DOMAIN MODEL REFINEMENT

OBJECTIVES

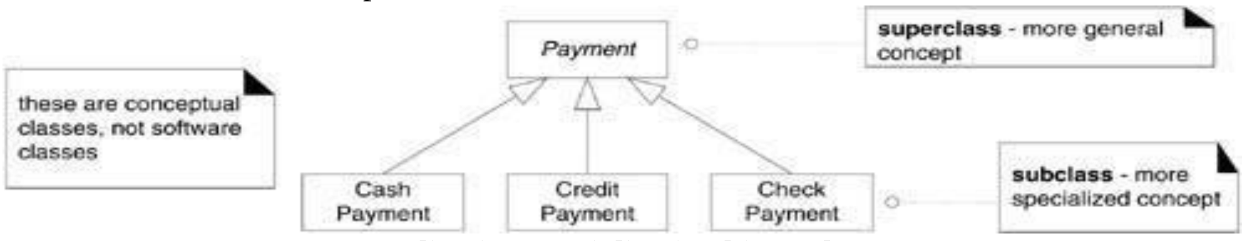
- Refine the domain model with generalizations, specializations, association classes, time intervals, composition, and packages.
- Generalization and specialization are fundamental concepts in domain modeling that support an economy of expression;
- Association classes capture information about an association itself.
- Time intervals capture the important concept that some business objects are valid for a limited time.
- Packages are a way to organize large domain models into smaller units.

Concepts Category List : This Table shows some concepts being considered in this iteration.

Category	Examples
physical or tangible objects	CreditCard, Check
Transactions	CashPayment, CreditPayment, CheckPayment
other computer or electro-mechanical systems external to our system	CreditAuthorizationService, CheckAuthorizationService
abstract noun concepts	
Organizations	CreditAuthorizationService, CheckAuthorizationService
records of finance, work, contracts, legal matters	AccountsReceivable

Generalization

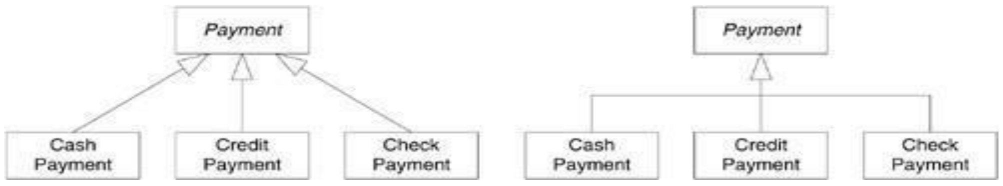
The concepts CashPayment, CreditPayment, and CheckPayment are all very similar. In this situation, it is possible (and useful) to organize them (as in following Figure) into a generalization-specialization class hierarchy (or simply **class hierarchy**) in which the **super class** Payment represents a more general concept, and the **subclasses** more specialized ones.


these are conceptual classes, not software classes
Payment
superclass - more general concept
Cash Payment
Credit Payment
Check Payment
subclass - more specialized concept

Generalization-specialization hierarchy.

Generalization is the activity of identifying commonality among concepts and defining superclass (general concept) and subclass (specialized concept) relationships. Identifying a superclass and subclasses is of value in a domain model because their presence allows us to understand concepts in more general, refined and abstract terms.

Guideline : Identify domain superclasses and subclasses relevant to the current iteration, and illustrate them in the Domain Model.



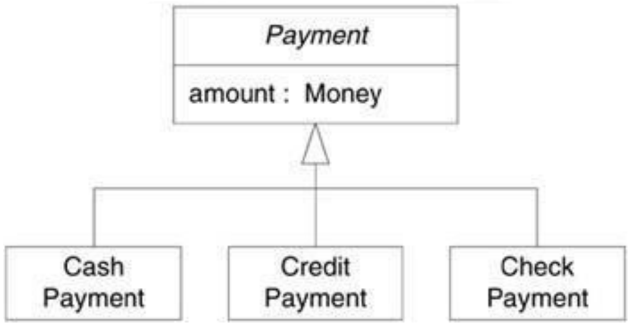
Class hierarchy with separate and shared arrow notations.

<u>Defining Conceptual Superclasses and Subclasses :</u>

Definition ***:*** A conceptual super class definition is more general or encompassing than a subclass definition.

For example, consider the superclass Payment and its subclasses (CashPayment, and so on). Assume the definition of Payment is that it represents the transaction of transferring money (not necessarily cash) for a purchase from one party to another, and that all payments have an amount of money transferred. The model corresponding to this is shown in following Figure.

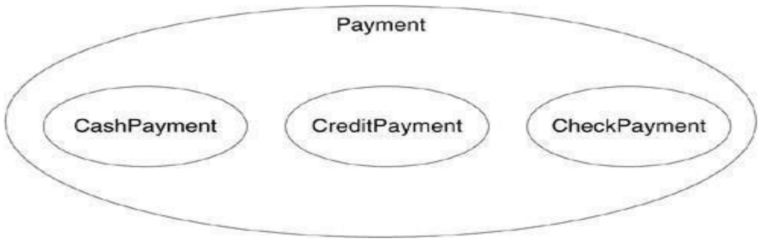
Payment class hierarchy.



A Credit Payment is a transfer of money via a credit institution which needs to be authorized. My definition of Payment encompasses and is more general than my definition of Credit Payment.

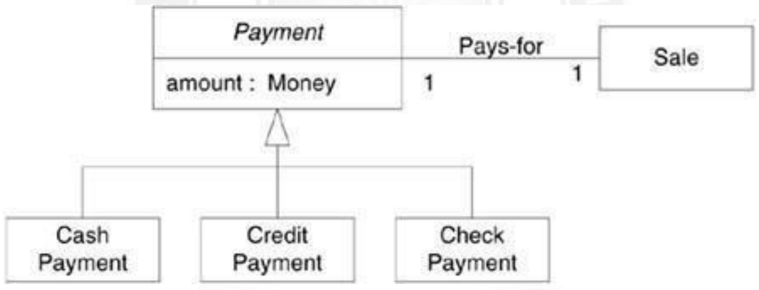
Definition : All members of a conceptual subclass set are members of their superclass set.For example, in terms of set membership, all instances of the set CreditPayment are also members of the set Payment. In a Venn diagram, this is shown as in following Fig

Venn diagram of set relationships.



Conceptual Subclass Definition Conformance : When a class hierarchy is created, statements about superclasses that apply to subclasses are made. For example, the following Figure states that all Payments have an amount and are associated with a Sale.

Subclass conformance.



Guideline: <u>100% Rule</u>

100% of the conceptual superclass's definition should be applicable to the subclass. The subclass must conform to 100% of the superclass's:

- attributes
- associations

Conceptual Subclass Set Conformance : A conceptual subclass should be a member of the set of the superclass. Thus, CreditPayment should be a member of the set of Payments.

Guideline: <u>Is-a Rule</u>
All the members of a subclass set must be members of their superclass set.

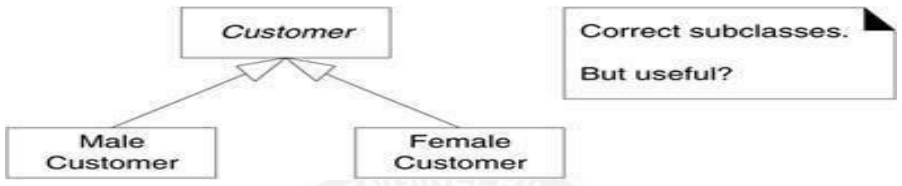
In natural language, this can usually be informally tested by forming the statement: Subclass is a Superclass.

Guideline :Correct Conceptual Subclass
A potential subclass should conform to the:

- 100% Rule (definition conformance)
- Is-a Rule (set membership conformance)

When to Define a Conceptual Subclass?

Definition: A conceptual class partition is a division of a conceptual class into disjoint subclasses. For example, in the POS domain, Customer may be correctly partitioned (or subclassed) into MaleCustomer and FemaleCustomer. But is it relevant or useful to show this in our model (see following figure)? This partition is not useful for our domain; the next section explains why



Legal conceptual class partition, but is it useful in our domain

Motivations to Partition a Conceptual Class into Subclasses

Create a conceptual subclass of a superclass when:

1. The subclass has additional attributes of interest.
2. The subclass has additional associations of interest.
3. The subclass concept is operated on, handled, reacted to, or manipulated differently than the superclass or other subclasses, in ways that are of interest.
4. The subclass concept represents an animate thing (for example, animal, robot) that behaves differently than the superclass or other subclasses, in ways that are of interest.

Based on the above criteria, it is not compelling to partition Customer into the subclasses MaleCustomer and FemaleCustomer because they have no additional attributes or associations, are not operated on (treated) differently, and do not behave differently in ways that are of interest . This table shows some examples of class partitions from the domain of payments and other areas, using these criteria

Example subclass partitions

Conceptual Subclass Motivation	Examples
The subclass has additional attributes of interest.	Payments not applicable.Library Book, subclass of LoanableResource, has an ISBN attribute.

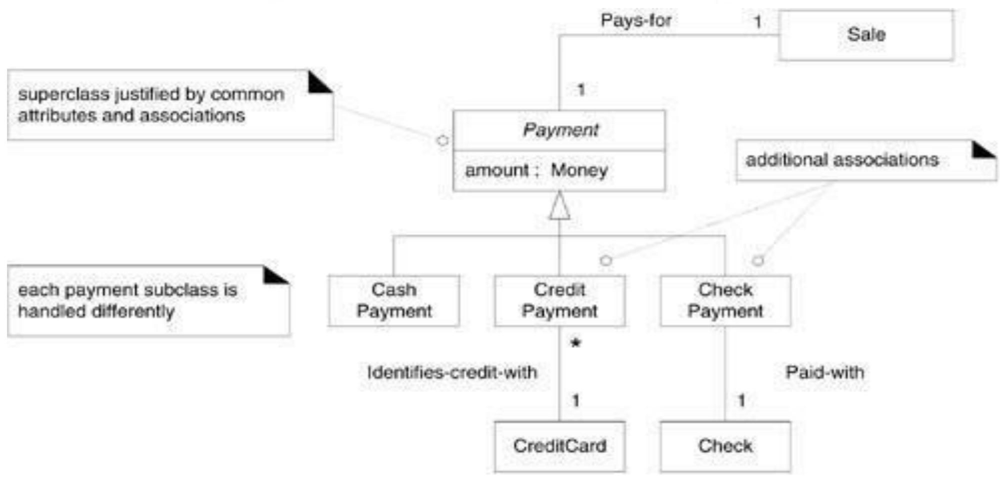
Conceptual Subclass Motivation	Examples
The subclass has additional associations of interest.	Payments CreditPayment, subclass of Payment, is associated with a CreditCard. Library Video, subclass of LoanableResource, is associated with Director.
The subclass concept is operated upon, handled, reacted to, or manipulated differently than the superclass or other subclasses, in ways that are of interest.	Payments CreditPayment, subclass of Payment, is handled differently than other kinds of payments in how it is authorized. Library Software, subclass of LoanableResource, requires a deposit before it may be loaned.
The subclass concept represents an animate thing (for example, animal, robot) that behaves differently than the superclass or other subclasses, in ways that are of interest.	Payments not applicable. Library not applicable. Market Research MaleHuman, subclass of Human, behaves differently than FemaleHuman with respect to shopping habits.

When to Define a Conceptual Superclass?

Motivations to generalize and define a superclass: Guideline
Create a superclass in a generalization relationship to subclasses when:

- The potential conceptual subclasses represent variations of a similar concept.
- The subclasses will conform to the 100% and Is-a rules.
- All subclasses have the same attribute that can be factored out and expressed in the superclass.
- All subclasses have the same association that can be factored out and related to the superclass.

NextGen POS Conceptual Class Hierarchies

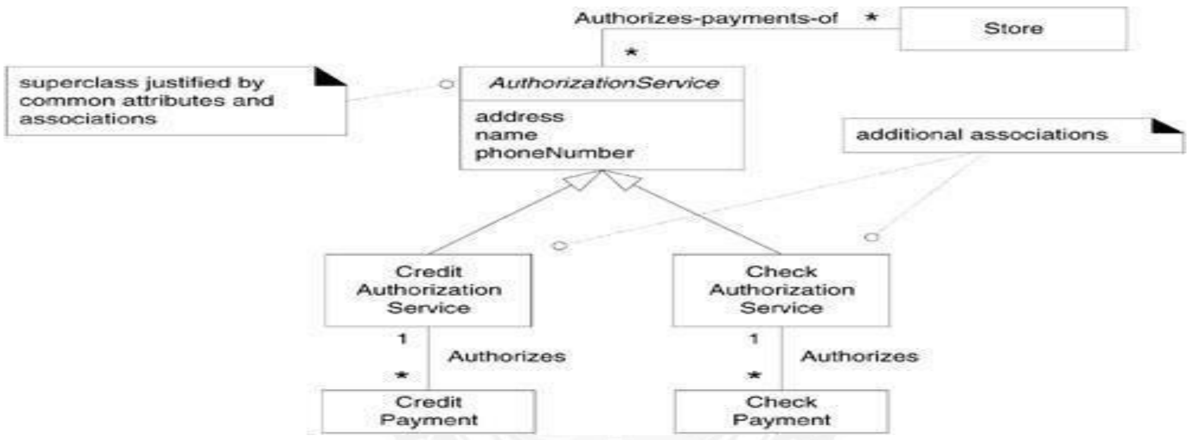


Justifying Payment subclasses.

Payment Classes : Based on the above criteria for partitioning the Payment class, it is useful to create a class hierarchy of various kinds of payments. The justification for the superclass and subclasses is shown in Figure .

Authorization Service Classes : Credit and check authorization services are variations on a similar concept, and have common attributes of interest. This leads to the class hierarchy in following Figure.

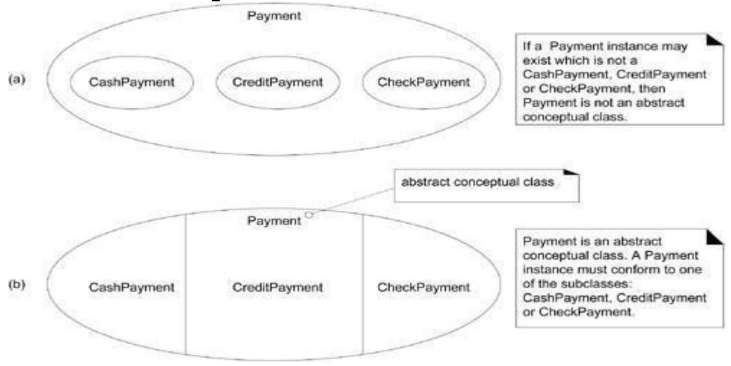
Justifying the AuthorizationService hierarchy



Abstract Conceptual Classes

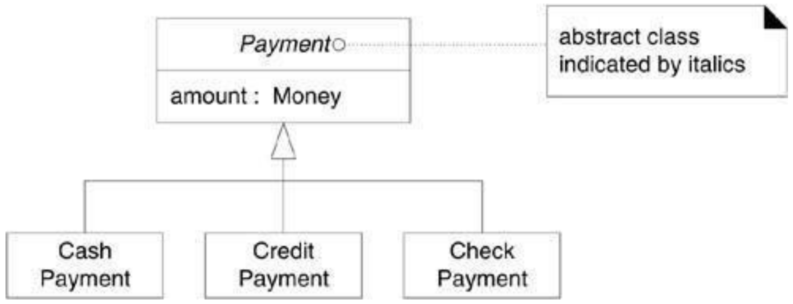
Definition : If every member of a class C must also be a member of a subclass, then class C is called an abstract conceptual class.For example, assume that every Payment instance must more specifically be an instance of the subclass CreditPayment, CashPayment, or CheckPayment. This is illustrated in the Venn diagram of Figure (b). Since every Payment member is also a member of a subclass, Payment is an abstract conceptual class by definition.

Abstract conceptual classes.


Payment
(a)
CashPayment
CreditPayment
CheckPayment
If a Payment instance may exist which is not a CashPayment, CreditPayment or CheckPayment, then Payment is not an abstract conceptual class.
abstract conceptual class
Payment
(b)
CashPayment
CreditPayment
CheckPayment
Payment is an abstract conceptual class. A Payment instance must conform to one of the subclasses: CashPayment, CreditPayment or CheckPayment.

Abstract Class Notation in the UML : To review, the UML provides a notation to indicate abstract classes the class name is italicized

Abstract class notation.



Guideline : Identify abstract classes and illustrate them with an italicized name in the Domain Model, or use the {abstract} keyword.

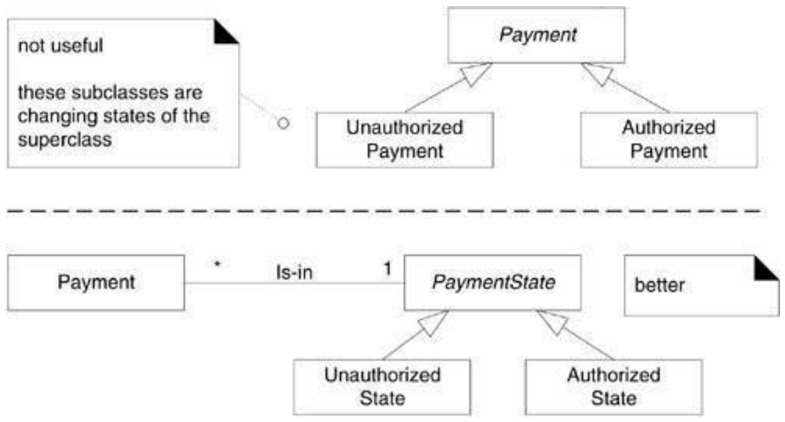
<u>Modeling Changing States</u>

Assume that a payment can either be in an unauthorized or authorized state, and it is meaningful to show this in the domain model. As shown in Figure , one modeling approach is to define subclasses of Payment: Unauthorized Payment and Authorized Payment.

Guideline : Do not model the states of a concept X as subclasses of X. Rather, either:

- Define a state hierarchy and associate the states with X, or
- Ignore showing the states of a concept in the domain model; show the states in state diagrams instead.

Modeling changing states.



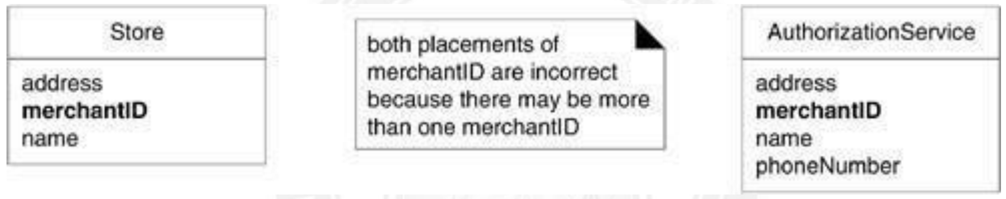
Association Classes

The following domain requirements set the stage for association classes:

- Authorization services assign a merchant ID to each store for identification during communications.
- A payment authorization request from the store to an authorization service needs the merchant ID that identifies the store to the service.
- Furthermore, a store has a different merchant ID for each service.

Placing merchantID in Store is incorrect because a Store can have more than one value for merchantID. The same is true with placing it in AuthorizationService (see Figure).

Inappropriate use of an attribute.

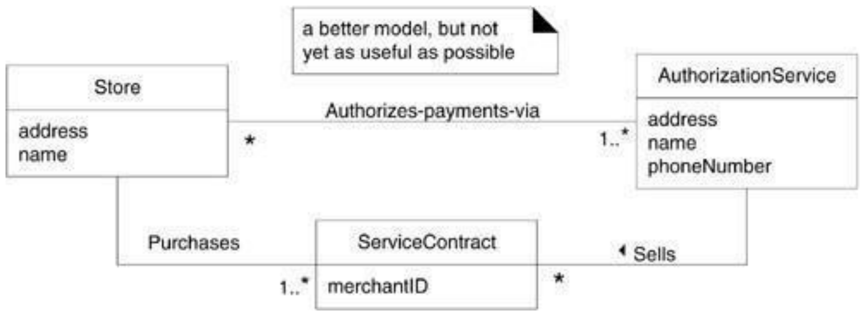

Store
address
merchantID
name
both placements of merchantID are incorrect because there may be more than one merchantID
AuthorizationService
address
merchantID
name
phoneNumber

Guideline : In a domain model, if a class C can simultaneously have many values for the same kind of attribute A, do not place attribute A in C. Place attribute A in another class that is associated with C.

For example:

- A Person may have many phone numbers. Place phone number in another class, such as PhoneNumber or ContactInformation, and associate many of these to Person.

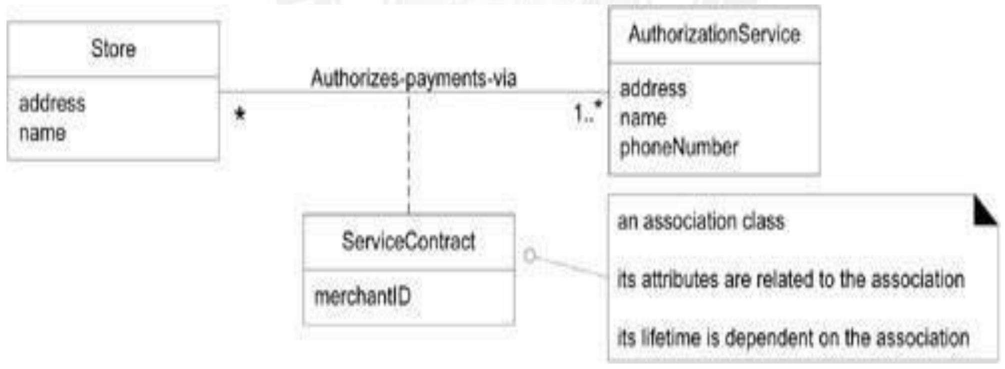
First attempt at modeling the merchantID problem.



The fact that both Store and AuthorizationService are related to ServiceContract is a clue that it is dependent on the relationship between the two. The merchantID may be thought of as an attribute related to the association between Store and AuthorizationService.

This leads to the notion of an association class, in which we can add features to the association itself. ServiceContract may be modeled as an association class related to the association between Store and AuthorizationService.

.



An association class

Guideline : Clues that an association class might be useful in a domain model:

- An attribute is related to an association.
- Instances of the association class have a lifetime dependency on the association.
- There is a many-to-many association between two concepts and information associated with the association itself.