ICT2622 Object-Oriented Analysis Notes Phase Ch 4 – Summary Notes
SATZINGER JACKSON BURD (6TH EDITION)

CHAPTER 4: Domain Modelling

Data entities or domain classes define the sources for the tables used in a relational database management system.

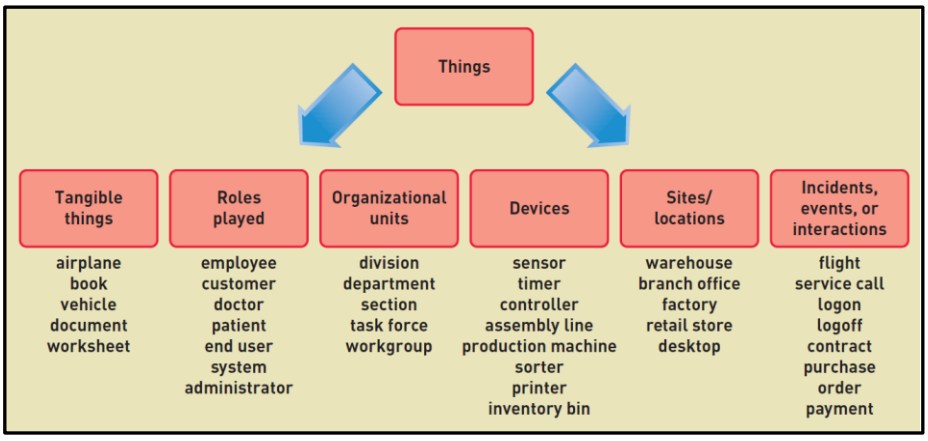
"THINGS" IN THE PROBLEM DOMAIN

- Data entities or domain classes are what end users deal with when they do their work e.g. products, sales, shippers, shipments, and customers.
- These are often referred to as "**things**" in the context of a system's problem domain.
- The problem domain is the specific area of the user's business that is included within the scope of the new system
- The new system involves working with and remembering these "things."
- Things may be related to the people who interact with the system or to other stakeholders.
- A customer is a person who places an order, but the system needs to store information about that customer, so a customer is also a thing in the problem domain.

Two techniques for identifying the important things in the problem domain: **the brainstorming technique** and the **noun technique**:

The Brainstorming Technique - useful for working with users to identify things in the problem domain

- This is a technique to identify problem domain objects in which developers work with users in an open group setting
- An analyst should ask the users to discuss the types of things they work with routinely.
- The analyst can ask about several types of things to help identify them. Many things are tangible and therefore more easily identified, but others are intangible.
- Different types of things are important to different users, so it is important to involve all types of users to help identify problem domain things.
- Types of things: book, vehicle, product in the warehouse. For example, a sale, a shipment, and a return are all important incidents. Sometimes, these incidents are thought of as associations between things
- A sale is an association between a customer and an item of inventory.
- Initially, the analyst might simply list all these as things and then make adjustments as required by different approaches to analysis and design.
- The analyst identifies these types of things by thinking about each use case, talking to users, and asking questions



ICT2622 Object-Oriented Analysis Notes Phase Ch 4 – Summary Notes
SATZINGER JACKSON BURD (6TH EDITION)

CHAPTER 4: Domain Modelling

Steps to follow when using the brainstorming technique:

1. Identify a user and a set of use cases.
2. Brainstorm with the user to identify things involved when carrying out the use case — i.e the things for which the system should capture information
3. Using thing categories systematically ask questions about potential things, such as the following: Are there any tangible things you store information about? Are there any locations involved? Are there roles played by people that you need to remember?
4. Continue to work with all types of users and stakeholders to expand the brainstorming list.
5. Merge the results, eliminate any duplicates, and compile an initial list.

The Noun Technique

- A technique to identify problem domain objects by finding and classifying the nouns in a dialog or description
- Identifying nouns may help identify what should be stored by the system.
- List all the nouns that users mention when talking about the system since nouns used to describe events, use cases, and the actors are potential things.
- Add to the list any additional nouns that appear in information about the existing system or that occur in discussions with stakeholders about the problem domain of the system
- The list of nouns may ultimately be large in size and will require refining.
- The noun technique differs from the brainstorming technique in that the analyst lists all nouns without thinking too much about them and without talking much to users
- Only later will the list be refined based on consultation with stakeholders and users.

Steps to follow when using the noun technique:

1. Identify all nouns using the use cases, actors, inputs and outputs and other information about the system
2. Using other information from existing systems, current procedures, and current reports or forms, add items or categories of information needed
3. As this list of nouns builds, it will require refining.

    - Ask these questions about each noun to assess whether it ought to be included:
        - Is it a unique thing the system needs to know about?
        - Is it inside the scope of the system I am working on?
        - Does the system need to remember more than one of these items?

    - Ask these questions about each noun to assess whether it ought to be excluded:

        - Is it a synonym for another thing that has been identified?
        - Is it simply an output of the system produced from other identified information?
        - Is it simply an input that results in recording other identified information?

    - Ask these questions about each noun to assess whether it should be researched:

        - Is it likely to be a specific piece of information (attribute) about some other thing identified?
        - Is it something I might need if assumptions change?

4. Create a master list of all nouns identified and then note whether each should be included, excluded, or researched further.

CHAPTER 4: Domain Modelling

5. Review the list with users, stakeholders, and team members and then refine the list of things in the problem domain.

Attributes of Things

- The noun technique involves listing all the nouns that come up in discussions or documents about the requirements.
- Many of these nouns are actually attributes.
- Most information systems store and use specific pieces of information about each thing called **attributes**
- The analyst needs to identify the attributes of each thing that the system is required to store
- One attribute may be used to identify a specific thing, such as an ID number for an employee or an order number for a purchase.
- The attribute that uniquely identifies the thing is called an **identifier** or **key**
- Sometimes, the identifier is already established (an ID number, vehicle ID number, or product ID number).
- Sometimes the system needs to assign a specific identifier via a compound attribute.
- **Compound attribute** - an attribute that consists of multiple pieces of information but is best treated in the aggregate (e.g. full name)

Associations Among Things

- After recording and refining the list of things and determining potential attributes, the analyst needs to research and record additional information.
- Many important relationships among things are important to the system.
- An association is a naturally occurring relationship between specific things

- **Association (aka relationship)** - the UML term that describes a naturally occurring relationship between specific things
- Associations between things apply in two directions e.g. a customer places an order describes the association in one direction. Similarly, an order is placed by a customer (the other direction)
- Sometimes it might seem more important for the system to record the association in one direction than in the other

- **Cardinality of the association** - the number of links that occur in an association. Cardinality can be one-to-one or one-to-many. The term multiplicity is used to refer to the number of links in UML and should be used when discussing UML models. Multiplicity is established for each direction of the association.

The relationship for a customer placing an order can have a range of zero, one, or more, usually indicated as zero or more. The zero is the minimum multiplicity, and more is the maximum multiplicity. These terms are referred to as **multiplicity constraints**.

- **Multiplicity constraints** - the actual numeric count of the constraints on objects allowed in an association
- In some cases, at least one association is required (a mandatory as opposed to optional association, where zero multiplicity is a possibility).

binary associations - associations between exactly two distinct types of things
unary association - an association between two instances of the same type of things
ternary association - an association between exactly three distinct types of things
n-ary association - an association between n distinct types of things

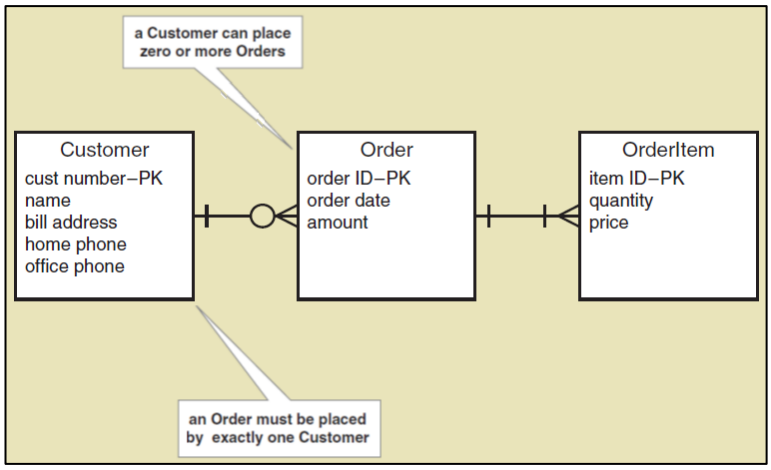
ICT2622 Object-Oriented Analysis Notes Phase Ch 4 – Summary Notes
SATZINGER JACKSON BURD (6TH EDITION)

CHAPTER 4: Domain Modelling

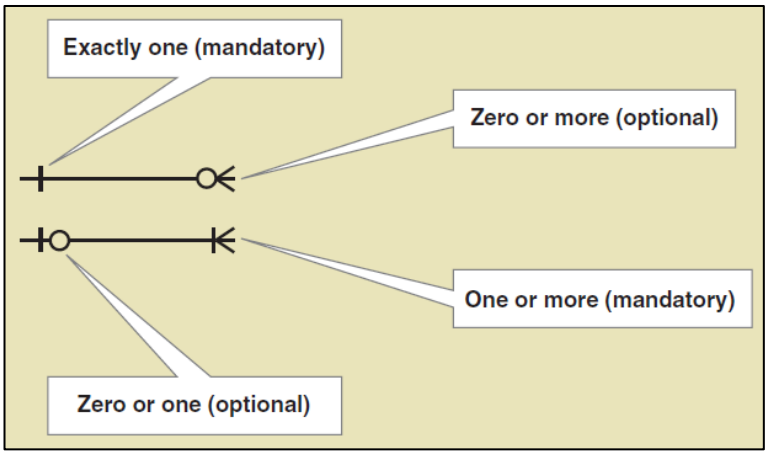
- Storing information about the associations is just as important as storing information about the specific things. It is important to have information like the name and address of each customer, but it is equally important to know what items each customer has ordered.

THE ENTITY RELATIONSHIP DIAGRAM

- **Data entities** - sets of things about which the system needs to store information. Used in ERDs.
- Data storage requirements include the data entities, their attributes, and the associations among the data entities
- On the ERD rectangles represent data entities, and the lines connecting the rectangles show the relationships (associations) among data entities.



A simple ERD – ERD 1

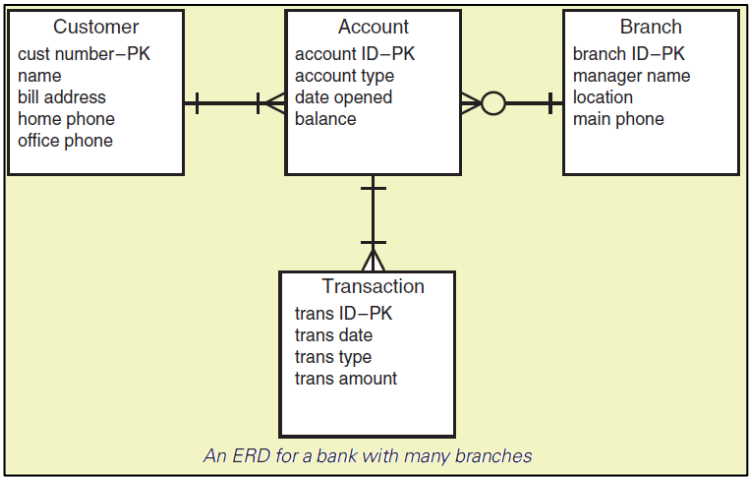


Cardinality Symbols – Crow's Foot method

- The Crow's Foot notation can express precise details about the system.
- The constraints reflect the business policies that management has defined, and the analyst must discover what these policies are.
- The analyst does not determine that two customers cannot share one order; management does.
- The attributes of the data entity are listed below the name, with the key identifier listed first, usually followed by "PK" to indicate primary key.

CHAPTER 4: Domain Modelling

Eg ERD1 - Each order contains a minimum of one and a maximum of many items (there could not be an order if it did not contain at least one item)



An ERD for a bank with many branches

This ERD is for a bank that has many branches. Each branch has one or more accounts. Each account is owned by one customer and results in one or more transactions. There are a few other issues to consider in the bank example. First, there is no data entity named Bank. That is because the ERD shows data storage requirements for the bank. There is only one bank. Therefore, there is no need to include Bank in the model. This is a general rule that applies to ERDs. If the system were for state bank regulators, then Bank would be an important data entity because there are lots of banks under the state regulators' jurisdiction.

A customer must have at least one account. The bank would not add a customer unless he or she were adding an account. Note also that the branch can have zero accounts. A branch might be added long before it opens its doors, so it is possible that it is does not have any accounts. Additionally, there might be some branches that do not have accounts, such as a kiosk at a university or airport. Note that an account must have at least one transaction. The rationale is that opening a new account requires an initial deposit, which is a transaction. Questions about the cardinality and minimum and maximum cardinality constraints must be discussed and reviewed with stakeholders.

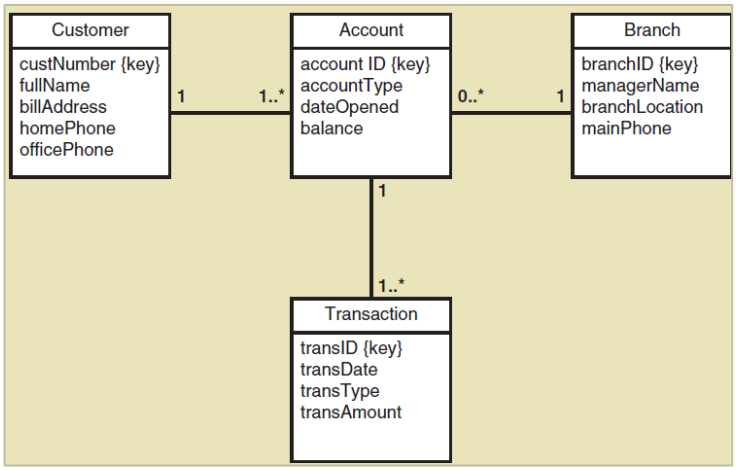
THE DOMAIN MODEL CLASS DIAGRAM

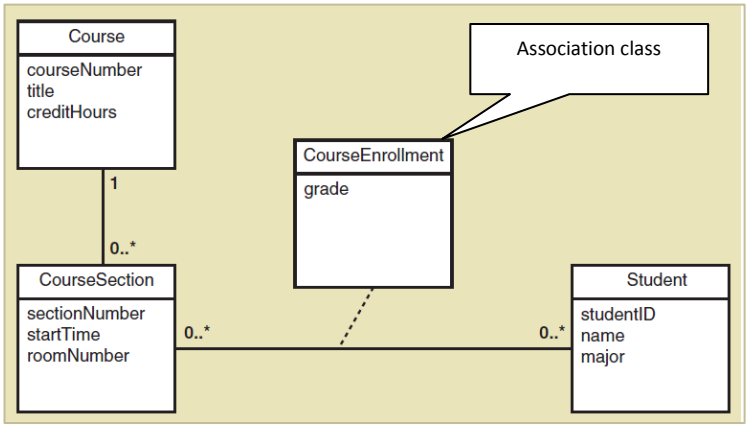
- Many current approaches to system development use the term class rather than data entity and use concepts and notations based on UML to model the things in the problem domain. These concepts come from the object-oriented approach to systems.
- A class is a category or classification used to describe a collection of objects. Each object belongs to a class. Therefore, students Mary, Joe, and Maria belong to the class Student.
- Classes that describe things in the problem domain are called domain classes. D
- Domain classes have attributes and associations.
- Multiplicity (called cardinality in an ERD) applies among classes.
- Initially, when defining requirements, the approach to modeling using an ERD or UML is very similar
- Class Diagram - a diagram consisting of classes (i.e., sets of objects) and associations among the classes
- Domain Model Class Diagram - a class diagram that only includes classes from the problem domain

- The UML class diagram is used to show classes of objects for a system.
- One type of UML class diagram that shows the things in the users' problem domain is called the domain model class diagram.
- Another type of UML class diagram is called the design class diagram, and it is used when designing software classes.

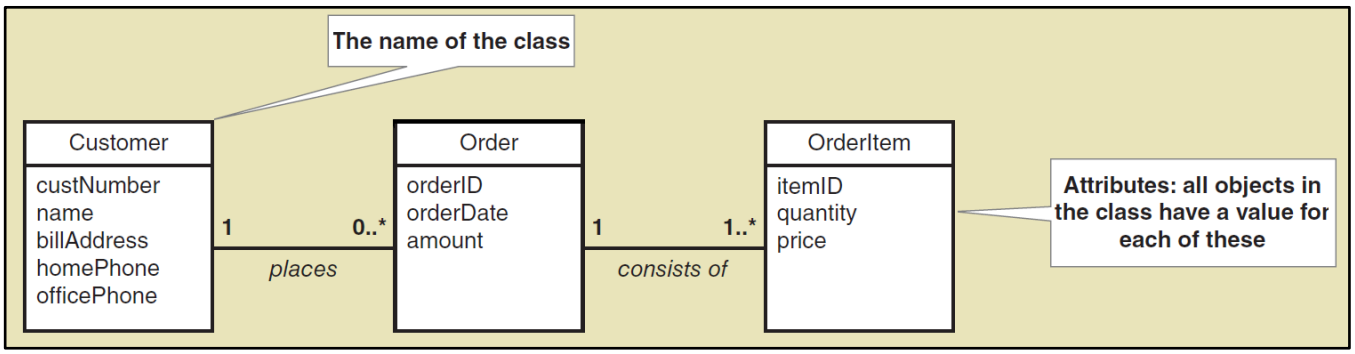
CHAPTER 4: Domain Modelling

- On a class diagram, rectangles represent classes, and the lines connecting the rectangles show the associations among classes.
- The domain class symbol is a rectangle with two sections. The top section contains the name of the class, and the bottom section lists the attributes of the class.
- The design class symbol includes a third section at the bottom for listing methods of the class - methods do not apply to problem domain classes.
- Class names and attribute names use camelback notation, in which the words run together without a space or underscore. Class names begin with a capital letter; attribute names begin with a lowercase letter





A refined university course enrolment domain model class diagram with an association class



A simple UML domain model class diagram with name and attributes

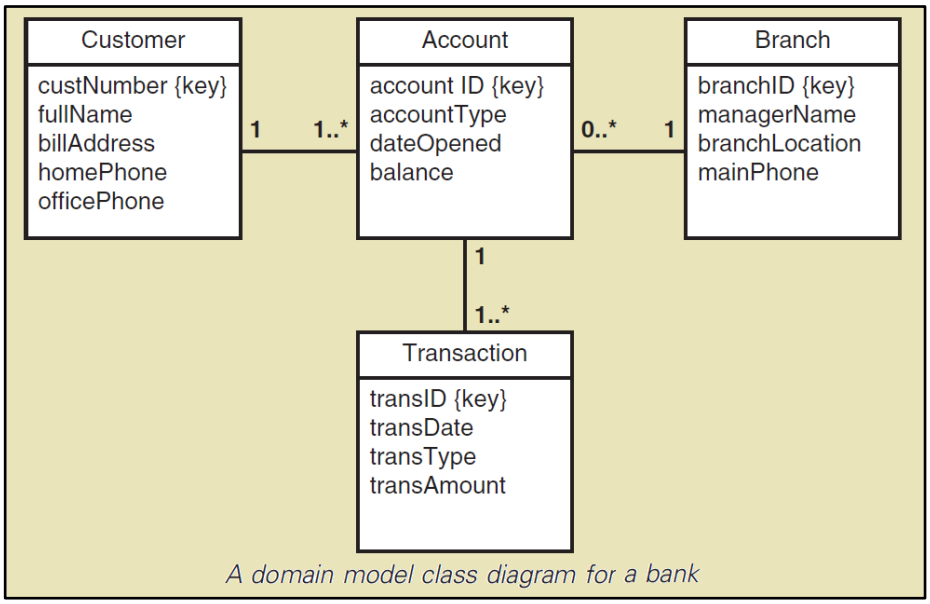
ICT2622 Object-Oriented Analysis Notes Phase Ch 4 – Summary Notes
SATZINGER JACKSON BURD (6TH EDITION)

CHAPTER 4: Domain Modelling

Domain Model Class Diagram Notation

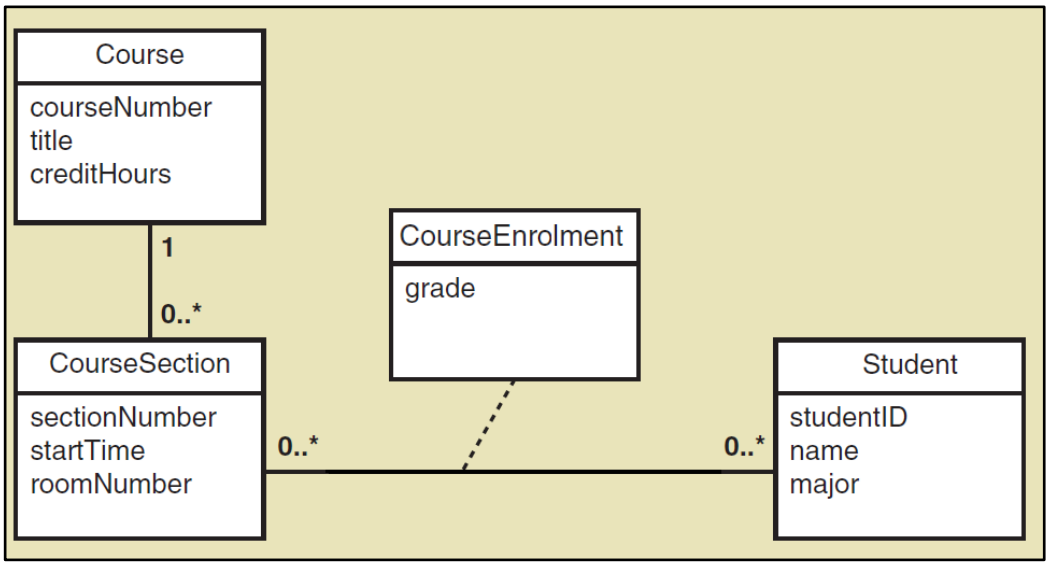
The simplified domain model class diagram shown above has three classes:
Customer, **Order**, and **OrderItem** (just like the example of an ERD shown in Figure 4-9). Here, each class symbol includes two sections.

In diagram notation, note that each Customer can place many Orders (a minimum of zero and a maximum of many) and that each Order is placed by one Customer. The associations **places** and **consists** of can be included on the diagram for clarity, but this detail is optional. The multiplicity is one-to-many in one direction and one-to-one in the other direction. The multiplicity notation, shown as an asterisk on the line next to the Order class, indicates many orders. The other association shows that an Order consists of one or more OrderItems, and each OrderItem is associated with one Order.



A domain model class diagram for a bank

A domain model class diagram, such as the one for the bank with multiple branches that was discussed earlier and shown as an ERD shows above. In this example, the UML notation for indicating an attribute that is an identifier or key is **{key}**.



A refined university course enrolment domain model class diagram with an association class

CHAPTER 4: Domain Modelling

The previous figure is an example of a domain model class diagram with a many-to-many association.

- At a university, courses are offered as course sections, and a student enrols in many course sections.
- Each course section contains many students.
- Association between **CourseSection** and Student is therefore many-to-many.
- Many-to-many associations involve additional data that are important and must be stored.
- The grade that each student receives for the course is important data
- A domain class **CourseEnrolment** is created to represent the association between student and course section; this is called an *association class*.
- Association class **CourseEnrolment** stores the attribute **grade**.
- A dashed line connects the association class with the association line between the **CourseSection** and **Student** classes.

Reading the association in the figure from left to right:

- One course section has many course enrolments—each with its own grade—and each course enrolment applies to one specific student.

Reading from right to left:

- One student has many course enrolments—each with its own grade—and each course enrolment applies to one specific course section.

A database implemented by using this model will be able to produce grade lists showing all students and their grades in each course section as well as grade transcripts showing all grades earned by each student.

More Complex Issues about Classes of Objects

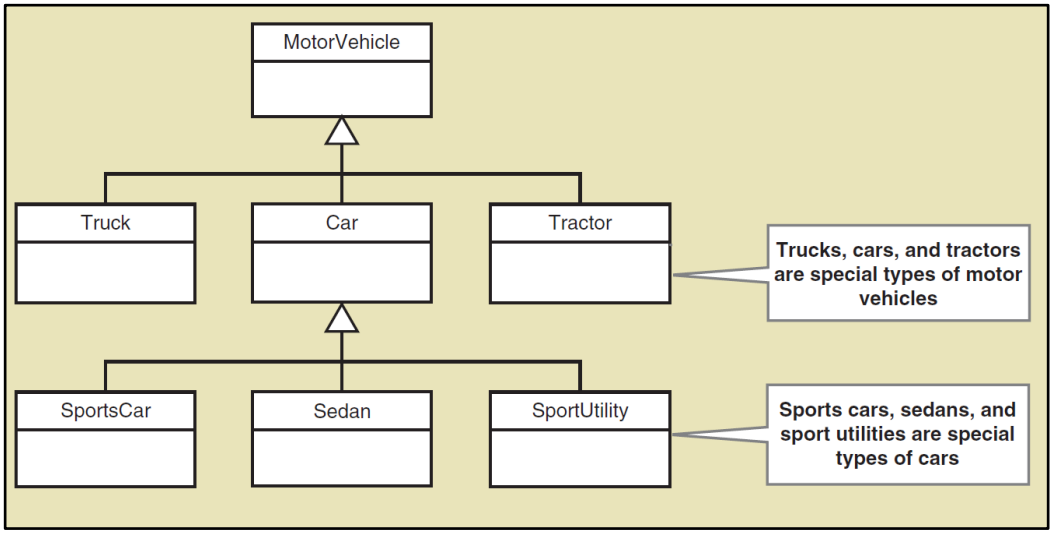
- In UML, an association is one of many types of relationships, so we need to be more precise when discussing UML diagrams than when discussing ERDs
- With class diagrams, there are three types of relationships among classes of objects:
    - ***Association relationships*** - association is a naturally occurring relationship between specific things
    - ***Generalisation / specialisation relationships***
    - ***Whole / Part relationships***

Generalisation / specialisation relationships (aka ***inheritance relationships***)

- Classes are classified in terms of similarities and differences (i.e. in terms of an "inheritance hierarchy")
- Generalizations - judgments that group similar types of things e.g. motor vehicles (main or general class): cars, trucks, and tractors (more specific classes, subclasses of motor vehicles)
- Specialisations - judgments that group different types of things e.g. special types of cars - SUVs, sports cars, sedans. There is similarity at parent level called SUPERCLASS, but a uniqueness at child level, called SUBCLASS.
- The subclass “inherits” characteristics of the superclass, but has its own special characteristics that distinguish itself from other subclasses.
- In the object-oriented approach, inheritance is a key concept that is possible because of generalization/specialization hierarchies.

ICT2622 Object-Oriented Analysis Notes Phase Ch 4 – Summary Notes
SATZINGER JACKSON BURD (6TH EDITION)

CHAPTER 4: Domain Modelling



Abstract class - a class that describes a category or set of objects but that never includes individual objects or instances.

- Is a class that exists so subclasses can inherit from it - there is never an actual object
- Class name or title is shown in *italics* to designate Abstract Class

Concrete class - a class that allows individual objects or instances to exist

Whole-Part Relationships

Whole-part relationships are used to show an association between one class and other classes that are parts of that class - one class is a part or a component portion of another class

- Two types of whole-part relationships:
  - Aggregation
  - Composition

Aggregation - refers to a type of whole-part relationship between the aggregate (whole) and its components (parts), where the component parts can also exist as individual objects apart from the aggregate

- Aggregation relationship is represented in UML as an unfilled diamond symbol

Composition refers to whole-part relationships where the parts, once associated, can no longer exist separately

- Composition relationship is represented in UML as an filled in diamond symbol

- Aggregation and Composition whole-part relationships mainly allow the analyst to express subtle distinctions about associations among classes
- As with any association relationship, multiplicity can apply.

- A domain model class diagram for an information system evolves as the project proceeds and unlike the use case diagrams, where many diagrams are created, there is eventually only one domain model class diagram.
- Unlike the use case diagram, the domain model class diagram is not produced just for presentations.
- The process of developing and refining the domain model class diagram is how the analyst explores and learns about the problem domain.
- Therefore, the information depicted in the domain model class diagram must be very detailed and rich in specific meaning.