Relationships Between Classes

COMP 3831

Larman: Chapter 9

Relationships

- Relationships provide a pathway for communication between objects

- Sequence and/or collaboration diagrams are examined to determine what links between objects need to exist to accomplish the behavior -- if two objects need to "talk" there must be a link between them

Relationships

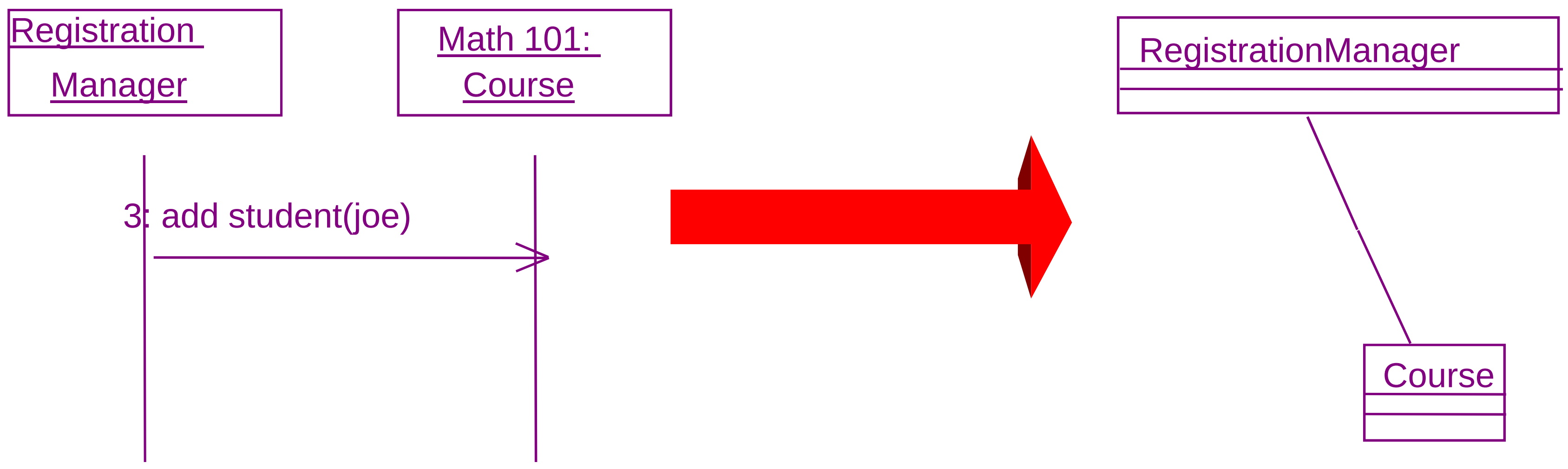
- Four types of relationships:
  - Association
  - Aggregation (Composition)
  - Dependency
  - Generalization

Relationships

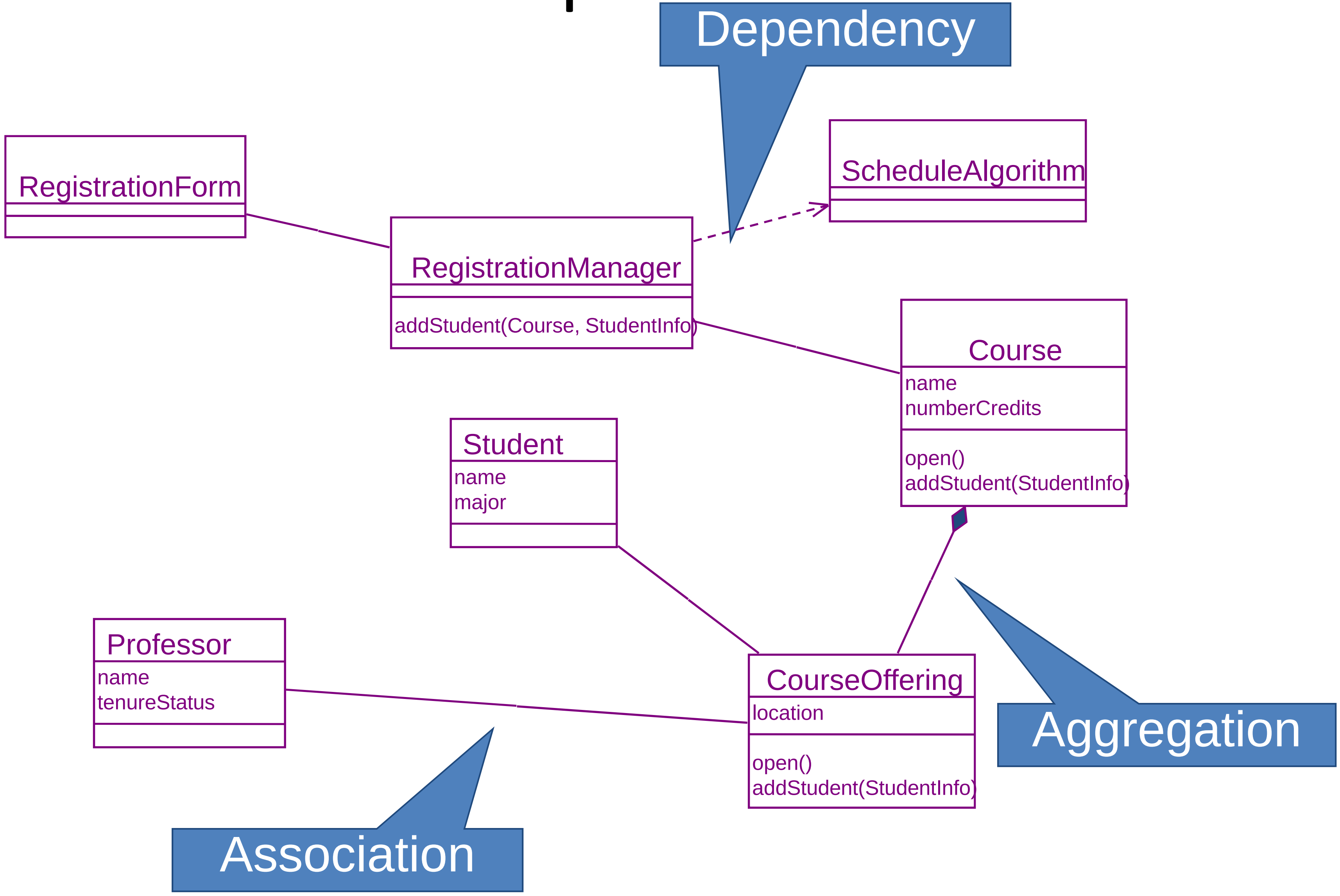
- **Association:** a bi-directional connection between classes
  - An association is shown as a line connecting the related classes
- **Aggregation:** a stronger form of relationship where the relationship is between a whole and its parts
  - An aggregation is shown as a line connecting the related classes with a diamond next to the class representing the whole
- **Dependency:** relationship is a weaker form of relationship showing an interest between a client and a supplier
  - A dependency is shown as a dashed line pointing from the client to the supplier
- **Generalization:** relationship in which one model element (the child) is based on another model element (the parent).

Finding Relationships

- Relationships are discovered by examining interaction diagrams
  - If two objects must "talk" there must be a pathway for communication

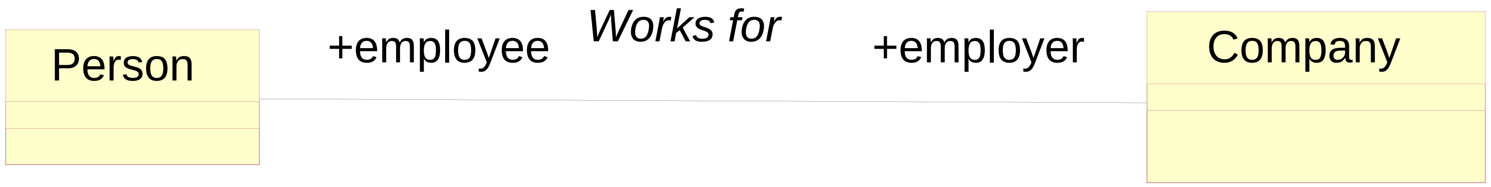


Relationships



Roles

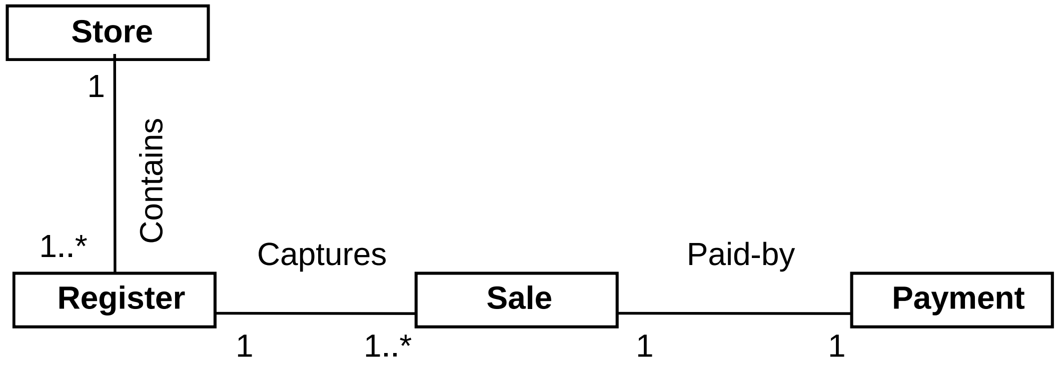
Each end of an association is called a Role.



- Name is read from left to right
- Plus sign on role indicates that they are public
- Roles may optionally have:
  - Name
  - Multiplicity
  - Navigability
  - Type

Naming associations

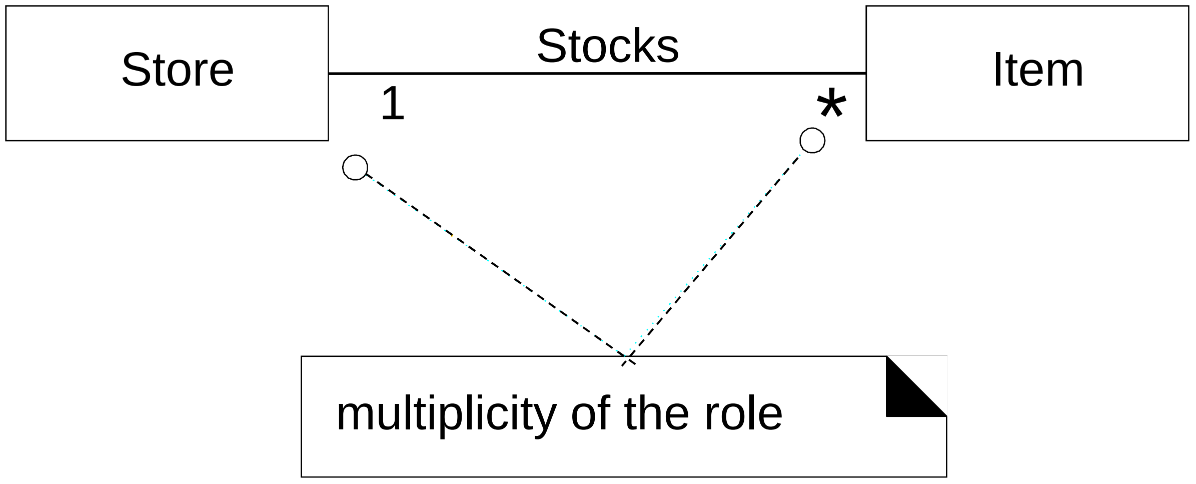
- Association may or may not have name
- Name an association based on *TypeName-VerbPhrase-TypeName* format.
  - Example: *Register-Captures-Sale*
- Association names should start with a capital letter.
- Describes nature of relationship
- Associations generally don't have names



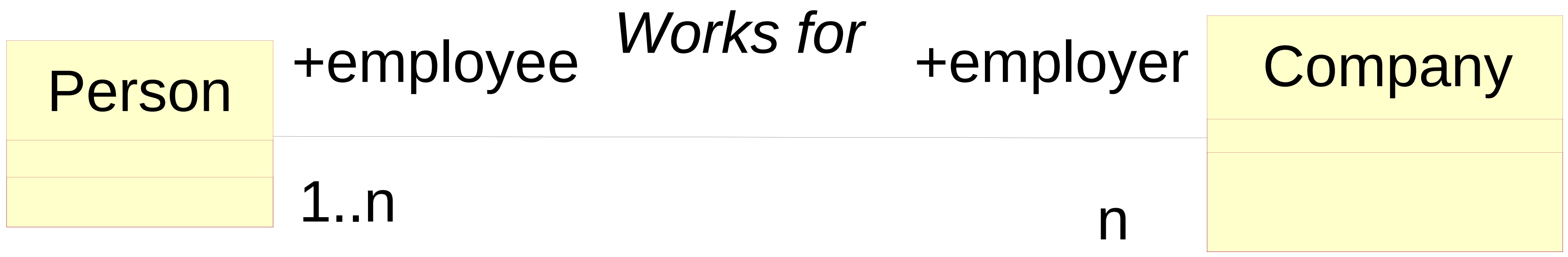
Multiplicity

Multiplicity defines how many objects participate in a relationship

– Multiplicity is the number of instances of one class related to ONE instance of the other class



Example of Multiplicities



Be aware that the UML uses * for many but the Rational Rose implementation uses n

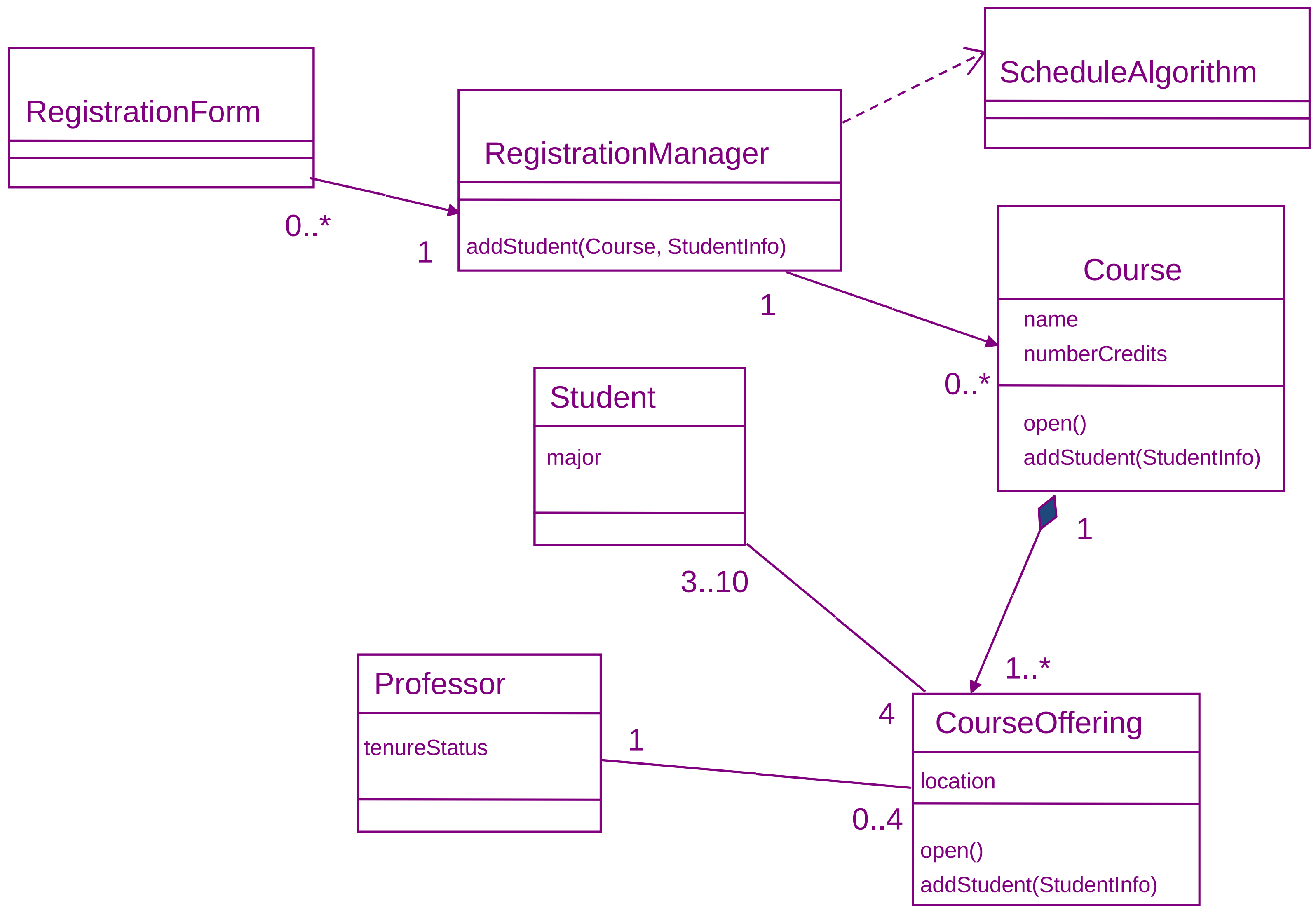
You can use n or put the * in yourself

Navigability

- Although associations and aggregations are bi-directional by default, it is often desirable to restrict navigation to one direction

- If navigation is restricted, an arrowhead is added to indicate the direction of the navigation

Multiplicity and Navigation

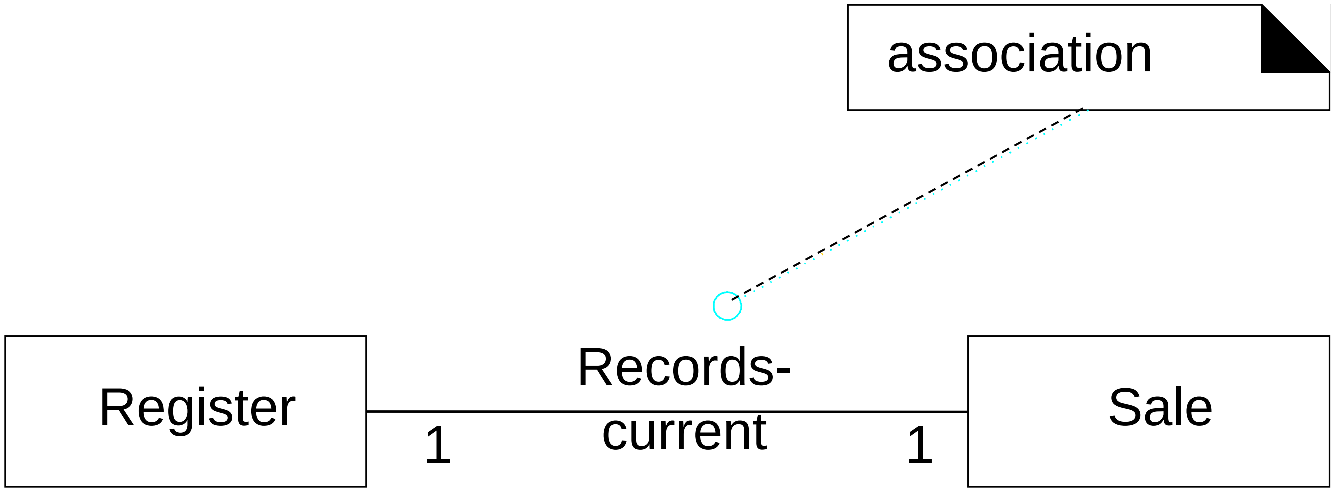


Association

- Bidirectional semantic connection between classes
- Not a data flow as defined in structured analysis and design
- Data may flow in either direction across the association
- An association between classes means that there is a link between objects in the associated classes

Focus on associations

- Link is physical or conceptual connection between object instances
- An association allows navigation from one object to another



Identifying Associations

- More difficult than finding classes
- A relationship that needs to be preserved for some duration (need-to-know associations)
  - Ask the question:
    - Between what objects do we need some memory of a relationship?
- Look at verbs and verb phrases in problem statement

Identifying Associations (continued ..)

- Any message between classes on a sequence or collaboration diagram requires a relationship between the classes
- Don’t worry about implementation details
- Consider deriving associations from the “Common Associations List”.

Common Association List

Category	Examples
A is a physical part of B	Drawer←→Register; Wing←→Airplane
A is a logical part of B	SalesLineItem←→Sale; FlightLeg←→FlightRoute
A is physically contained in B	Register←→Store; Passenger← →Airplane
A is logically contained in B	ItemDescription←→Catalog; Flight←→FlightSchedule
A is a description for B	ItemDescription←→Item; FlightDescription←→Flight
A is a line item of a transaction or report in B	SalesLineItem←→Sale; MaintenanceJob←→MaintenanceLog
A is known/logged/recorded/reported/captured in B	Sale←→Register; Reservation←→FlightManifest
A is a member of B	Cashier←→Store; Pilot←→Airline
A is an organizational sub-unit of B	Department←→Store;Maintenance ←→Airline
A uses or manages B	Cashier←→Register; Pilot←→Airplane
A communicates with B	Customer←→Cashier; ReservationAgent←→Passenger
A is related to a transaction B	Customer←→Payment; Passenger←→Ticket
A is a transaction related to another transaction B	Payment←→Sale; Reservation←→Cancellation
A is next to B	SalesLineItem←→SalesLineItem; City←→City
A is owned by B	Register←→Store; Plane←→Airline
A is an event related to B	Sale←→Customer; Departure←→Flight

High-priority associations

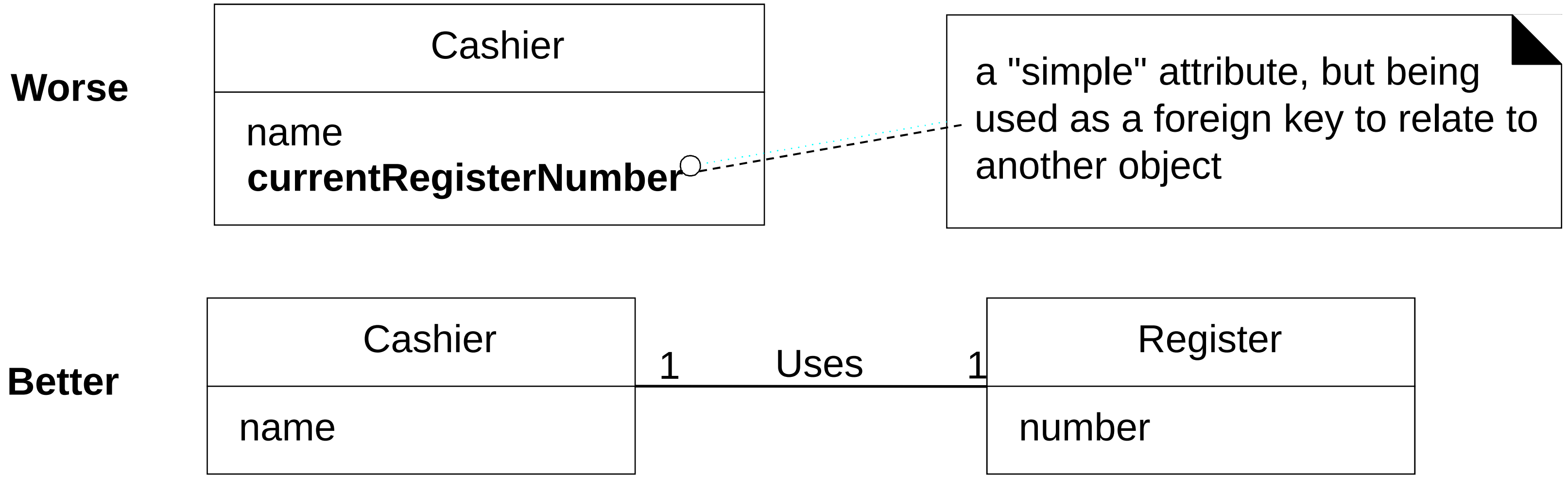
- A is a physical or logical part of B
- A is physically or logically contained in/on B
- A is recorded in B

Association guidelines

- Focus on need-to-know associations
- It is more important to identify conceptual classes than to identify associations

Pitfalls

- Attributes should not be used in place of associations

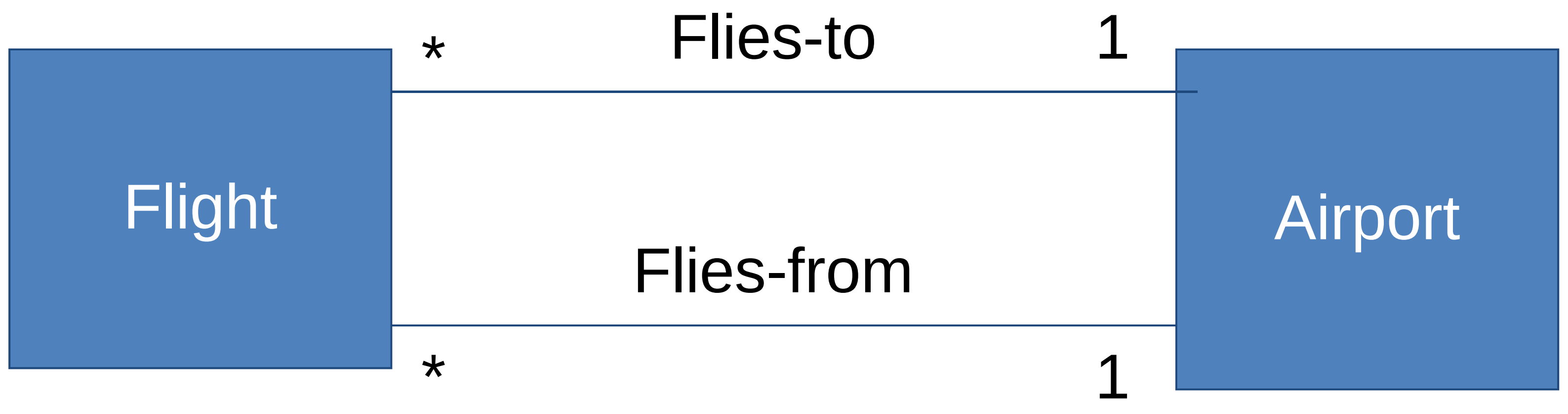


Worse
Cashier
name
currentRegisterNumber
a "simple" attribute, but being used as a foreign key to relate to another object
Better
Cashier
name
1
Uses
1
Register
number

- Many lines on a diagram will clutter it (visual noise) and make in incomprehensible.
  - In a diagram with n different conceptual classes, there can be n(n-1) associations
  - Do not include associations and are not useful in the context of the requirements
  - Avoid showing redundant or derivable associations

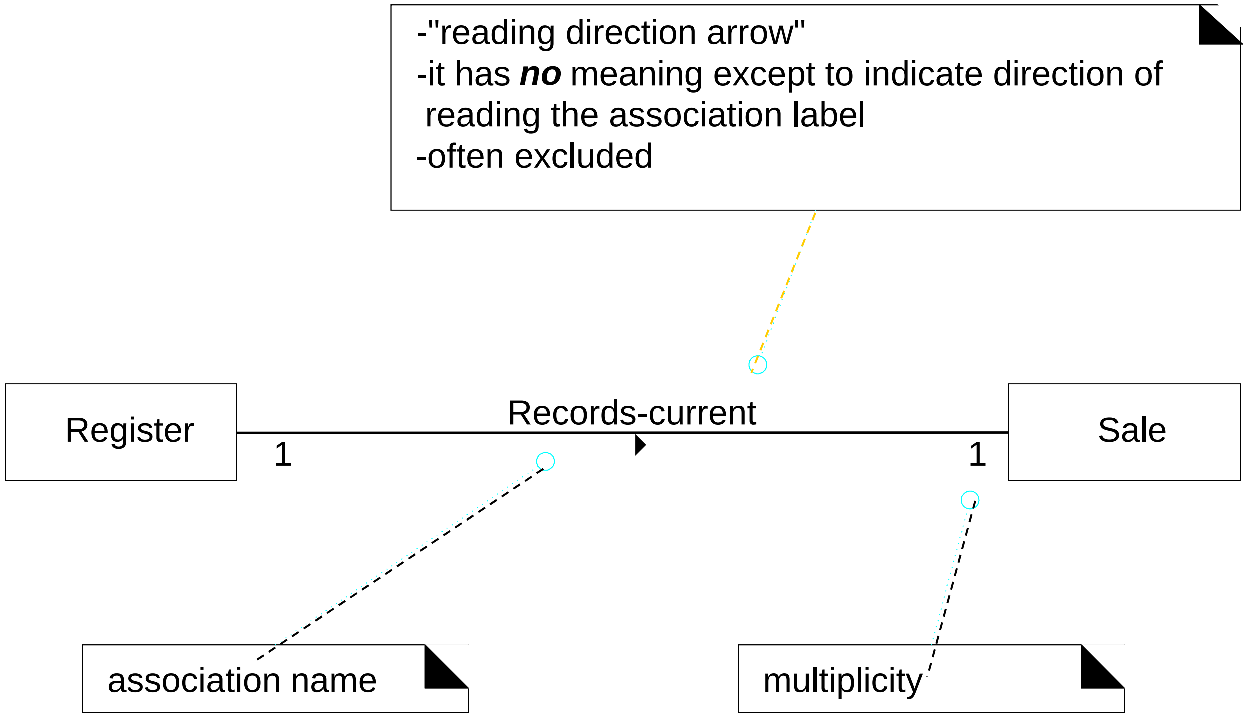
Multiple Associations

It is not uncommon for two types to have multiple associations between them



UML Association notation

- an association is represented as a line between classes with an association name.
- associations inherently bi-directional (logical traversal from either instance to the other)
- may contain multiplicity
- conventional to read from left to right and top to bottom


-"reading direction arrow"
-it has ***no*** meaning except to indicate direction of reading the association label
-often excluded
Register
Records-current
Sale
1
1
association name
multiplicity

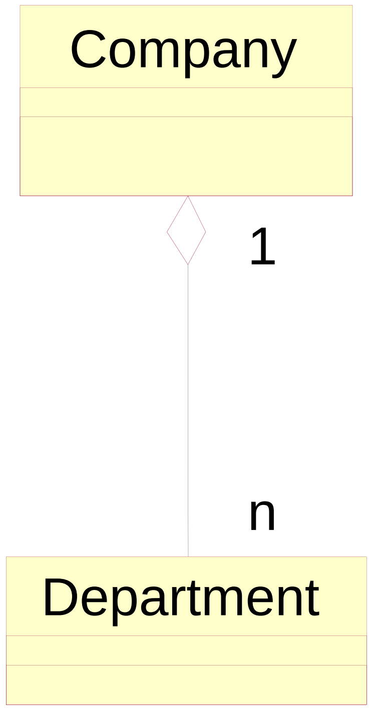
Aggregation = “has a”

- A specialized case of association
  - All aggregations are associations
  - Not all associations are aggregations
- A whole/part relationship
- Obvious example of a "has a" relationship
- An object of the whole contains an object or objects of each part

Testing for Aggregation

- Is the phrase "part of" used to describe the relationship?
- Are some operations on the whole automatically applied to its parts? For example, delete a course then delete all of its course offerings.
- Is there an intrinsic asymmetry to the relationship where one class is subordinate to the other?

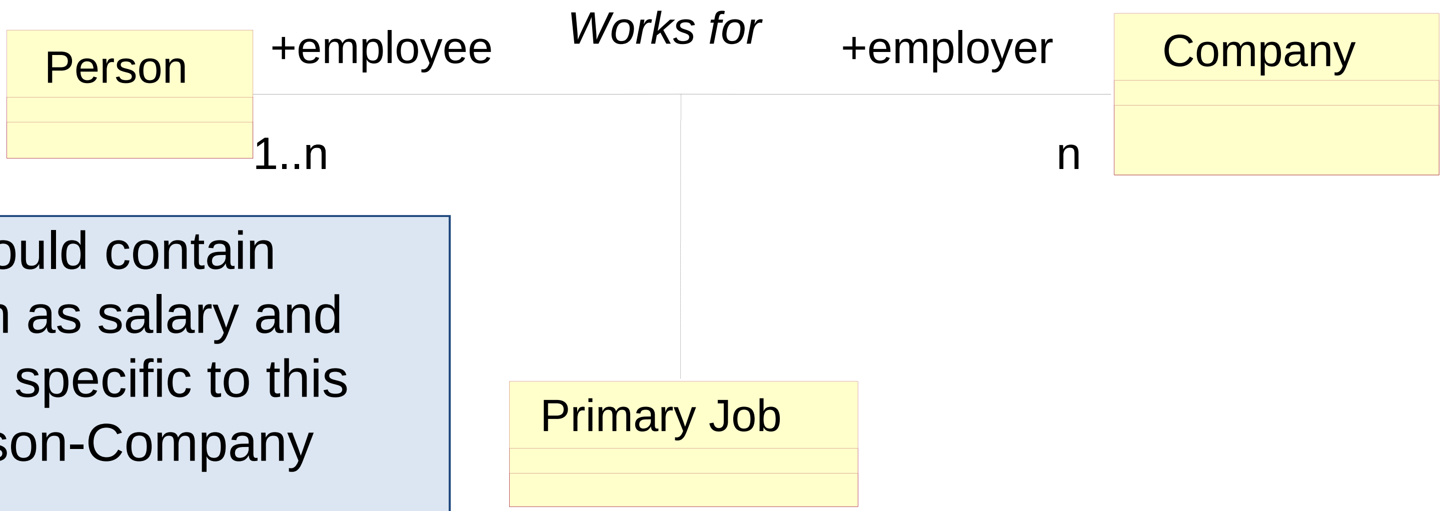
Example of Aggregation



An aggregation is represented as an open diamond with diamond on the aggregate end

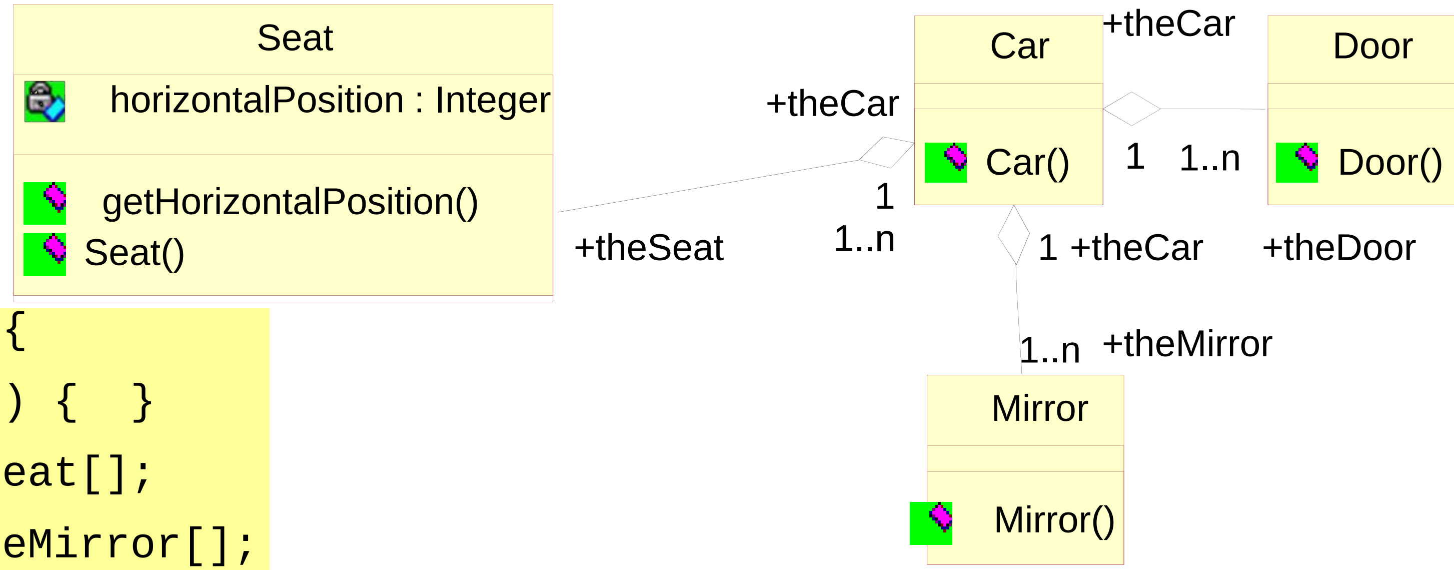
Modeling Association as Class

- Each link between objects is an instance of the class
- Most commonly done when:
  - Properties associated with the link
  - Association is many to many
- Operations are less common



Primary job would contain attributes such as salary and hours of work, specific to this particular Person-Company association.

Resolving associations in code?



```
public class Car {
 public void Car() {  }
 public Seat theSeat[];
 public Mirror theMirror[];
 public Door theDoor[];
 }
```

```
public class Door {
   public void Door() {  }
}
```

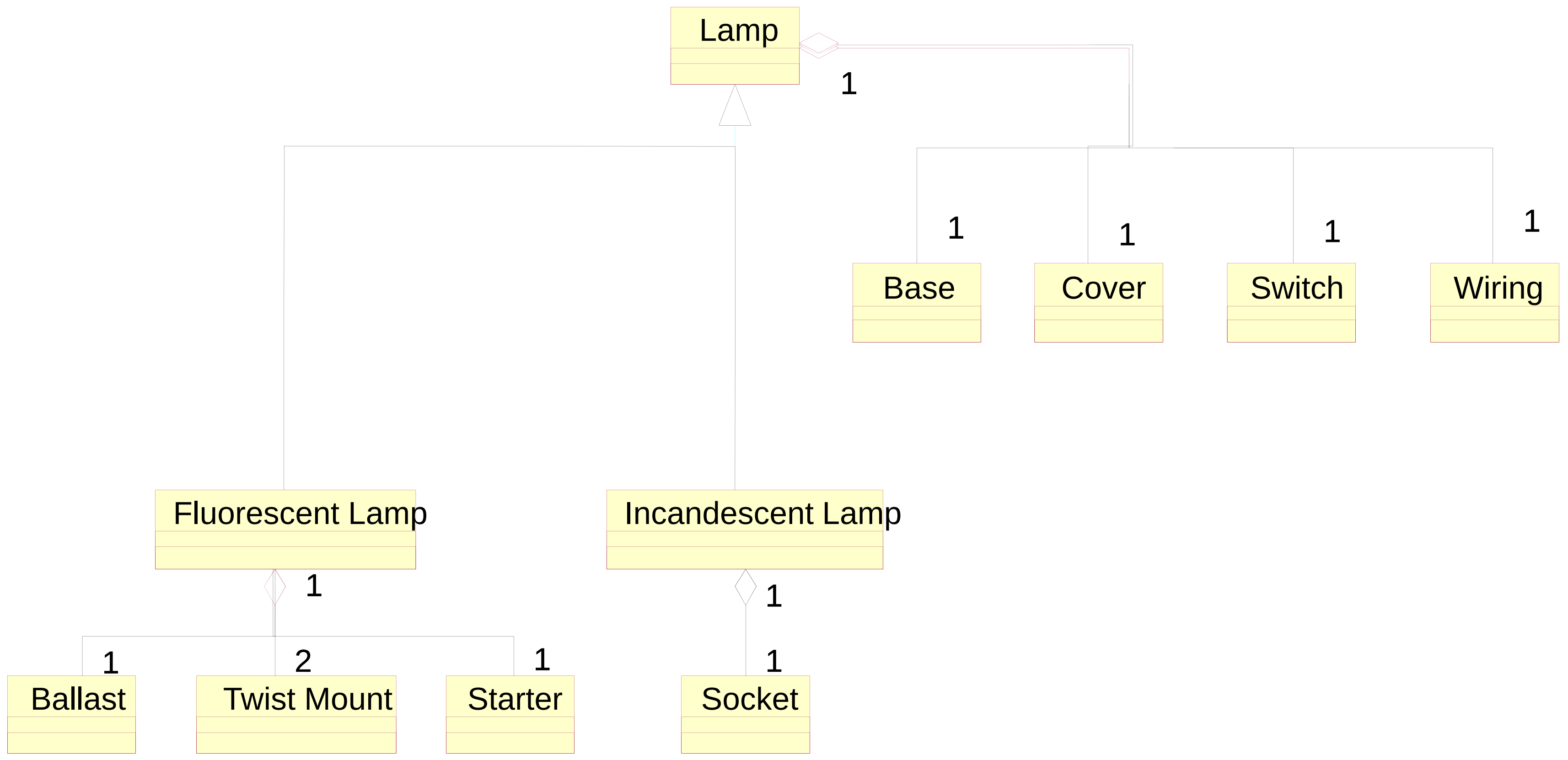
```
package Auto;
public class Seat {
   public void Seat() {   }
   public void getHorizontalPosition() {   }
   private Integer horizontalPosition;
}
```

```
package Auto;
public class Mirror {
   public Car theCar;
   public Mirror() {  }
}
```

Recursive Aggregation

- Recursive aggregation is common
- Recursive aggregate contains an instance of itself
- A block of code is either a compound statement or a simple statement
- A compound statement is made up of blocks

Aggregation vs Generalization



Questions and Conclusions