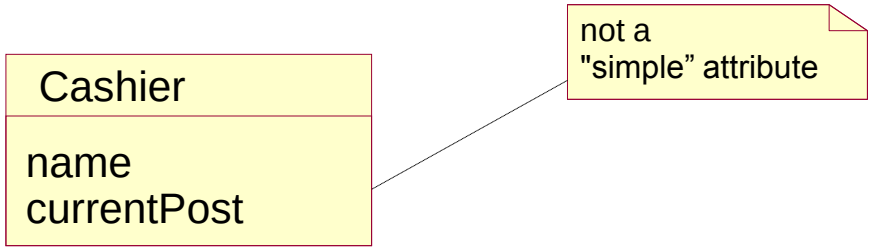


Attributes [2]

- Attributes in a conceptual model should preferably be simple attributes or pure data values
- Common simple attribute types include
  - boolean, date, number, string, time

Attributes: Examples

worse



better

Relate two items with associations, not attributes, in conceptual model.

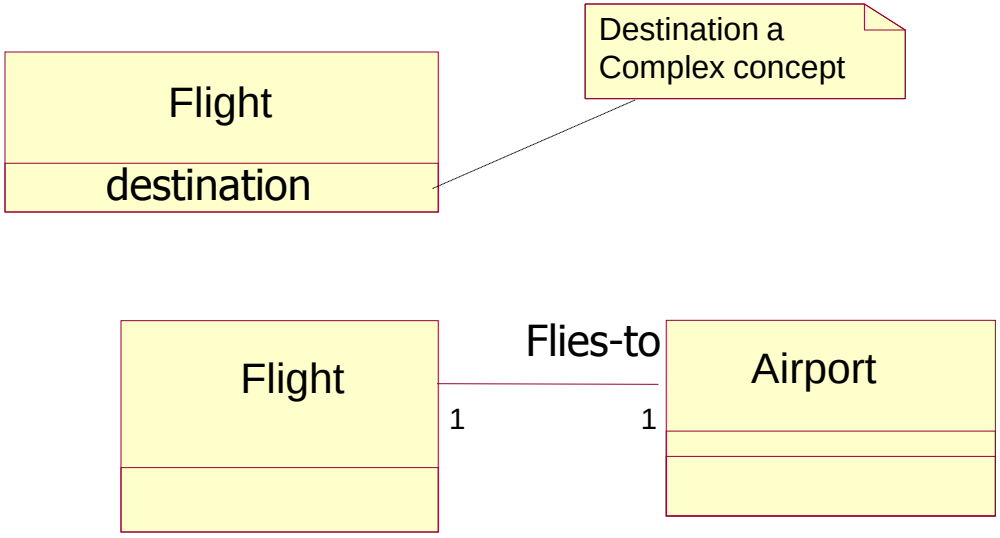
Complex Attributes

- Pure data values - expressed as attributes; they do not illustrate specific behaviors;
  - Example - Phone number
  - A Person can have many Phone numbers

- Non-primitive attribute types
  - represent attributes as non-primitive types (concepts or objects) if
    - it is composed of separate sections (name of a person)
    - there are operations associated with it such as validation
    - it is a quantity with a unit (payment has a unit of currency)

Complex Attributes

- It is desirable to show non-primitive attributes as concepts in a conceptual model



THANK YOU