CS6302 Object Oriented Analysis and Design

V-Sem-CSE **V-Sem-IT** **Anna University**
2013 Regulations

Question Bank
UNIT-III

1) Define a Domain Model.

Object-oriented analysis is concerned with creating a description of the domain from the perspective of objects. There is an identification of the concepts, attributes, and associations that are considered noteworthy.

The result can be expressed in a **domain model** that shows the *noteworthy* domain concepts or objects. For example, a partial domain model is shown in Figure 1.2.

. It can be noted that a domain model is not a description of software objects; it is a visualization of the concepts or mental models of a real-world domain and it is also called a **conceptual object model**.

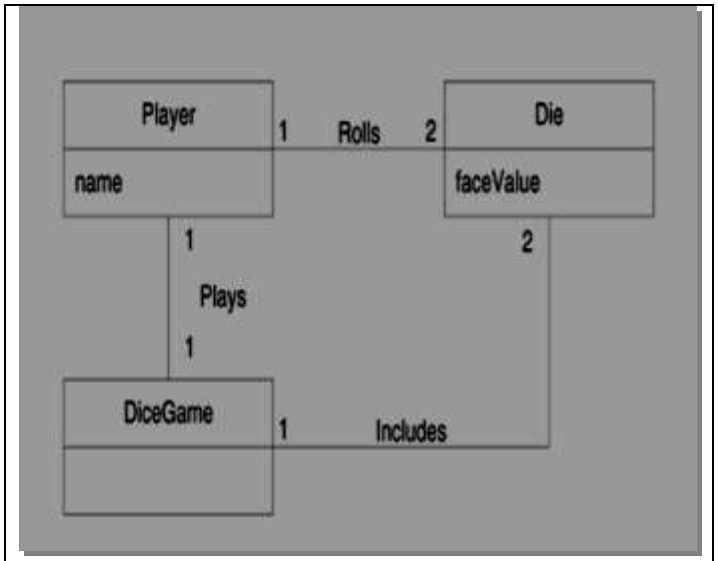


Figure 1.2. Partial domain model of the dice game

2) Define a) Conceptual Class,b) Software Class,c) Implementation Class.

Conceptual class - real-world concept or thing. A conceptual or essential perspective. The UP Domain Model contains conceptual classes.

Software class - a class representing a specification or implementation perspective of a software component, regardless of the process or method.

Implementation class - a class implemented in a specific OO language such as Java.

3) What are the different UP Phases?

An UP Project organizes work and iterations across four major phases :

1) **Inception** – approximate Vision,Business case,Scope,vague estimates

2) **Elaboration** – Refined BVision,iterative implementation of the core architecture,resolution of high risks,identification of most requirements and scope,more realistic estimates
3) **Construction** – Iterative implementation of the remaining lower risk and easier elements,and preparation for deployment
4) **Transition** – beta tests, deployment

4) What are the UP disciplines?

In the UP,an artifact is the general term for any work product : Code,Web Graphics,Schema,Test Documents,diagrams,model and so on.

Some of the artifacts in the following Disciplines are :

a) Business Modeling – The Domain Model artifact,to visualize noteworthy concepts in the application domain
b) Requirements – The Use Case Model and Supplementary specification artifacts to capture functional and non-functional requirements
c) Design – The Design Model artifact,to design the software artifacts
d) Implementation – Programming and building the system,not deploying it

5) What is Inception?

Envision the product scope,vision,and business case.

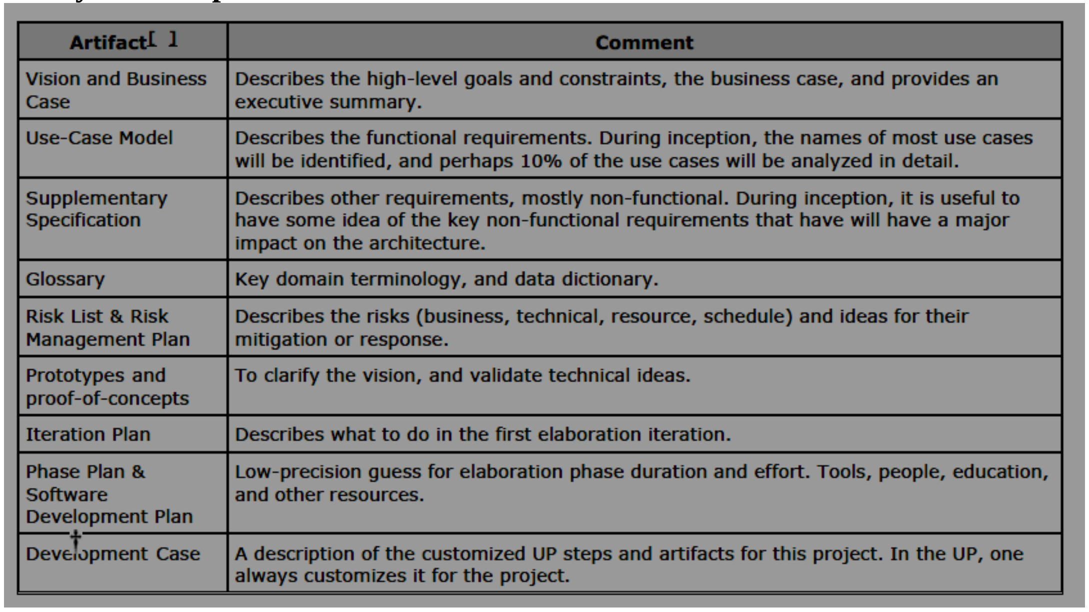
Do the stakeholders have basic agreement on the vision of the project,and is it worth investing in serious investigations?

The purpose of inception stage is not to define all the requirements. The Up is not the waterfall and the first phase inception is not the time todo all requirements or create believable estimates or plans. That happens during elaboration.

6) How long is the Inception phase can be?

The intent of inception is to establish some initial common vision for the objectives of the project,determine if it is feasible,and decide if its is worth some serious investigation in elaboration. It can be brief.

7) List any five inception artifacts.



Artifact[ ]	Comment
Vision and Business Case	Describes the high-level goals and constraints, the business case, and provides an executive summary.
Use-Case Model	Describes the functional requirements. During inception, the names of most use cases will be identified, and perhaps 10% of the use cases will be analyzed in detail.
Supplementary Specification	Describes other requirements, mostly non-functional. During inception, it is useful to have some idea of the key non-functional requirements that have will have a major impact on the architecture.
Glossary	Key domain terminology, and data dictionary.
Risk List & Risk Management Plan	Describes the risks (business, technical, resource, schedule) and ideas for their mitigation or response.
Prototypes and proof-of-concepts	To clarify the vision, and validate technical ideas.
Iteration Plan	Describes what to do in the first elaboration iteration.
Phase Plan & Software Development Plan	Low-precision guess for elaboration phase duration and effort. Tools, people, education, and other resources.
Development Case	A description of the customized UP steps and artifacts for this project. In the UP, one always customizes it for the project.

8) Define Requirements.

Requirements are capabilities and conditions to which the system – and more broadly,the project must conform.
The UP promotes a set of best practices,one of which is to manage requirements.
In the context of changing and and unclear stakeholder's wishes – Managing requirements means – a systematic approach to finding,documenting,organizing,and tracking the changing requirements of a system.
A prime challenge of requirements analysis is to find,communicate,and remember(Towrite down) what is really needed,in a formthat clearly speaks to the client and development team members.

9) What are the types and categories of requirements?

In the UP, requirements are categorized according to the FURPS+ model [Grady92], a useful mnemonic with the following meaning :

- **Functional -** features, capabilities, security.
- **Usability -** human factors, help, documentation.
- **Reliability -** frequency of failure, recoverability, predictability.
- **Performance -** response times, throughput, accuracy, availability, resource usage.
- **Supportability -** adaptability, maintainability, internationalization, configurability.

The "+" in FURPS+ indicates ancillary and sub-factors, such as:

- **Implementation** resource limitations, languages and tools, hardware, ...
- **Interface** constraints imposed by interfacing with external systems.
- **Operations** system management in its operational setting.
- **Packaging** for example, a physical box.
- **Legal** licensing and so forth.

It is helpful to use FURPS+ categories (or some categorization scheme) as a checklist for requirements coverage, to reduce the risk of not considering some important facet of the system.

Some of these requirements are collectively called the **quality attributes, quality requirements**, or the "- ilities" of a system. These include usability, reliability, performance, and supportability. In common usage, requirements are categorized as **functional** (behavioral) or **non-functional** (everything else); some dislike this broad generalization [BCK98], but it is very widely used.

10) What are the key requirement artifacts?
The Key requirements artifacts are :

Use-Case Model - A set of typical scenarios of using a system. There are primarily for functional
(behavioral) requirements.
Supplementary Specification - Basically, everything not in the use cases. This artifact is primarily for all non-functional requirements, such as performance or licensing. It is also the place to record functional features not expressed (or expressible) as use cases; for example, a report generation.
Glossary - In its simplest form, the Glossary defines noteworthy terms. It also encompasses the concept of the data dictionary, which records requirements related to data, such as validation rules, acceptable values, and so forth. The Glossary can detail any

element: an attribute of an object, a parameter of an operation call, a report layout, and so forth.

Vision - Summarizes high-level requirements that are elaborated in the Use-Case Model and Supplementary Specification, and summarizes the business case for the project. A short executive overview document for quickly learning the project's big ideas.

Business Rules - Business rules (also called Domain Rules) typically describe requirements or policies that transcend one software projectthey are required in the domain or business, and many applications may need to conform to them. An excellent example is government tax laws. Domain rule details *may* be recorded in the Supplementary Specification, but because they are usually more enduring and applicable than for one software project, placing them in a central Business Rules artifact (shared by all analysts of the company) makes for better reuse of the analysis effort.

11) What are Use Cases?

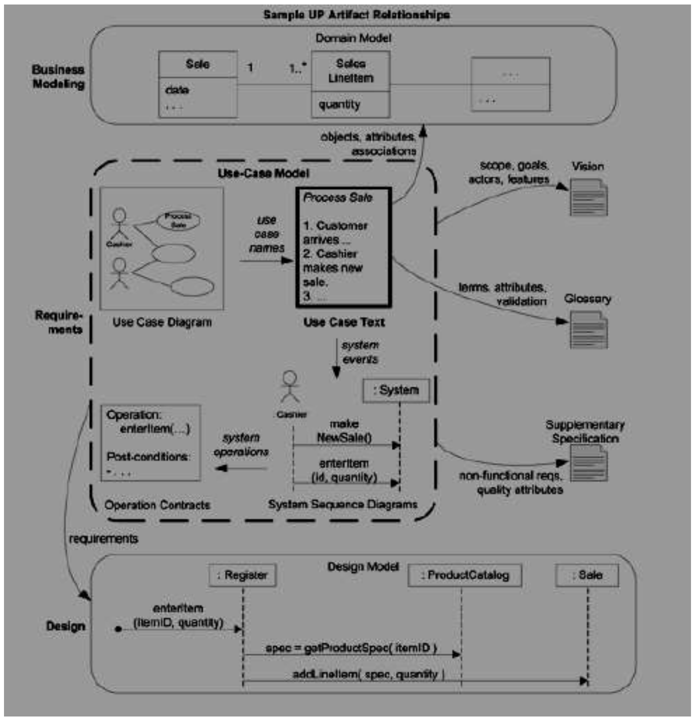
Informally, **use cases** are ***text stories*** of some **actor** using a **system** to meet **goals**.

Use Case Example :

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the **POS** system to record each purchased 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 i**nventory**. The customer receives a **receipt** from the system and then leaves with the items.

Notice that ***use cases are not diagrams, they are text.*** Focusing on secondary-value UML use case diagrams rather than the important use case text is a common mistake for use case novices.

Use cases often need to be more detailed or structured than this example, but the essence is **discovering and recording functional requirements by writing stories** of using a system to fulfill user goals; that is, *cases of use*.



12) What is Use-case Modeling?

Use-Case Model is the set of all written use cases; it is a model of the system's functionality and environment. **Use cases** are **text documents, not diagrams**, and use-case modeling is primarily an act of **writing text**, not **drawing diagrams**.
The Use-Case Model is not the only requirement artifact in the UP. There are also the **Supplementary Specification, Glossary, Vision, and Business Rules**. These are all useful for requirements analysis, but secondary at this point.
The Use-Case Model may optionally include a **UML use case diagram** to show the **names** of **use cases and actors, and their relationships.** This gives a nice **context diagram** of a system and its environment. It also provides a quick way to list the use cases by name.
There is nothing object-oriented about use cases; we're not doing OO analysis when writing them. Use cases are a key requirements input to classic OOA/D.

13) What are the three kinds of Actors?

Definition: What are Three Kinds of Actors?
Actors are roles played not only by people, but by organizations, software, and machines. There are three kinds of external actors in relation to the SuD:

1) **Primary actor** has user goals fulfilled through using services of the SuD. For example, the cashier.
   Why identify? To find user goals, which drive the use cases**.**
2) **Supporting actor** provides a service (for example, information) to the SuD. The automated payment authorization service is an example. Often a computer system, but could be an organization or person.
   Why identify? To clarify external interfaces and protocols.
3) **Offstage actor** has an interest in the behavior of the use case, but is not primary or supporting; for example, a government tax agency.
   Why identify? To ensure that *all* necessary interests are identified and satisfied.

14) What are preconditions and postconditions?

Preconditions and Success Guarantees (Postconditions)

Preconditions state what *must always* be true before a scenario is begun in the use case. Preconditions are *not* tested within the use case; rather, they are conditions that are assumed to be true. Typically, a precondition implies a scenario of another use case, such as logging in, that has successfully completed.
Success guarantees (or postconditions) state what must be true on successful completion of the use case either the main success scenario or some alternate path. The guarantee should meet the needs of all stakeholders.
EXAMPLE :
Preconditions: Cashier is identified and authenticated.

Success Guarantee (Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated.

15) **How to find Use cases?**

Guideline: How to Find Use Cases

Use cases are defined to satisfy the goals of the primary actors. Hence, the basic procedure is:

1) Choose the system boundary. Is it just a software application, the hardware and application as a unit, that plus a person using it, or an entire organization?
2) Identify the primary actorsthose that have goals fulfilled through using services of the system.
3) Identify the goals for each primary actor.
4) Define use cases that satisfy user goals; name them according to their goal. Usually, user-goal level use cases will be one-to-one with user goals, but there is at least one exception, as will be examined.

16) **What does Use case Diagram represent? Give an example.**

Applying UML: Use Case Diagrams

The UML provides use case diagram notation to illustrate the names of use cases and actors, and the relationships between them.

Guideline

A simple use case diagram is drawn in conjunction with an actor-goal list.

A use case diagram is an excellent picture of the system context; it makes a good context diagram, that is, showing the boundary of a system, what lies outside of it, and how it gets used. It serves as a communication tool that summarizes the behavior of a system and its actors. A sample *partial* use case context diagram for the NextGen system is shown below.

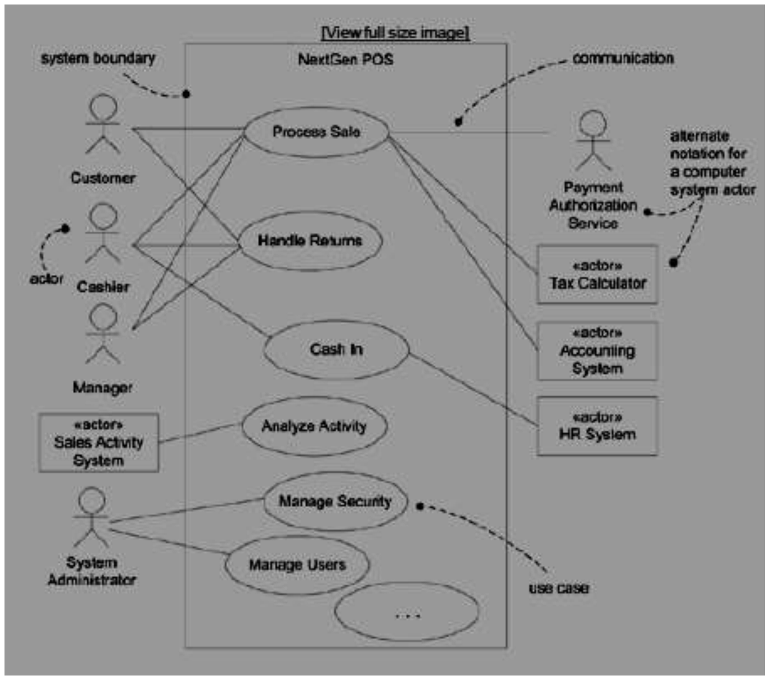


Figure 1.5. Partial use case context diagram.

17) Define use case relationships a)Include b) Extend c) Generalization

Include Relationship

Use cases can be related to each other.

It is common to have some partial behavior that is common across several use cases. For example, the description of paying by credit occurs in several use cases, including *Process Sale, Process Rental, Contribute to Lay-away Plan,* and so forth. Rather than duplicate this text, it is desirable to separate it into its own subfunction use case, and indicate its inclusion. This is simply refactoring and linking text to avoid duplication.

The include relationship can be used for most use case relationship problems.

To summarize:

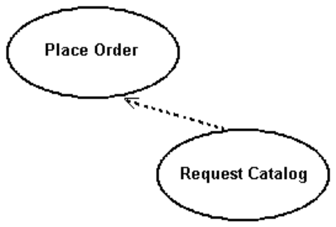
Factor out subfunction use cases and use the *Include* relationship when:

• They are duplicated in other use cases.

• A use case is *very* complex and long, and separating it into subunits aids comprehension.

Extend Relationship

- Extend puts additional behavior in a use case that does not know about it.
- It is shown as a dotted line with an arrow point and labeled <<extend>>
- In this case, a customer can request a catalog when placing an order



The following diagram illustrates use case Include relationship

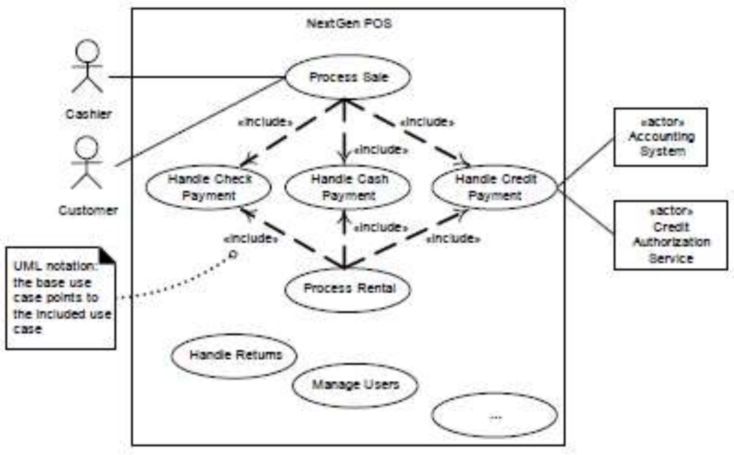
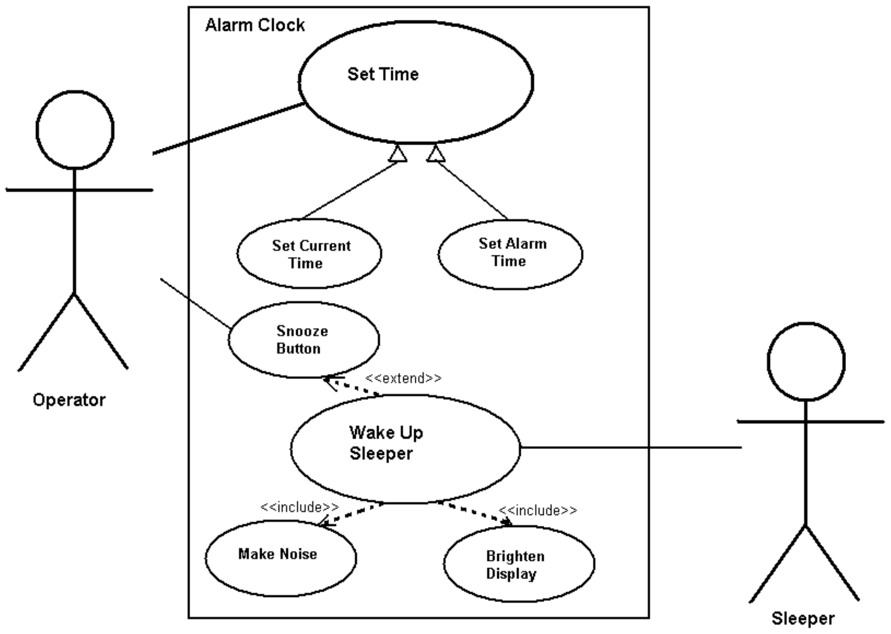


Figure 25.1 Use case include relationship in the Use-Case Model.

Example of Use Case Extend relationship



18) What is Elaboration?

- Elaboration often consists of two or more iterations(2 to 6 weeks duration)
- Each iteration is time-boxed(i.e. End Date fixed)
- Elaboration is not Design Phase(i.e. the model is not fully developed)
- Also it is not throw away Prototype;rather the code and design are production quality

In other words,Elaboration is the initial series of iterations during which

- The core ,risky software architecture is programmed and tested
- The majority of requirements are discovered and stabilized
- The major risks are mitigated or reduced

19) Define Elaboration.

Build the core architecture,resolve the high risk elements,Define most Requirements,and Estimate the overall schedule and resources

20) Define the first iteration is elaboration phase.

Iteration-1 of Elaboration Phase emphasizes fundamental and common OOA/D skills used in building OO Systems.

Example – NextGen POS

:

Iteration 1 Requirements

The requirements for the first iteration of the NextGen POS application follow:

- 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, just a simple happy path scenario, and the design and implementation to support it.
- There is no collaboration with external services, such as a tax calculator or product database.
- No complex pricing rules are applied.
- The design and implementation of the supporting UI ,database,are done(Not in detail)
- Subsequent iterations will grow on this foundation.

21) What happens in inception?

Inception is a short step to elaboration. It determines basic feasibility,risk and scope,to decide if the project is worth more serious investigation

22) What are the likely activities and artifacts in inception?

- A short requirement workshop
- Most actors,goals and Use Cases named.
- Most use cases written in brief format; 1—20% of use cases written in fully dressed format
- Most influential and risky quality requirements identified
- Supplementary specification written(Version One)
- Risk List
- Technical proof-of-Concept,User Interface-Oriented Prototypes
- Decision on components : to buy/build/reuse taken(Eg. To buy Tax calculation package
- High Level candidate Architecture made(not a detailed,final or correct one made)
- Used as starting point of investigation(Eg. A Java Client side application with no Application Server, and no Oracle for Database

23) What artifacts may start in Inception?

Slno	Artifact	Features
01	Domain Model	Visualization of Domain concepts
02	Design Model	➢ Set of Diagrams describing the Logical Design(Software Class Diagrams,Object Interaction Diagrams,and Package Diagrams)
03	Software Architecture Document	➢ A Learning Aid ➢ Key Architectural Issues & their Resolution in summary form ➢ Summary of outstanding Design Ideas & its motivation
04	Data Model	➢ Database Schemas ➢ Mapping between Objects & Non-Object representations

05	Use Case Story Boards,UI Prototypes	➢ Describe UI,Navigation Paths,Usability Models

24) What is a Domain Model?

A domain Model is the most important and classic model in OO Analysis.

- ➢ It illustrates noteworthy Concepts in a Domain.
- ➢ Source of Inspiration for designing some software objects(which become inputs to several artifacts)

25) Give an example of domain model with UML class Diagram notation.

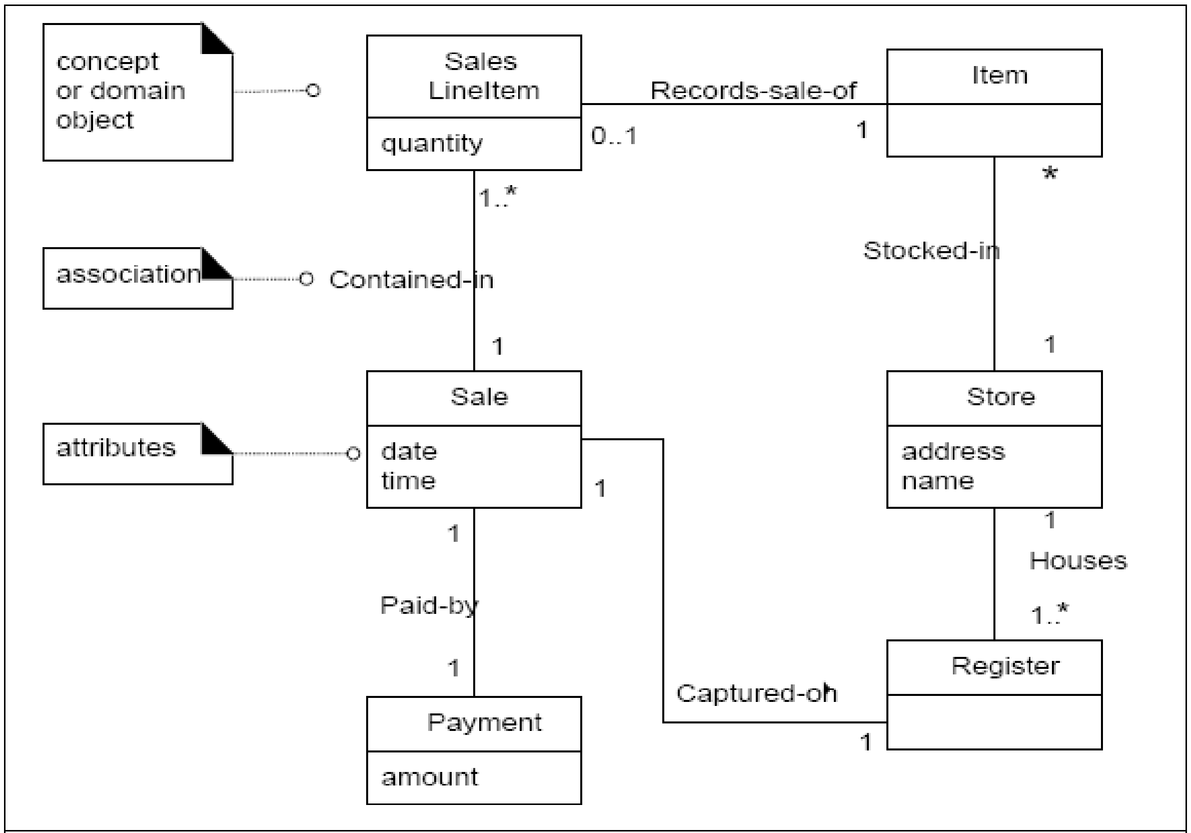


Fig 2.1 Partial Domain Model – A Visual Dictionary

Explanation

A partial Domain Model drawn with UML class diagram notation -:

- ➢ **Conceptual classes of** *Payment* and *Sale* are significant in this domain.
- ➢ **A** *payment* is related to *Sale* which is meaningful to note.
- ➢ The *Sale* has date and time (Attributes we care about)

26) What are the criteria in planning the next iteration during elaboration phase?

Risk	Technical complexity and factors such as uncertainity of effort or usability
Coverage	Major parts of the system are at least touched on early iterations
Criticality	Functions that client consider of high business value

27) Why call a domain model a visual dictionary?

A domain model is a visual dictionary of

- the noteworthy abstractions
- domain vocabulary,and
- information content

A domain model visualizes and relates words or concepts in the domain. It also shows an abstraction of the conceptual classes and shows how they relate to each other.

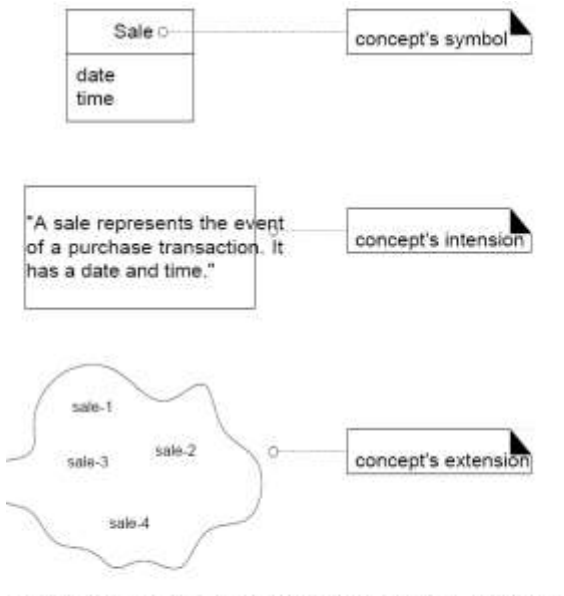
28) What are conceptual classes?

Conceptual Classes

Informally, a conceptual class is an idea, thing, or object.
More formally, a conceptual class may be considered in terms of its symbol, intension, and extension

- **Symbol**—words or images representing a conceptual class.
- **Intension**—the definition of a conceptual class.
- **Extension**—the set of examples to which the conceptual class applies.

For example, consider the conceptual class for the event of a purchase transaction. We may choose to name it by the symbol *Sale.* The intension of a *Sale* may state that it "represents the event of a purchase transaction, and has a date and time." The extension *of Sale* is all the examples of sales; in other words, the set of all sales.



A conceptual class has a symbol, intension, and extension.

29) Are domain and Data Models are the same thing?

A domain model is not a **data model** (which by definition shows persistent data to be stored somewhere)

30) How a domain model is created?

Steps involved in creating a domain model :

- ➢ Find the conceptual classes
- ➢ Draw them as classes in a UML class diagram
- ➢ Add associations and attributes

31) What are the three strategies to find conceptual classes?

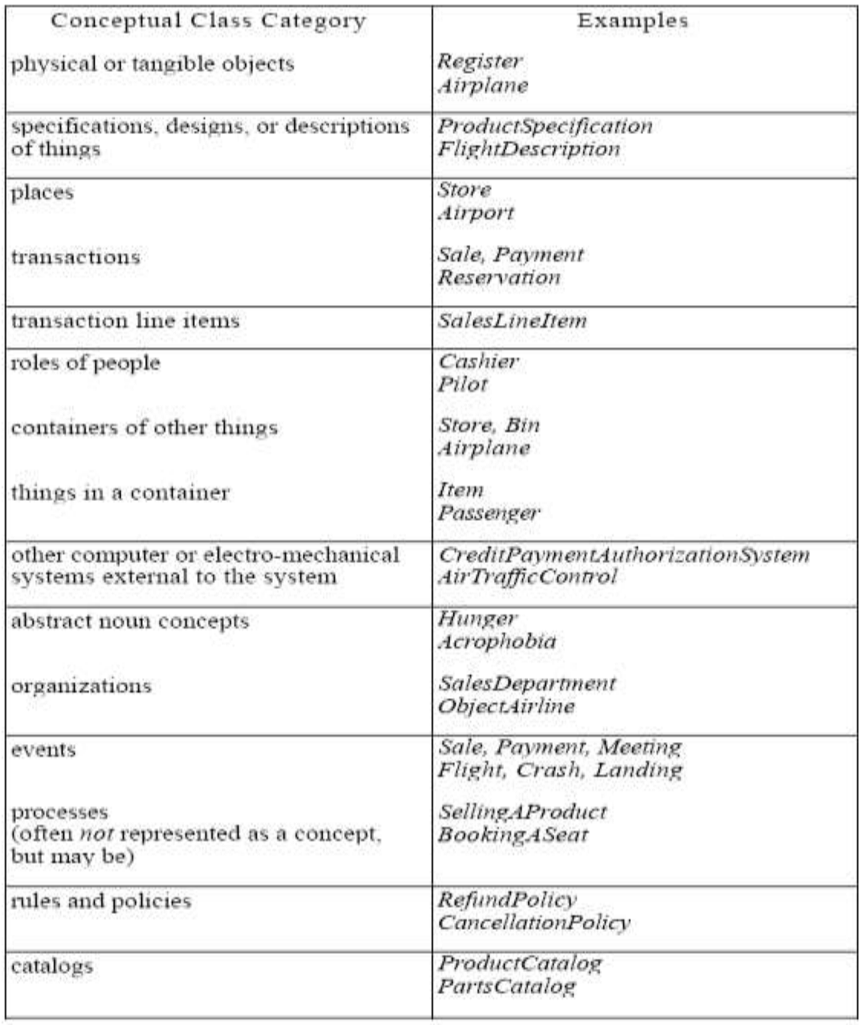
a) Reuse or modify existing models(First,Best,and easiest approach). There are published ,well crafted domain models and data models for many common domains ,such as inventory,finance,health,banking,and so forth.

b) Use a Category List

c) Identify noun phrases.

32) What is Conceptual Class Category List?

We can kick start the creation of a domain model by making a list of candidate conceptual classes. The following table contains many common categories(which are usually worth considering as meeting business information needs)



Conceptual Class Category	Examples
physical or tangible objects	*Register* *Airplane*
specifications, designs, or descriptions of things	*ProductSpecification* *FlightDescription*
places	*Store* *Airport*
transactions	*Sale, Payment* *Reservation*
transaction line items	*SalesLineItem*
roles of people	*Cashier* *Pilot*
containers of other things	*Store, Bin* *Airplane*
things in a container	*Item* *Passenger*
other computer or electro-mechanical systems external to the system	*CreditPaymentAuthorizationSystem* *AirTrafficControl*
abstract noun concepts	*Hunger* *Acrophobia*
organizations	*SalesDepartment* *ObjectAirline*
events	*Sale, Payment, Meeting* *Flight, Crash, Landing*
processes (often *not* represented as a concept, but may be)	*SellingAProduct* *BookingASeat*
rules and policies	*RefundPolicy* *CancellationPolicy*
catalogs	*ProductCatalog* *PartsCatalog*

Conceptual Class Category	Examples
records of finance, work, contracts, legal matters financial instruments and services	*Receipt, Ledger, EmploymentContract MaintenanceLog* *LineOfCredit* *Stock*
manuals, documents, reference papers, books	*DailyPriceChangeList* *RepairManual*

Table 10.1 Conceptual Class Category List.

33) Explain the method of finding conceptual classes using Noun Phrase Identification.

Noun phrase identification is another useful technique which is based on **linguistic analysis.**

- It is based on identifying the **nouns** and **noun phrases** in textual descriptions of a domain, which can be considered as candidate **conceptual classes or attributes**.
- The fully dressed use cases are an excellent description to draw from for this analysis. For example, the current scenario of the *Process Sale* use case can be used.

For example,**Noun phrases** are identified (shown in bold) from **Process Sale Use case** as per the text description below :

Main Success Scenario (or Basic Flow):
1. Customer arrives at a **POS checkout** with **goods** and/or **services** 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.** Price calculated from a set of price rules.
Cashier repeats steps 2-3 until indicates done.
5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment.**
7. Customer pays and System handles payment.
8. System logs the completed **sale** and sends sale and payment information to the external **Accounting** (for accounting and **commissions)** and **Inventory** systems (to update inventory).
9. System presents **receipt.**
10.Customer leaves with receipt and goods (if any).
Extensions (or Alternative Flows):
7a. Paying by cash:
1. Cashier enters the cash **amount tendered.**
2. System presents the **balance due,** and releases the **cash drawer.**
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

34) Explain with an example,the method of finding and drawing conceptual classes.

From the category list and known phrase analysis,a list is generated of candidate conceptual classes for the domain. The list is constrained to the requirements and a simplified version as for iteration-1. As an example the following are identified list of conceptal classes for Process Sale scenario :

Sale	Cashier
CashPayment	Customer
SalesLineItem	Store
Item	ProductDescription
Register	ProductCatalog
Ledger	

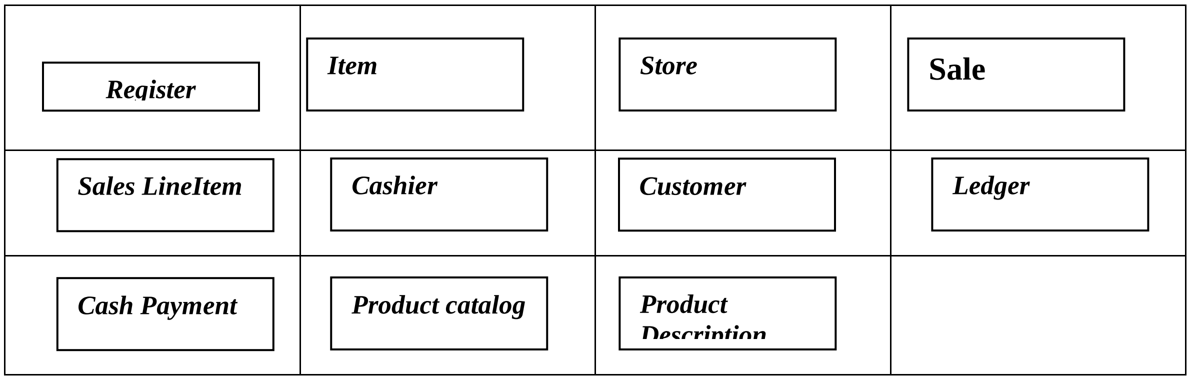


Fig. Initial POS Domain Model

35) What are Description classes? Give examples.

A Description class contains information that descrkibes something else. For example,a ProductDescription that records the price,picture,and text descriptions of an item.This was first named the ***Item-Descriptor pattern.***

The need for description classes

- An item instance represents a physical item in a store;it may have a serial number
- An item has description,price,and itemID
- A Product Description class records information about items
- Even if all inventoried items are sold and corresponding item software instances are deleted,the Product Description still remains.

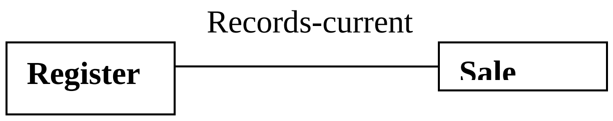
36) **What is Association? Explain with an example**.

An association is a relationship between classes(more precisely,instances of these classes) that indicates some meaningful and interesting connections.

37) Explain Association using UML notation.

UML Definition:

Associations are defined as semantic relationship between two or more classifiers that involve connections among their instances



38) What is multiplicity?

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

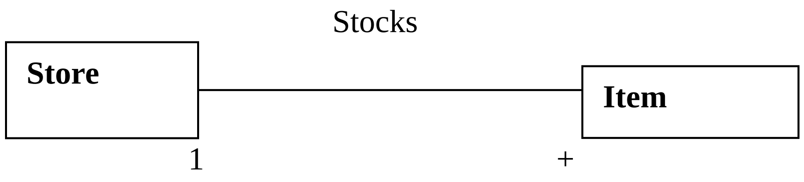


Fig. Multiplicity of an Association

39) Give the method of finding Associations using Common Association List.

Category	Examples
A is a transaction related to another transaction B	CashPayment – Sale Cancellation - Reservation
A is a line item of a transaction B	SalesLineItem - Sale
A is a product or service for a transaction B(or line item)	Item – SalesLineItem (or Sale) Flight - Reservation
A is a role related to a transaction B	Customer – Payment Passenger - Ticket
A is a physical or logical part of B	Drawer – Register Square - Board Seat - AirPlane
A is physically or logically contained in /or B	Register – Store Item-Shelf Square-Board Passenger - Airline
A is a description for B	ProductDescription – Item FlightDescription - Flight
A is known /logged/recorded/reported/captured in B	Sale – Register Piece – Square Reservation - Flightmanifest
A is a member of B	Cashier – Store Player – monnopolyGame Pilot - Airline
A is an organizational Subunit of B	Department – Store

	Maintenance - Airline
A uses or manages or owns B	Cashier – Register Player – Piece Pilot - Airplane
A is nest to B	SalesLineItem – SalesLineItem Square – Square City - City

40) Define an attribute. Explain with an example using UML notation.

An attribute is a logical data value of an object.

Example

Sale needs a dateTime attribute
Store needs a name and address
Cahier needs an ID

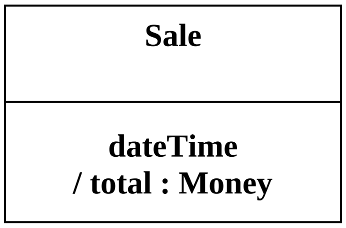
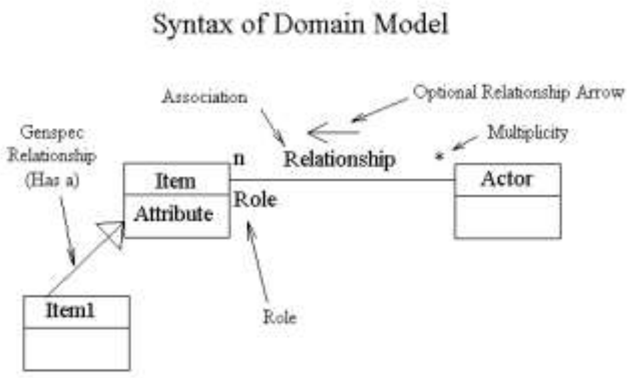
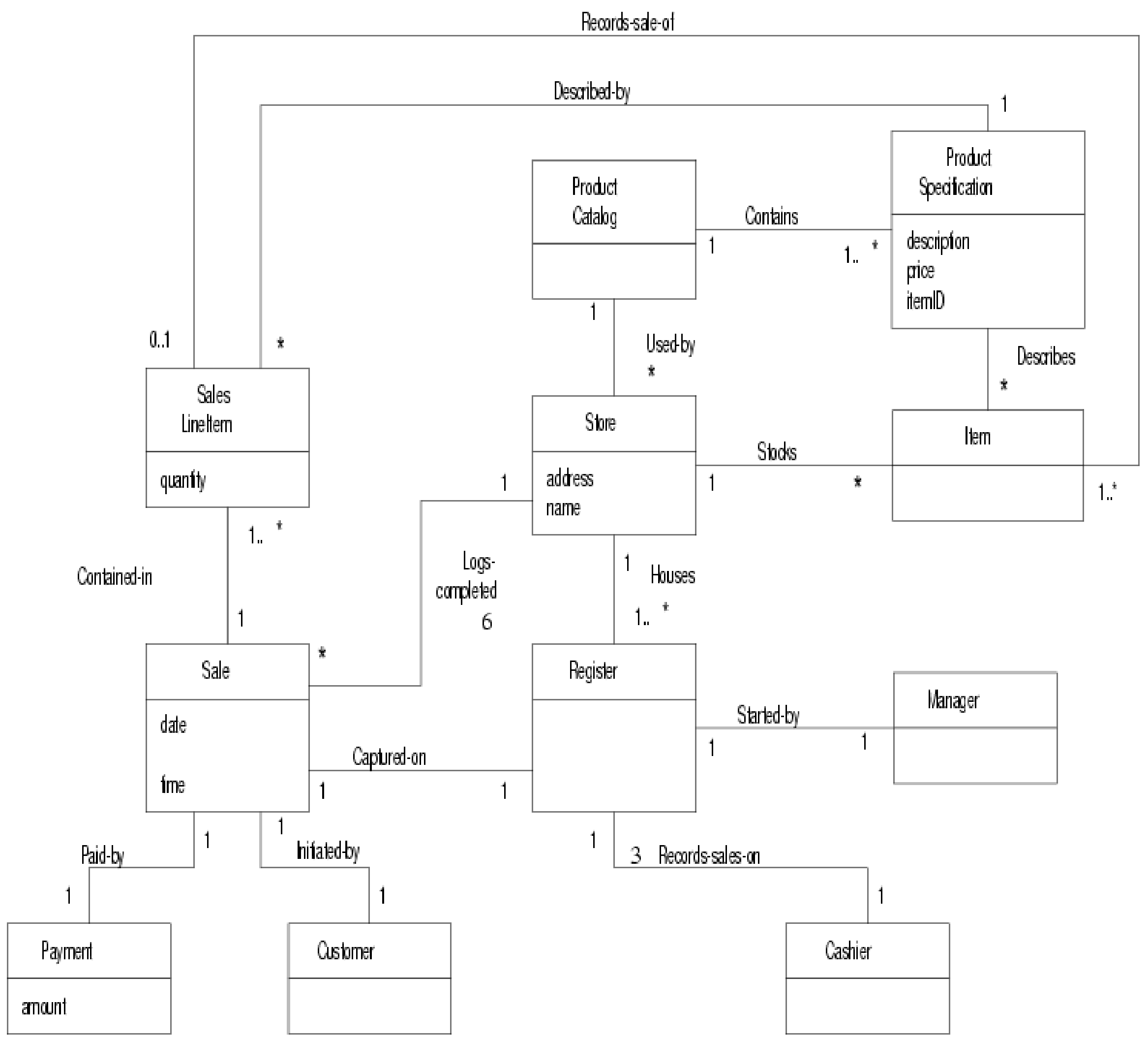


Fig**. Class and attributes**

41) How attributes are used in Domain Models? Explain using examples.





42) How domain model is further refined after the first iteration?

Generalization and specializations are fundamental concepts in domain modeling. Conceptual class hierarchies are often inspiration for Software class hierarchies that exploits inheritance and reduce duplication of code.

Packages are a way to organize large domain models into smaller units.

Domain model is further refined with Generalization,Specialization,Association classes,Time intervals,Composition and packages,usage of subclasses

43) How Domain Model is incrementally developed?

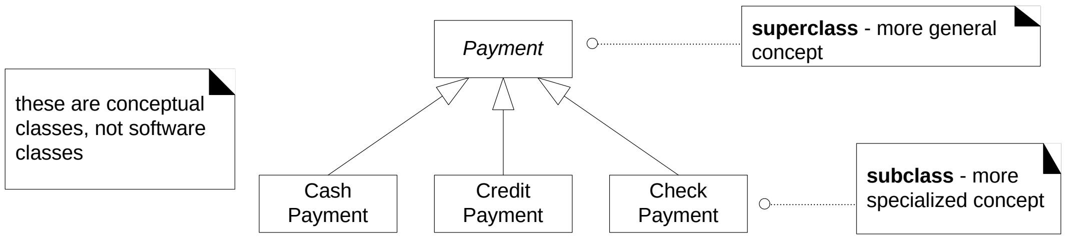
The domain model is incrementally developed by considering concepts in the requirements for this iteration.

44) Explain Generalization-Specialization hierarchy with an example.

Generalization is the activity of identifying commonality among concepts and defining superclass(general concept) and subclass(specialized concept) relationships. It is a way to construct taxonomic classifications among concepts which are illustrated in class hierarchies.

Identifying a superclass and subclasses leads to economy of expression,improved comprehension and a reduction in repeated information.

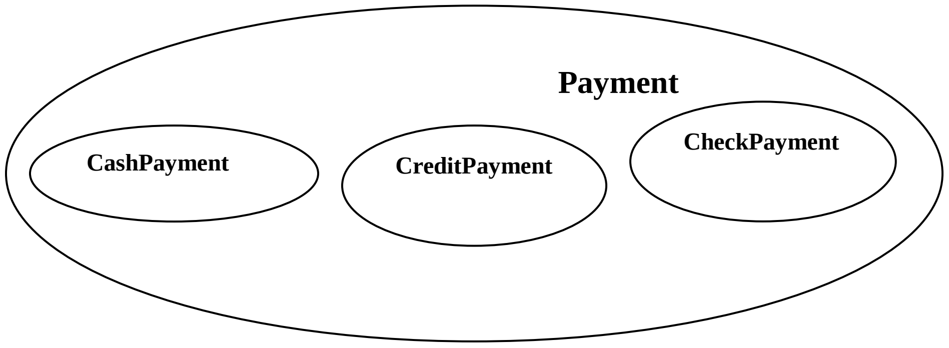
45) Explain Class Hierarchies with an example


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

- Identify superclasses and subclasses when they help us understand concepts in more general, abstract terms and reduce repeated information.
- Expand class hierarchy relevant to the current iteration and show them in the Domain Model.

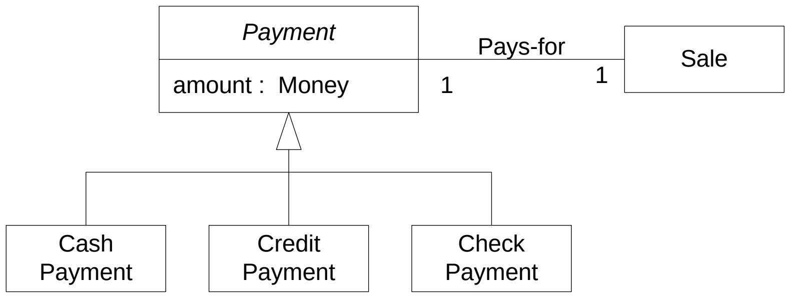
46) How Conceptual subclass and Super Classes are related in terms of set membership? Explain using Venn Diagram.

Conceptual subclasses and superclasses are related interms of set membership. By definition,all members of a conceptual subclass set are members of their superclass set. For example,interms of set membership,all instances of the set *CreditPayment* are also members of the set *Payment.* This is shown in the venn diagram shown below.



47) Explain Subclass Conformance.

When a class hierarchy is created,statements about superclasses that apply to subclasses are made. For example,the following fig. states that all P***ayments*** have an ***amount*** and are associated with a ***Sale***.



All Payment subclasses must **conform** to having an amount and paying for a Sale.

48) What is 100% Rule? Or What is the rule of Conformance to Superclass Definition?

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

1) **attributes**
2) **associations**

49) What are the strong motivation to partition a conceptual class with subclasses?

The following are the strong motivations to partition a 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

50) Give examples of motivations to partition a Conceptual Class into Subclasses

Conceptual Subclass Motivation	Examples
The subclass has additional attribute of interest	Payments – NA Library – Book,Subclass of LoanableResource,has an ISBN attribute
The Subclass has additional associations of interest	Payments – CreditPayment,subclass of Payment,is associated with CreditCard Library –video,subclass of LoanableResource,is associated with Director.
The subclass concept is operated upon,handled,reacted to,or manipulated differently than the super class or other subclasses,in ways that are of interest	Payments – CreditPayment,subclass of payment,is handled differently than than other kinds of payments in how it is authorized. Library-Software,subclass of LoanableResource,requires a depsosit 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 FwemaleHuman with respect to shopping habits.

51) Explain how a Conceptual Super Class is defined and when?

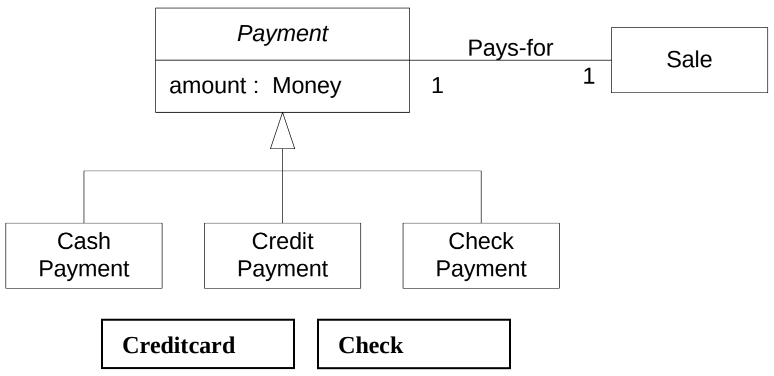
Generalization into a common Superclass is made when commonality is identified among potential subclasses.

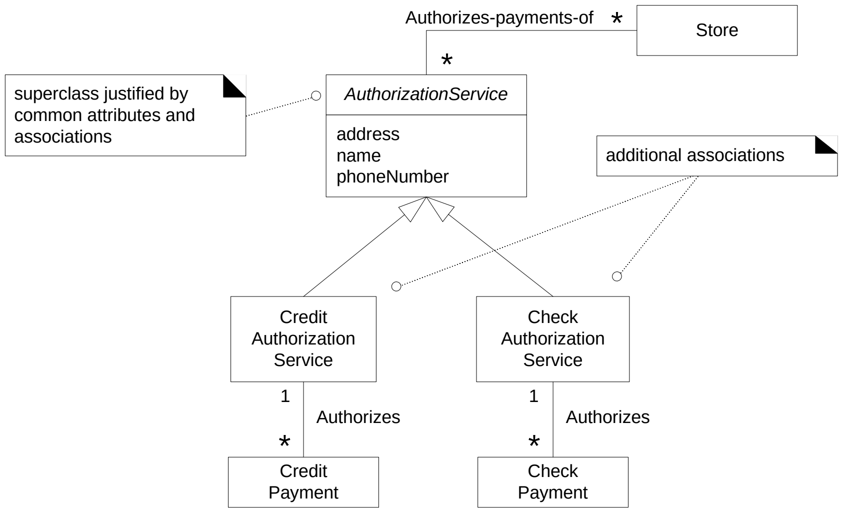
52) What are the guide lines followed in defining a Super Class?

GuideLine
Create a superclass in a generalization relationship to subclasses when :

- The potential conceptual subclasses repreent variations of a similar concept.
- The subclasses will confirm 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 super class.

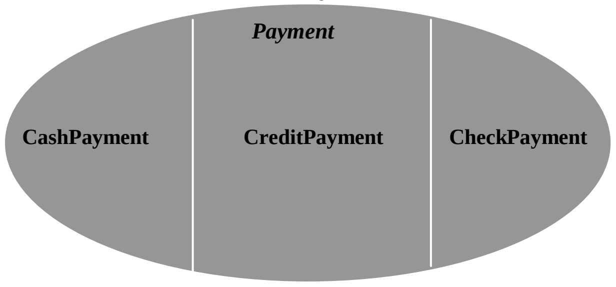
53) Explain in detail an example how a superclass-subclass hierarchies are defined and give justification.





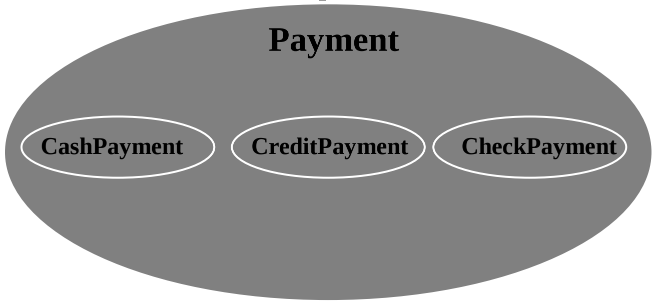
54) What are Abstract Conceptual Classes?
If every member of a class C must also be a member of a subclass C is called an **abstract conceptual class**.

For example,assume that every *Payment* instance must more specifically be an instance of the subclass *CreditPayment,CashPayment,or CheckPayment*. Since every *Payment* member is also a member of a subclass,*Payment* is an abstract conceptual class by definition.



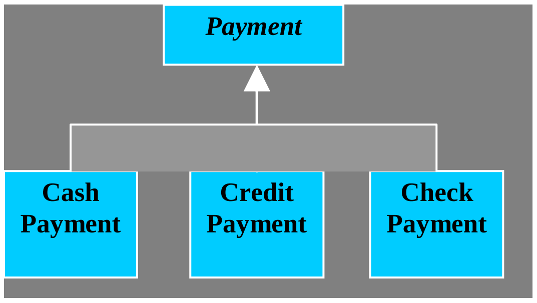
Abstract Classes

- Def.: If every instance of a class C must also be an instance of a subclass, then C is called an abstract conceptual class.

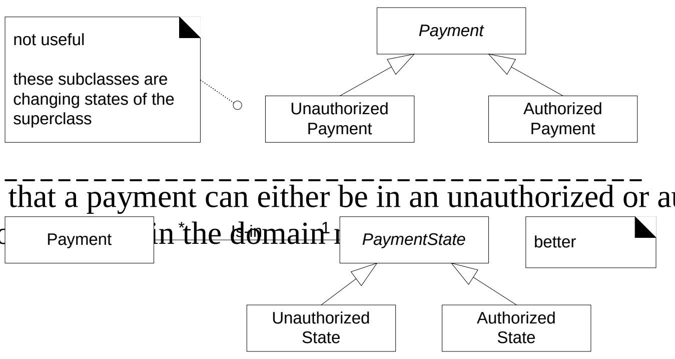


- If a Payment instance exists which is not a member of a subclass, then Payment is not abstract – it is concrete.

Abstract Classes



55) **Explain with an example modeling of changing states.**

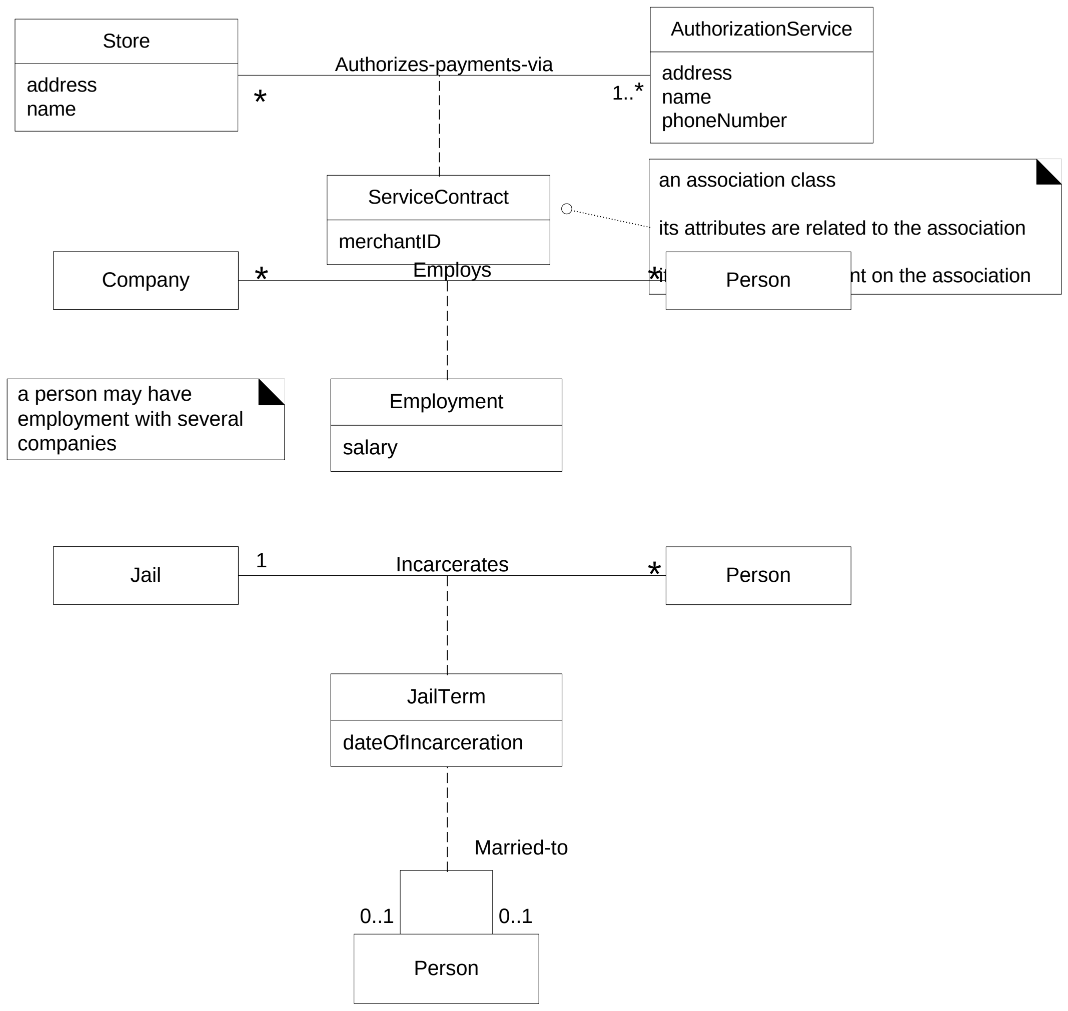


Assume that a payment can either be in an unauthorized or authorized state and it is meaningful to [illegible] in the domain [illegible]
Guideline.

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 in of a concept in the domain model,show the states in the state diagrams instead.

56) Explain with example Assocition classes.



57) Define a) Aggregation b) Composition.

Aggregation is a vague kind of association in the UML that loosely suggests whole-part relationships

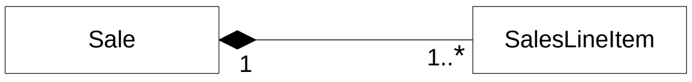
58) What are the guidelines for identifying composition?

Compositions is also known as composite aggregation,is a strong kind of whole-part aggregation and is useful to show in some models. A composition relationship implies that 1) an instance of the part(such as a *Square*) belongs to only one composite instance(such as a *Board*) at a time, 2) the part must always belong to a composite,and 3) the composite is responsible for the creation and deletion of its parts – either by itself creating/deleting the parts,or by collaborating with other objects. Related to this constraint is that if the composite is destroyed,its parts must either be destroyed,or attached to another composite – no free floating Fingers allowed.

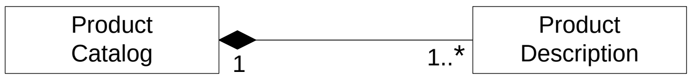
Example

In the POS domain,the *SalesLineItems* may be considered as part of a composite *Sale*; In general,transaction line items are viewed as parts of an aggregate transaction. In addition to conformance to this pattern,there is a create/delete dependency of the line items on the Sale – their lifetime is bound within the lifetime of the *Sale*.

Figure. **Aggregation in the point-of-sale application**



.



59) What are the benefits of showing Composition?

It clarifies the domain constraints regarding the eligible existence of the part independent of the whole. In composite aggregation,the part may not exist outside of the lifetime of the whole.

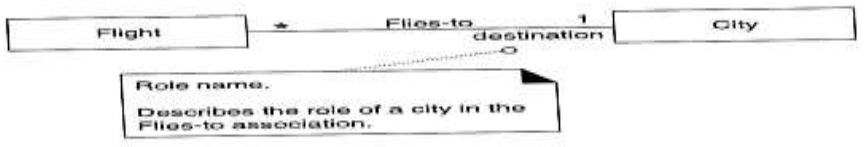
- During design work,this has an impact on the create-delete dependencies between the whole and part software classes and database elements(in terms of referential integrity and cascading delete paths)
- It consists in the identification of a creator(the composite) using the GRASP Creator Pattern.
- Operations – such as copy and delete – applied to the whole often propagate to the parts.

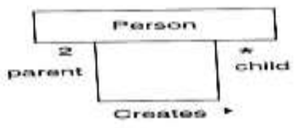
60) What are Association Role Names?

Each end of an association is a role,which has various properties,such as :

1. Name
2. Multiplicity

A role name identifies an end of an association and ideally describes the role played by objects in the association

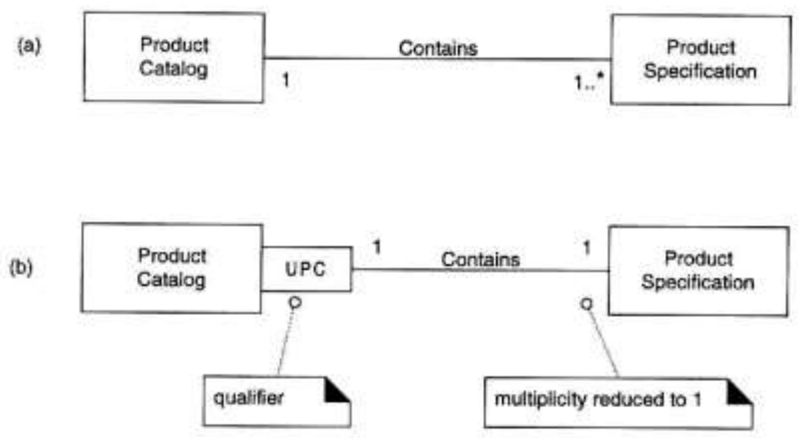




61) What is qualified Association?

A **qualifier** may be used in an association; it distinguishes the set of objects at the far end of the association based on the qualifier value. An association with a qualifier is a **qualified association**.

For example,*ProductDescriptions* may be distinguished in a ProductCatalog by their itemID,as illustrated in the figure below.



- ➢ As contrasted in previous figure (a) vs. (b), qualification reduces the multiplicity at the far end from the qualifier, usually down from many to one.
- ➢ Qualifiers do not usually add compelling useful new information, and we can fall into the trap of “design – think”.
- ➢ However,they can sharpen understanding about the domain.