Software Engineering

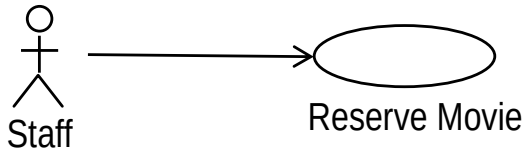
System Analysis and Design

EXERCISE: CLASS DESIGN

Reserve Video—*Staff* Scenario

This use case allows a staff to reserve up to 5 movies concurrently for a member.

Use-case Diagram



Preconditions

1. The member has less than 5 movies reserved.

Basic Flow

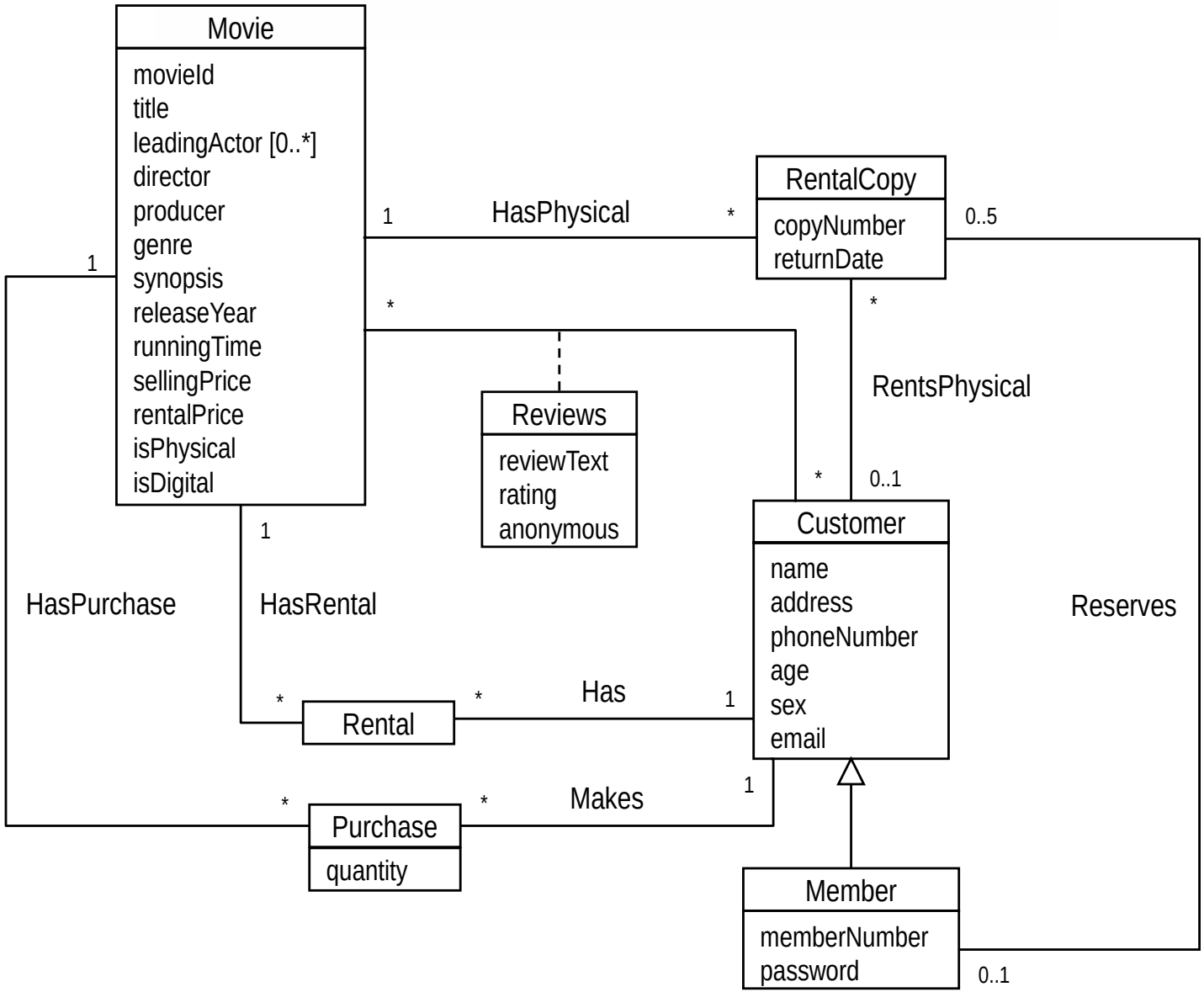
1. The use case begins when the staff actor chooses to reserve a movie.
2. The system prompts the staff to input the member number and movie ID.
3. The staff inputs the member number and movie ID.
4. The system reserves a copy of the movie for the member.
5. The system notifies the staff that the reservation has been made.
6. The use case ends.

Given the above use case and the Movie Shop domain model:

(a) How would you determine whether a copy of a movie can be reserved?

(b) Given your answer to (a), how can the process of determining whether a copy of a movie can be reserved be made more efficient?

EXERCISE: CLASS DESIGN



EXERCISE: CLASS DESIGN—SOLUTION

(a) How would you determine whether a copy of a movie can be reserved?

The first important thing to note is that the question asks how to determine whether *a copy of a movie* can be reserved, not whether *a member can reserve a copy of a movie*.

To determine whether a copy of a movie can be reserved, we need to check two things with respect to the corresponding RentalCopy object:

1. Is the copy currently reserved (i.e., does the RentalCopy object currently have a Reserves link; if yes, then it cannot be reserved);
2. Is the copy currently rented (i.e., does the RentalCopy object currently have a RentsPhysical link; if yes, then it cannot be reserved).

EXERCISE: CLASS DESIGN—SOLUTION (CONT'D)

(b) Given your answer to (a), how can the process of determining whether a copy of a movie can be reserved be made more efficient?

Since it can be quite expensive to check whether links exist and since in some cases both tests need to be done, one simple way to optimize this checking is to add an new attribute *status* to the RentalCopy class. The value of this attribute can be either: available, reserved or rented. The attribute is set whenever a copy is returned, reserved or rented, respectively. Now all that we need to check is the value of this attribute to determine whether a copy of a movie can be reserved (or rented).

Since we are given only a movie ID to find a rental copy, another optimization would be to include the attribute *movieID* in the class *RentalCopy* so that we do not need to follow the links from *Movie* to *RentalCopy* when checking whether a copy is available to be reserved. (In fact, if the domain model is stored as a relational database, this would be done anyway to implement the *HasPhysical* association.)