CPS122 Lecture: Associations

Last revised January 26, 2022

Objectives:

1. To introduce the association relationship between objects
2. To introduce navigability and multiplicity of associations
3. To show how multiplicity implies that a given association is either optional or mandatory
4. To introduce the idea of qualified associations
5. To introduce the notion of roles of objects participating in an association
6. To introduce the use of associations to represent aggregation and composition
7. To introduce association names and associations classes

Materials:

1. Projectables of various class diagrams illustrating concepts

I. **Introduction**

A. An earlier class dealt with initial identification of the key classes comprising a system - an analysis task. At this point, we begin to explore overall class structure, which will ultimately be represented by a class diagram, that will continue to be refined as system development proceeds.

1. The point of constructing a class diagram is that it forces us to <u>think</u> about certain key issues, and then to represent our thinking in a pictorial way that guides further design.

2. I will later demonstrate using astah to create class diagrams - but you can use another tool (or hand drawing) if you prefer.

B. In the spirit of "seamless development" that characterizes OO, the initial development of a class structure is an analysis task, which is refined as part of design.

C. We need to do more than just identify the classes that are implied by a given problem - we also need to consider the relationships between those classes.

D. At the outset, we note that there are two different sorts of relationship, that we handle similarly but need to keep distinct in our thinking.

1. There are relationships between *individual objects*. Such a relationship describes how a particular object of one class relates to a particular object of another class.

    a) Among humans, the relationship known as marriage is such a relationship. It relates one individual to another specific individual. You may know many married people, but each has a different spouse. We call something like this an association.

    b) It is also possible to have a "part of" relationship between objects, which we call an aggregation or composition. From the standpoint of representation, we will treat this a s special kind of association - so we will sometimes just use the word "association" to encompass both simple association and aggregation/composition.

    c) In either case, then, each individual object participates in the relationship (or doesn't participate in the relationship, as the case may be) with its own particular partner or partners.

    d) Where things get a bit confusing is that when we identify an individual relationship between objects, we are also identifying a relationship between the corresponding classes. The fact that an object of class Book is related to one or more objects of class Author implies that there is a relationship between the *classes* Book and Author such that a member of the one class can participate in this relationship with a member of the other class.

2. There are relationships *between classes*. Such a relationship describes how one whole class of objects is related to another class.

    a) Among humans, the fact that all CS majors are also students is such a relationship.

b) In the OO world, generalization, or inheritance, is such a relationship.

c) In the case of a class relationship, all the objects that belong to a given class participate in the relationship in the same way.

3. Today we focus on relationships between objects - associations (including aggregations/compositions). We will say a lot more about relationships between classes when we talk about class diagrams.

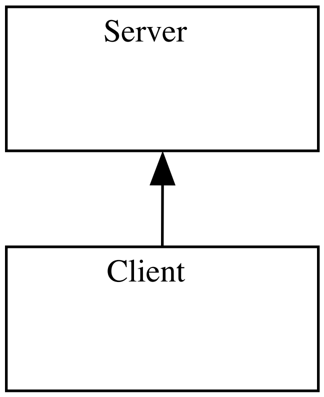
II. **Properties of Associations (and Aggregations and Compositions)**

A. Associations have numerous properties which governs how they are used and how they are represented in a programming language like Java. We will discuss a few key properties now, and will discuss them in more depth when we talk about class diagrams.

1. Navigability (directionality):

a) Ordinarily, associations are conceived of as being bidirectional - e.g. in the association between a Book and its Author(s), we might want to go from a Book object to its Author object(s), and likewise to go from an Author object to the Book(s) it is the author of.

b) Sometimes, though, an association is conceptually unidirectional - e.g. if were to try to depict the relationship between a Server system and a Client system that uses it, we might draw it this way:



PROJECT

The arrow says that the Client must know about the Server, but the Server does not need to know about the Client (except briefly, during the time it is responding to a message received from the Client.) Thus, the Client object can tell us what Server(s) it is using, but the Server object need not know which Clients are using it.

For example: this is way the web is set up - which is why cookies are used to allow servers to know something about their clients.

c) Why would we want to identify an association as being unidirectional where this is appropriate is? The presence of an association in the class diagram implies that the implementation will need to maintain information about this association. Keeping information about a bidirectional association means that both objects will have to maintain information about the association. If this is not necessary, maintaining the association in only one direction will simplify the implementation.

d) Think-pair-share activity: What navigability is appropriate for each of the following pairs of classes:

(1) Musician and Album

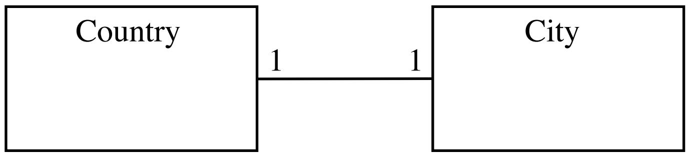
(2) Grocery Store and Loaf of Bread

(3) Bookstore and Book

PROJECT

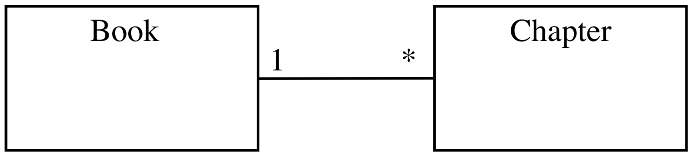
2. Multiplicity: Some associations are conceptually one to one - one object of a given type relates to one object of another type. Others allow one object of a given type to be related to many objects of another type.

   a) One-to-one. Example: relationship between a country and its capital city.



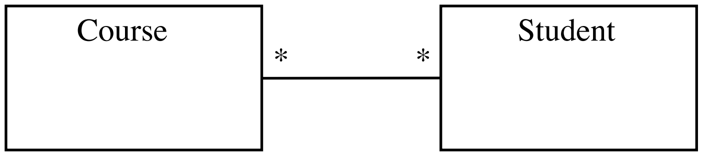
   PROJECT

   b) One-to-many: Example: the relationship between a book and the individual chapters that are part of it.



   PROJECT

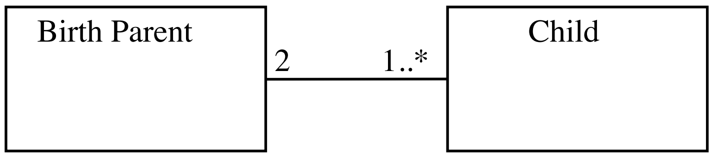
   c) Many-to-many: Example: students and courses



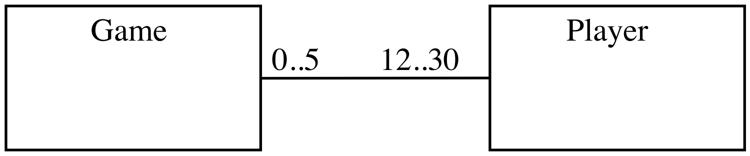
   PROJECT

   d) Often, the multiplicities will be expressed as *ranges*, rather than as simple values

(1)Example: a person has exactly two birth parents. A parent has at least one child (else he or she is not a parent!), but can have any number:



(2)Example: the annual volleyball competition between the Math and CS wings of our department involves up to 5 games. In each game, at least 12 but no more than 30 students can participate.



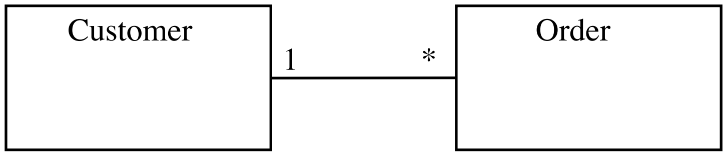
PROJECT

(This one's a bit contrived to illustrate a point, I admit :-).

(3)The symbol * we have previously used means "0 or more" - hence it is equivalent to 0..*

e) If the lower limit on the multiplicity of a certain relationship is 0, we say that the relationship is *optional*. If the lower limit is greater than 0, we say that the relationship is *mandatory*. Note that the same relationship may be optional in one direction, and mandatory in the other.

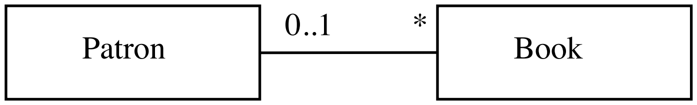
(1)Example: the relationship between a customer and the orders he/she has placed with a company. Assuming a person can register as a customer before placing an order, we have the following scenario:



PROJECT

The relationship from an order to a customer is mandatory - every order *must* be associated with a customer. The relationship from customers to orders is *optional* - a customer does not need to have any orders.

(2) It's certainly possible to have a relationship that's optional both ways - e.g. the relationship between a library patron and books. he/she currently has checked out. A patron does not have to have any books checked out at a given time, nor does any particular book have to be checked out at a given time. (Note that while we allow a patron to have any number of books out, a book can only be checked out to one patron at a time.)



PROJECT

(3) Recall that the notation "*" is short for "0..*", and so stands for a relationship that is inherently optional. If the relationship is mandatory, but of unlimited multiplicity, we must use the form "1..*".

(4) Also note that some writers use the notation "n" instead of * in a range - so * (= 0..*) is written as "0..n" and 1..* is written as "1..n".

f) Think-pair-share activity: What multiplicity is appropriate for each of the following pairs of classes (consider both ends)

(1) Soccer Team and Player

(2) Employer and Employee

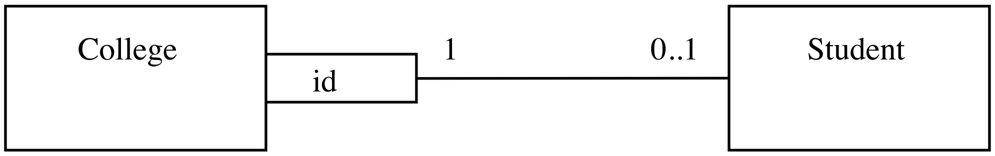
(3)Husband and Wife (Tricky: must be 1:1, not 0..1: 0..1
because not a husband or wife unless married!

PROJECT

3. Qualified Association: Sometimes, a given object can be associated with many objects of some other class, but there is some <u>qualifier</u> such that, for any given value of the qualifier, the object is associated with at most one other object.

   EXAMPLE:

   A college is associated with many students; but for any given student id, there is at most one associated student (or possibly none). We say that the association between the college and students is a <u>qualified association</u>, with student id as the qualifier. This can be depicted as follows:

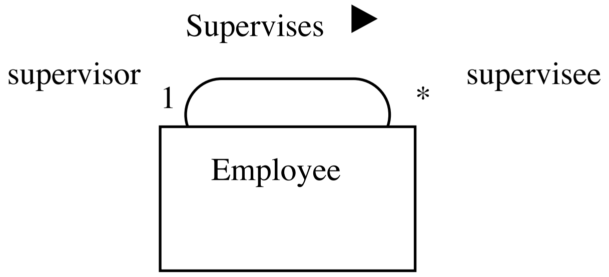


   PROJECT

   (Note how the effect of the qualification is to reduce the multiplicity from 1 : n to 1 : 0..1 - for any given id value, there is at most one matching student)

4. Association roles: Sometimes an association both ends of an association might go to different objects of the same class.

   For example: consider the relationship between an Employee and their supervisor - who is also an Employee (though not the same person), This might be handled as follows:



PROJECT

5. Aggregation/Composition: As we noted earlier, aggegation/ composition is actually a form of association in which it is the case that one of the objects can be thought of as being *part of* the other - i.e. the relationship is one between a whole and its constituent parts..

   a) Aggregation is appropriate when we can meaningfully use the phrase "is a part of" to describe the relationship between the part and the whole, or "has a" to describe the relationship between the whole and the part.

      EXAMPLES:

      (1)In the ATM system, the CardReader, CustomerConsole, etc. objects are *parts of* the ATM object. This is a stronger connection than most of the examples of associations we have considered thus far.

      PROJECT class diagram

      (2)The relationship between a course and its students might also be thought of as an aggregation, though this is perhaps a bit more debatable. (Perhaps most appropriate in a situation were we are modeling student registrations in a course.)

   b) We also noted that aggregation actually comes in two forms: simple aggregation, and a stronger form, called *composition*.

(1)Composition has the additional characteristic that the “part” has no existence independent of the “whole”. This leads to two additional characteristics:

(a)Each “part” can belong to only one "whole".

(b)Each "part" cannot be moved to a different "whole"

(c)The “whole” is responsible for creating and destroying the “parts”.

i) The "whole" may potentially create that "part" at any time.

ii) The "whole" may potentially destroy the "part" at any time.

iii)If the “whole” is destroyed, the “parts” are destroyed too.

(d)Of the two examples we have considered:

i) The relationship between the ATM and its component parts is composition. From a software standpoint, one cannot imagine a component like a CardReader having an independent existence apart from an ATM nor can a CardReader belong to two different ATM’s, nor is it meaningful to think of moving the CardReader software to a different ATM.

ii) On the other hand, the relationship between courses and students is simple aggregation: students exist apart from their courses, and a given student can be - and typically is - a part of more than one course as the same time. Further, if a course is destroyed the student is not!

c) Think-pair-share activity: Some of the following involve either aggregation or composition, but not all. Which is appropriate in each case?)

(1) Soccer Team and Player

(2) Birth Mother and child

(3) Bank and BankAccount

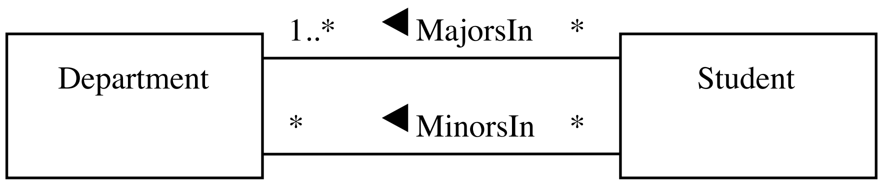
PROJECT

6. Association names. Often, it is useful to give an association a name.

   a) Example: Consider the association between students and the courses they are enrolled in. We might give this association the name EnrolledIn.

   b) This becomes particularly important when the same two classes are connected by more than one association.

      For example, a student might have both major(s) and minor(s). In this case, there are two kinds of association between a Student and a Department. This can be handled by the user of association names:



      PROJECT

7. Association classes

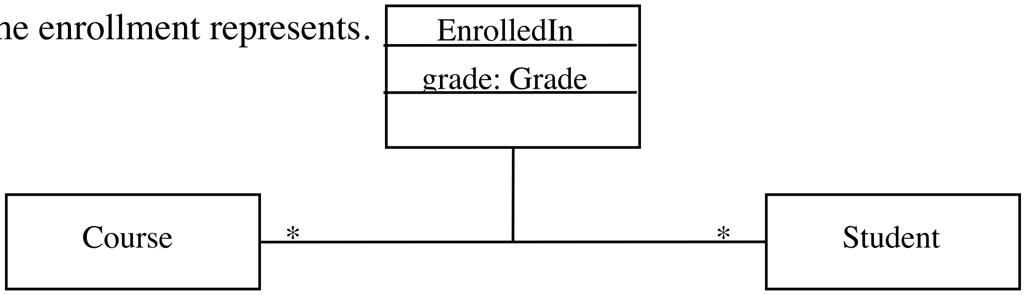
   a) Sometimes, an association has an attribute that is a property of the association itself, rather than of the participating classes. For example, consider the case of a student's enrollment in a course - an association between a Student object and a Course object that is preserved even after the end of the semester for purposes like creating a transcript or calculating GPA.

(1)The grade earned is not an attribute of the Student - most students earn a variety of different grades in different courses.

(2)It is not an attribute of the Course. In most courses, different students earn different grades.

(3)Rather, it is an attribute of the <u>association</u> between the student and the course.

b) A case like this calls for representing the association by an object that has a grade attribute - not just a simple link between two objects. This class this object belongs to is called an association class. Each object of this class represents the enrollment of a specific student in a specific course. There is an association between this object and the Student, denoting that this particular student is the one enrolled, and there is an association between this object and the Course, denoting that this particular course is the one the enrollment represents.



PROJECT

c) The use of an association class to represent an association adds complexity, so it is only used where needed!

B. Associations are used for three general purposes:

1. We have already seen that associations can be used to represent a situation in which an object of one class *uses* the services of an object of another object, or they mutually use each others services -

i.e. one object sends messages to the other, or they send messages back and forth. (In the former case, the navigability can be monodirectional; in the latter case it must be bidirectional.)

2. Associations can be used to represent aggregation or composition - where objects of one class are wholes that are composed of objects of the other class as parts. In this case, a uses relationship is implicitly present - the whole makes use of its parts to do its job, and the parts may also need to make use of the whole.

3. As a third possibility, associations can also be used to represent a situation in which objects are related, even though they don't exchange messages. This typically happens when at least one of the objects is basically used to store information - e.g. in the AddressBook problem we used in lab 3, this is the relationship between the AddressBook object and the various Person objects it stores. (The AddressBook doesn't directly send messages to Persons, though it can be used to retrieve a Person that some other object can then send a message to.)

   (Some writers call this a *weak relationship*. This is not a standard term, however.)