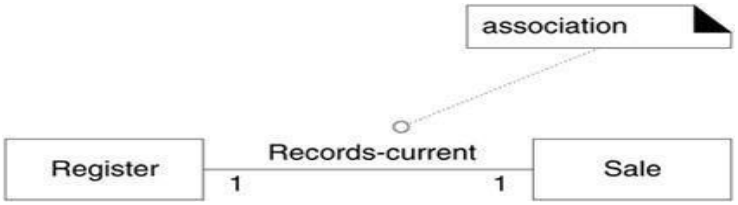
ASSOCIATIONS

An **association** is a relationship between classes (more precisely, instances of those classes) that indicates some meaningful and interesting connection (see Figure)



In the UML, associations are defined as "the semantic relationship between two or more classifiers that involve connections among their instances."

Include the following associations in a domain model:

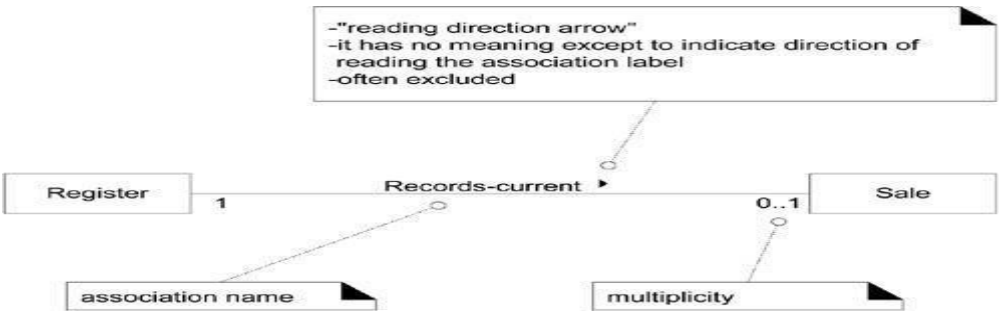
- Associations for which knowledge of the relationship needs to be preserved for some duration ("need-to-remember" associations).
- Associations derived from the Common Associations List.

Guideline 1. Avoid Adding Many Associations

- We need to avoid adding too many associations to a domain model. In a graph with n nodes, there can be (n·(n-1))/2 associations to other nodes-a potentially very large number. A domain model with 20 classes could have 190 associations lines!
- During domain modeling, an association is not a statement about data flows, database foreign key relationships, instance variables, or object connections in a software solution; it is a statement that a relationship is meaningful in a purely conceptual perspective-in the real domain.

Applying UML: Association Notation

An association is represented as a line between classes with a capitalized association name. See Figure



The UML notation for associations.

The ends of an association may contain a multiplicity expression indicating the numerical relationship between instances of the classes.

The association is inherently bidirectional, meaning that from instances of either class, logical traversal to the other is possible. This traversal is purely abstract; it is not a statement about connections between software entities.

An optional "reading direction arrow" indicates the direction to read the association name; it does not indicate direction of visibility or navigation. If the arrow is not present, the convention is to read the association from left to right or top to bottom.

Guideline 2: To Name an Association in UML

Name an association based on a ClassName-VerbPhrase - ClassName format where the verb phrase creates a sequence that is readable and meaningful. Simple association names such as "Has" or "Uses" are usually poor, as they seldom enhance our understanding of the domain.

For example,

- Sale Paid-by CashPayment
  - bad example (doesn't enhance meaning): Sale Uses CashPayment
- Player Is-on Square
  - bad example (doesn't enhance meaning): Player Has Square

Association names should start with a capital letter, since an association represents a classifier of links between instances; in the UML, classifiers should start with a capital letter.

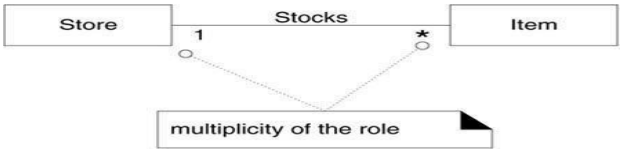
Applying UML: Roles

Each end of an association is called a **role**. Roles may optionally have:

- multiplicity expression
- name
- navigability

Applying UML: Multiplicity

Multiplicity defines how many instances of a class A can be associated with one instance of a class B**Multiplicity on an association.**



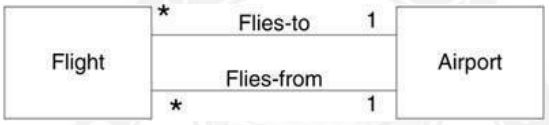
For example, a single instance of a Store can be associated with "many" (zero or more, indicated by the *) Item instances.

Multiplicity values.

Applying UML: Multiple Associations Between Two Classes

The domain of the airline is the relationships between a Flight and an Airport the flying-to and flying-from associations are distinctly different relationships, which should be shown separately.

Multiple associations.



Guideline 3 : To Find Associations with a Common Associations List

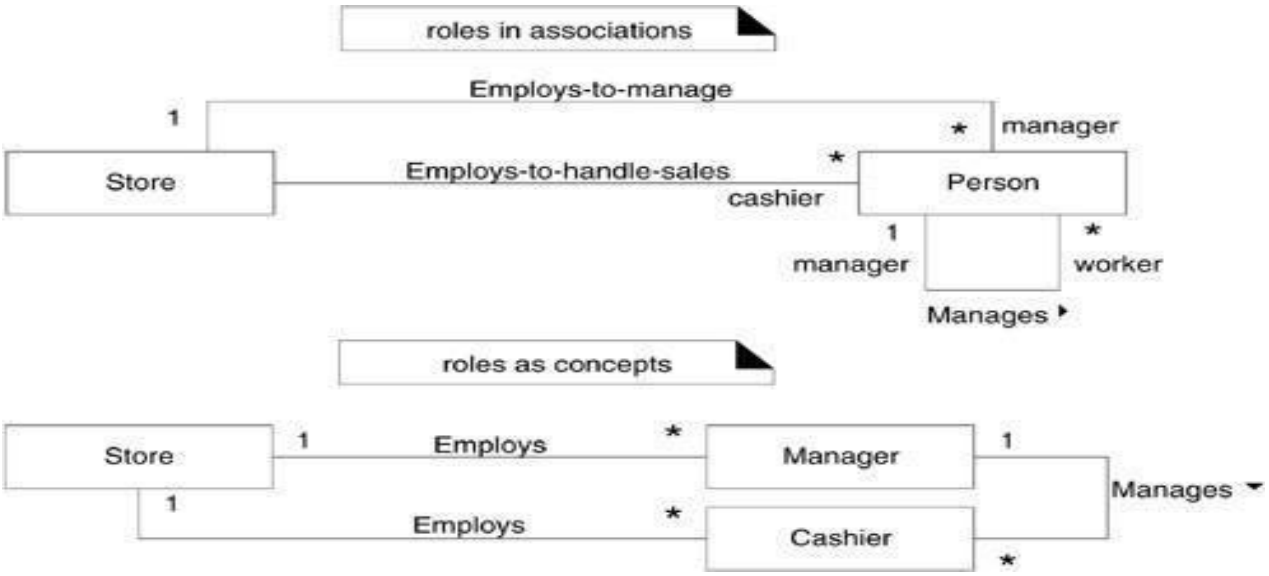
Start the addition of associations by using the list in Table . It contains common categories that are worth considering, especially for business information systems. Examples are drawn from the 1) POS, 2) Monopoly, and 3) airline reservation domains.

Table - Common Associations List.

Category	Examples
A is a transaction related to another transaction B	CashPaymentSale CancellationReservation
A is a line item of a transaction B	SalesLineItemSale

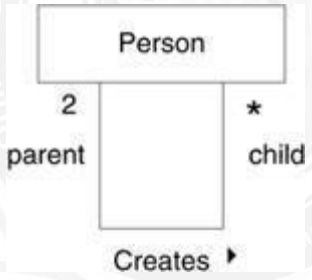
Category	Examples
A is a product or service for a transaction (or line item) B	ItemSalesLineItem(or Sale)FlightReservation
A is a role related to a transaction B	CustomerPayment PassengerTicket
A is a physical or logical part of B	DrawerRegister SquareBoard SeatAirplane
A is physically or logically contained 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

Roles as Concepts versus Roles in Associations : In a domain model, a real-world role especially a human role may be modeled in a number of ways, such as a discrete concept, or expressed as a role in an association.For example, the role of cashier and manager may be expressed in at least the two ways illustrated in Fig



Qualified Associations (Refer pg no 9)

Reflexive Associations : A concept may have an association to itself; this is known as a reflexive association

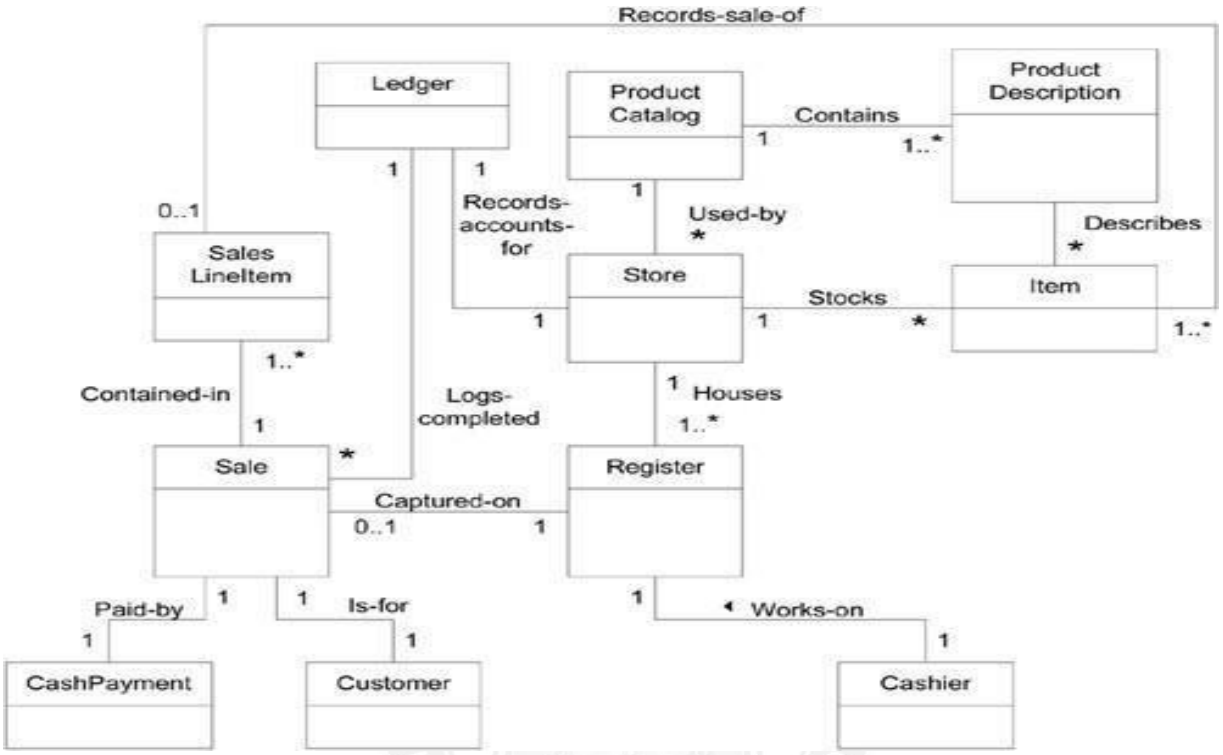


Example: Associations in the Domain Models

Case Study: NextGen POS : The domain model in Figure shows a set of conceptual classes and associations that are candidates for our POS domain model. The associations are primarily derived from the "need-to-remember" criteria of these iteration requirements, and the Common Association List. For example:

- Transactions related to another transaction Sale Paid-by CashPayment.
- Line items of a transaction Sale Contains SalesLineItem.
- Product for a transaction (or line item) SalesLineItem Records-sale-of Item.

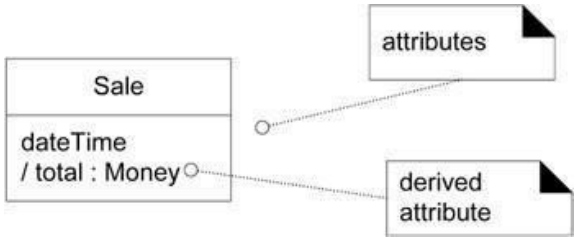
NextGen POS partial domain model.



Attributes : An **attribute** is a logical data value of an object. Include attributes that the requirements (for example, use cases) suggest or imply a need to remember information. For example, a receipt (which reports the information of a sale) in the Process Sale use case normally includes Therefore,

- Sale needs a dateTime attribute.
- Store needs a name and address.
- Cashier needs an ID.

Applying UML- Attribute Notation : Attributes are shown in the second compartment of the class box . Their type and other information may optionally be shown.



Class and attributes.

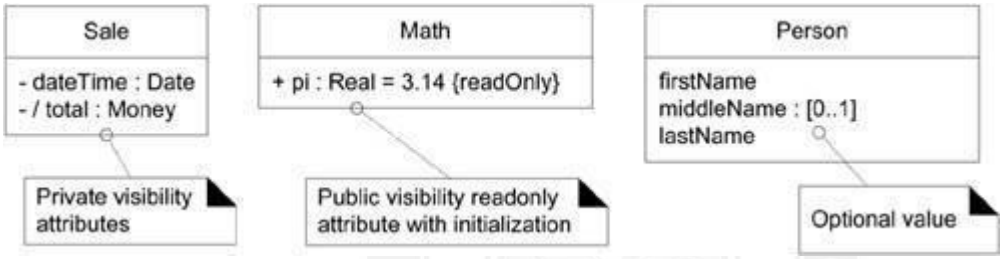
More Notation

The full syntax for an attribute in the UML is:

visibility name : type multiplicity = default {property-string}

Some common examples are shown in Fig

Attribute notation in UML.



{readOnly} is probably the most common property string for attributes.

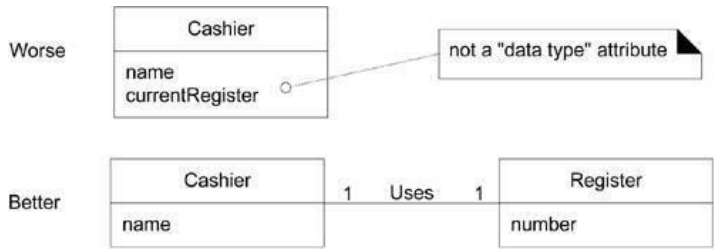
Multiplicity can be used to indicate the optional presence of a value, or the number of objects that can fill a (collection) attribute.

Derived Attributes : When we want to communicate that 1) this is a noteworthy attribute, but 2) it is derivable, we use the UML convention: a / symbol before the attribute name.

Guideline 1 : Suitable Attribute Types - Focus on Data Type Attributes in the Domain Model

Most attribute types should be what are often thought of as "primitive" data types, such as numbers and Booleans. For example, the current Register attribute in the Cashier class in Figure is undesirable because its type is meant to be a Register, which is not a simple data type (such as Number or String).

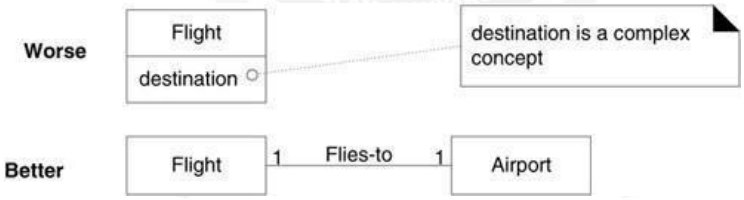
Relate with associations, not attributes



Guideline : The attributes in a domain model should preferably be data types. Very common data types include: Boolean, Date (or DateTime), Number, Character, String (Text), Time. Other common types include: Address, Color, Geometrics (Point, Rectangle), Phone Number, Social Security Number, Universal Product Code (UPC), SKU, ZIP or postal codes, enumerated types

A common confusion is modeling a complex domain concept as an attribute. To illustrate, a destination airport is not really a string; it is a complex thing that occupies many square kilometers of space. Therefore, Flight should be related to Airport via an association, not with an attribute, as shown in Fig.

Don't show complex concepts as attributes; use associations.



Guideline : Relate conceptual classes with an association, not with an attribute.

Data Types

Attributes in the domain model should generally be data types; informally these are "primitive" types such as number, boolean, character, string, and enumerations (such as Size = {small, large}).

For example, it is not (usually) meaningful to distinguish between:

- Separate instances of the Integer 5.
- Separate instances of the String 'cat'.

- Separate instance of the Date "Nov. 13, 1990".

Guideline 1 : When to define New Data type Classes ?

Guidelines for modeling data types

Represent what may initially be considered a number or string as a new data type class in the domain model if:

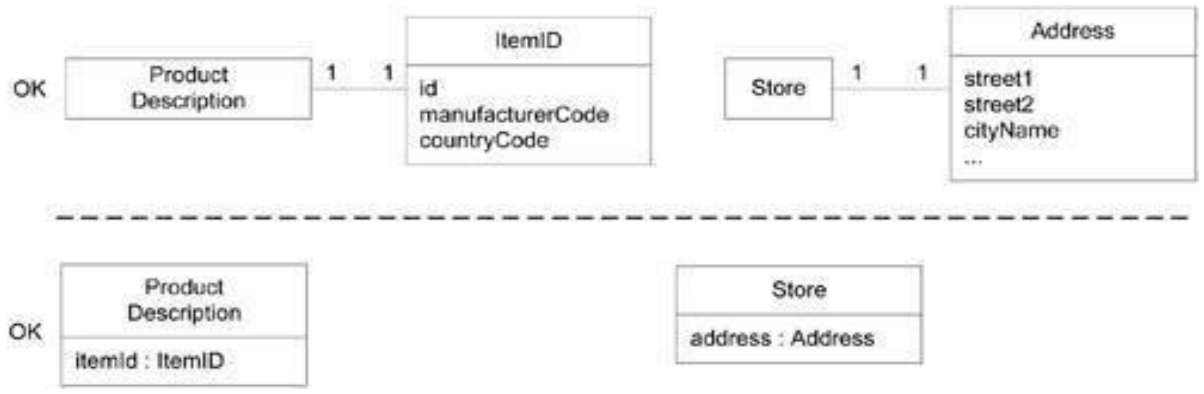
- It is composed of separate sections. –Ex phone number, name of person
- There are operations associated with it, such as parsing or validation. - social security number
- It has other attributes. - promotional price could have a start (effective) date and end date
- It is a quantity with a unit. - payment amount has a unit of currency
- It is an abstraction of one or more types with some of these qualities.

Item identifier in the sales domain is a generalization of types such as Universal Product Code (UPC) and European Article Number (EAN)

Applying these guidelines to the POS domain model attributes yields the following analysis:

- The item identifier is an abstraction of various common coding schemes, including UPC-A, UPC-E, and the family of EAN schemes. These numeric coding schemes have subparts identifying the manufacturer, product, country (for EAN), and a check-sum digit for validation. Therefore, there should be a data type ItemID class, because it satisfies many of the guidelines above.
- The price and amount attributes should be a data type Money class because they are quantities in a unit of currency.
- The address attribute should be a data type Address class because it has separate sections.

Applying UML: Where to Illustrate These Data Type Classes?



Two ways to indicate a data type property of an object.

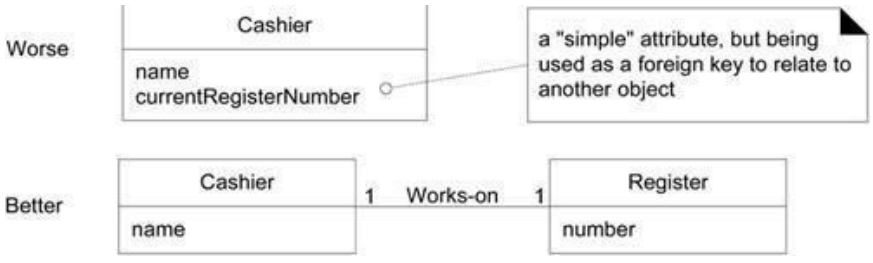
Should the ItemID class be shown as a separate class in a domain model?. Since ItemID is a data type (unique identity of instances is not used for equality testing), it may be shown only in the attribute compartment of the class box, as shown in above Figure. On the other hand, if ItemID is a new type with its own attributes and associations, showing it as a conceptual class in its own box may be informative.

Guideline 2 : No Attributes Representing Foreign Keys

In Following Fig the currentRegisterNumber attribute in the Cashier class is undesirable because its purpose is to relate the Cashier to a Register object. The

better way to express that a Cashier uses a Register is with an association, not with a foreign key attribute.

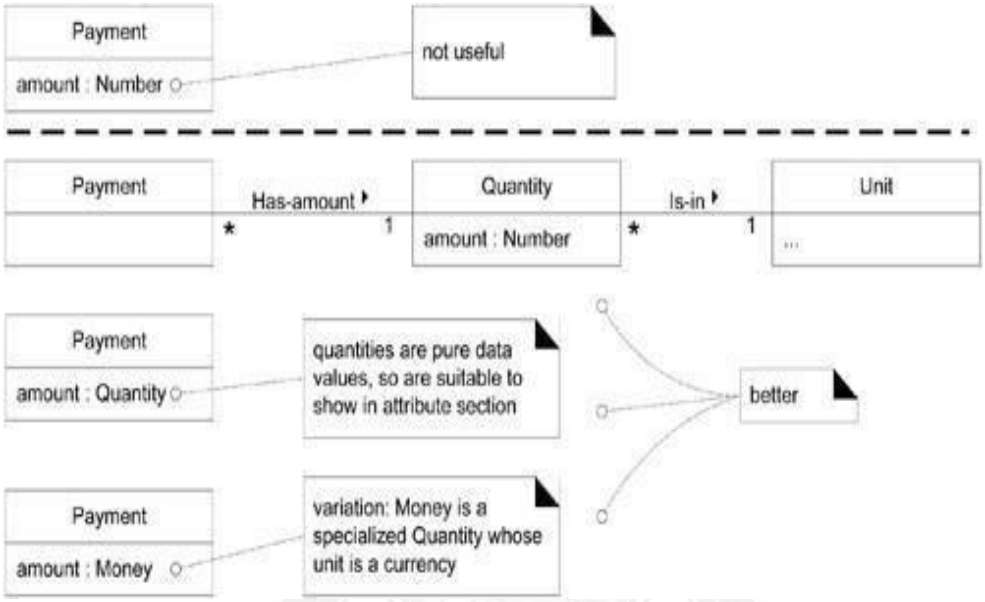
Do not use attributes as foreign keys.



Guideline 3 : Modeling Quantities and Units

Most numeric quantities should not be represented as plain numbers. Consider price

or weight. These are quantities with associated units, and it is common to require knowledge of the unit to support conversions.



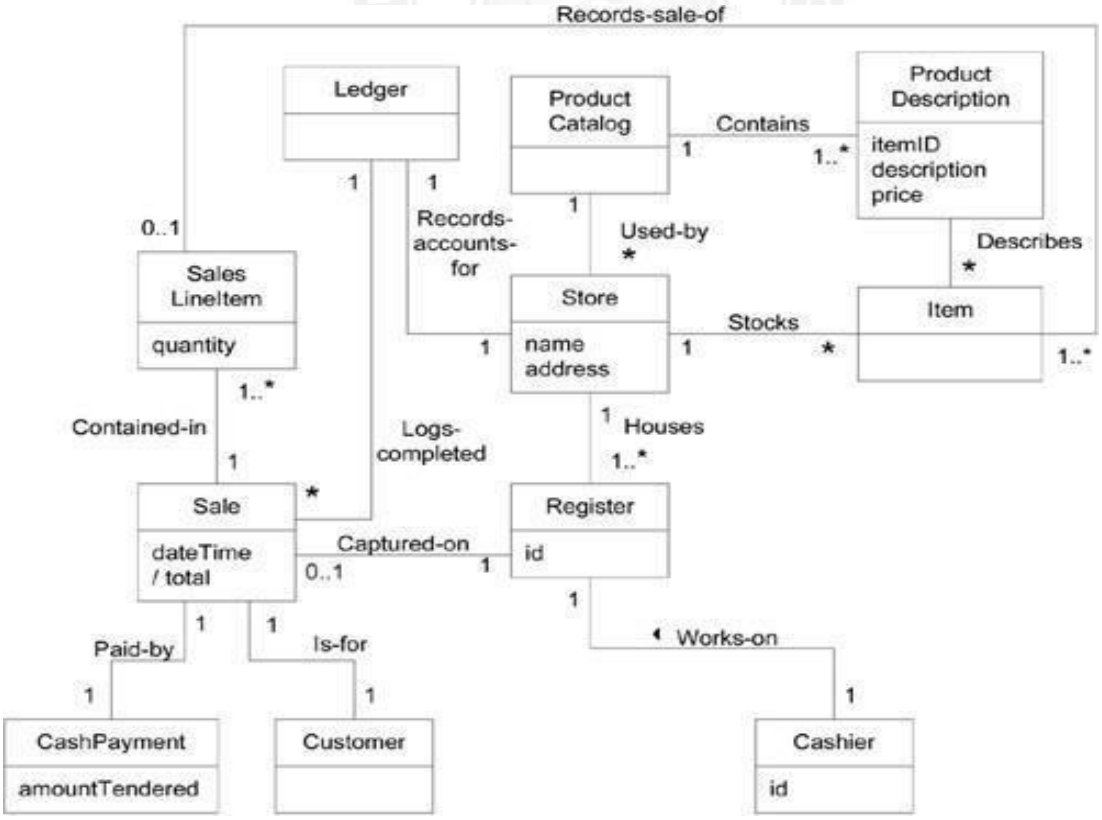
Modeling quantities.

Example: Attributes in the Domain Models -Case Study: NextGen POS

See following Fig. The attributes chosen reflect the information requirements for this iteration the Process Sale cash-only scenarios of this iteration. For example:

CashPayment	amountTendered To determine if sufficient payment was provided, and to calculate change, an amount (also known as "amount tendered") must be captured.

CashPayment	amountTendered To determine if sufficient payment was provided, and to calculate change, an amount (also known as "amount tendered") must be captured.
Product-Description	description To show the description on a display or receipt. itemId To look up a ProductDescription. price To calculate the sales total, and show the line item price.
Sale	dateTime A receipt normally shows date and time of sale, and this is useful for sales analysis.
SalesLineItem	quantity To record the quantity entered, when there is more than one item in a line item sale (for example, five packages of tofu).
Store	address, name The receipt requires the name and address of the store.



NextGen POS partial domain model.