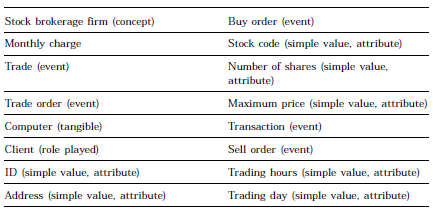
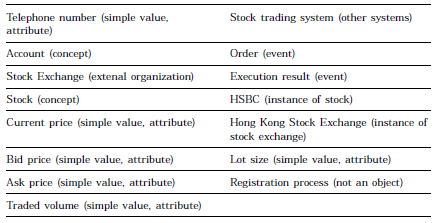
Step 1: **Identifying Objects and Classes**

**The following list shows the candidate classes:**

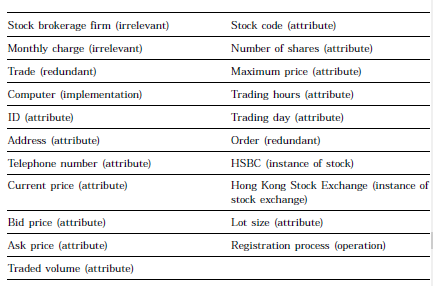




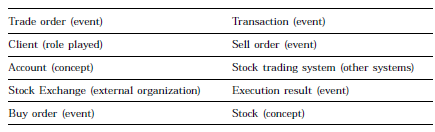
Step 2: Remove specific classes

redundant classes, irrelevant classes, vague classes, attributes, operations, roles, and implementation constructs are removed

The classes that may be removed are:

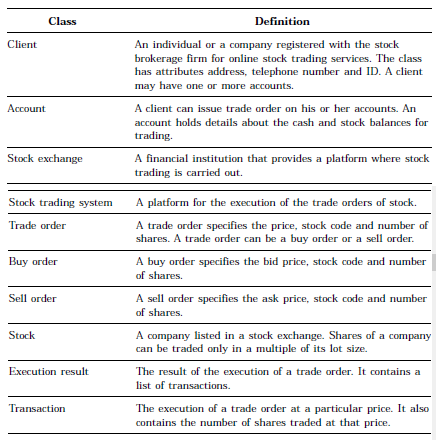


Final relevant classes:



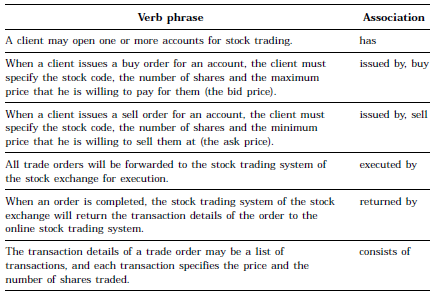
Step 3: **Developing Data Dictionary**

After the candidate classes have been consolidated, prepare a data dictionary to record the definition of classes.



Step 4: **Identifying Associations between Classes**

For example, in the statement “a client may *open* one or more *accounts* for stock trading” [emphasis added] contains the verb “open” which links the client and the account. The association between the client and the *account* may be named as *has* since it is an ownership relationship. The association can also be named as *opened by* to reflect the action performed by the *client*



From the domain knowledge, we have the following associations:

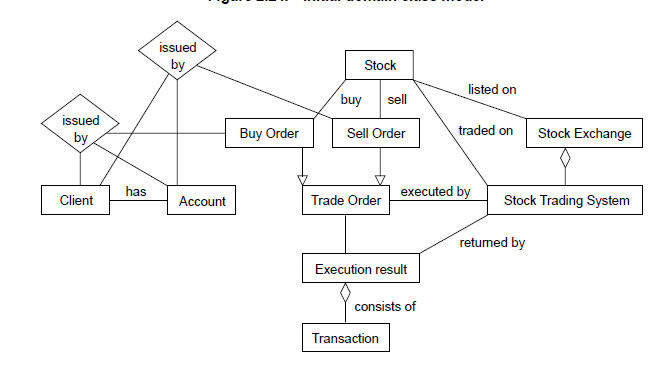
• A stock is listed on a stock exchange

• A stock is traded on a stock trading system of a stock exchange

• The result of a trade order is a list of transactions

• A stock exchange has one or more stock trading systems

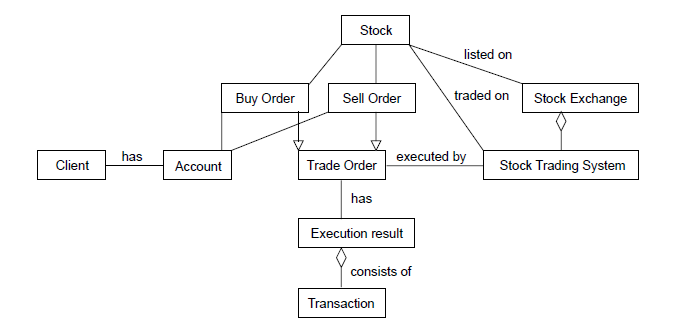
Based on these associations, we may draw:



The above diagram has:

1. Ternary relationships i.e. relation in three classes
2. Some actions
3. Irrelevant association

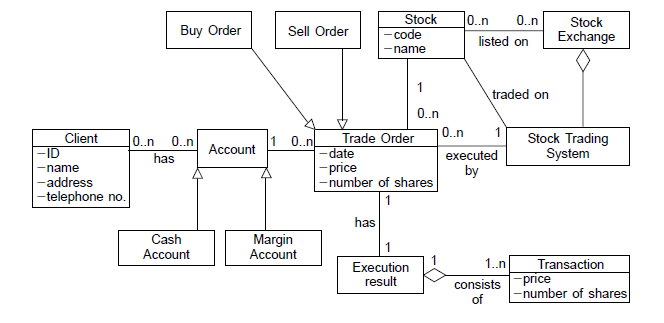
We remove them and get the diagram below:



Step 5: **Identifying Attributes of Classes and Association Classes**

Define attributes

Define inheritance relations : the buy order and sell order are the types of trade order



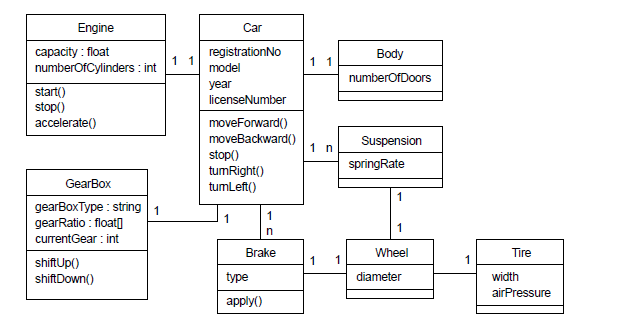
Some tips for the modelling:

* Set Focus and Context of Diagram
* Use Appropriate Names for Classes
* Organize Diagram Elements
* Annotate Diagram Elements
* Refine Structural Model Iteratively and Incrementally
* Show Only Relevant Associations

Structural models

Example 1:

A car consists of different structural components such as the engine, body, suspension, gearbox, etc. Each component in turn contains its own attributes and operations. For example, the engine has its capacity, and it can be started or stopped.



Example 2:

In this simple sales order system example, there are three methods of payment: cash, credit card or check. These three payment methods have the same attribute (*amount*), but they have their own individual behaviors and attributes. Figure 2.21 shows a structural model of this simple sales order system in a class diagram. The directional associations in the diagram indicate the direction of navigation from one class to another. For example, the *Order*

class can access information from the *Payment* class, but not the other way round.

