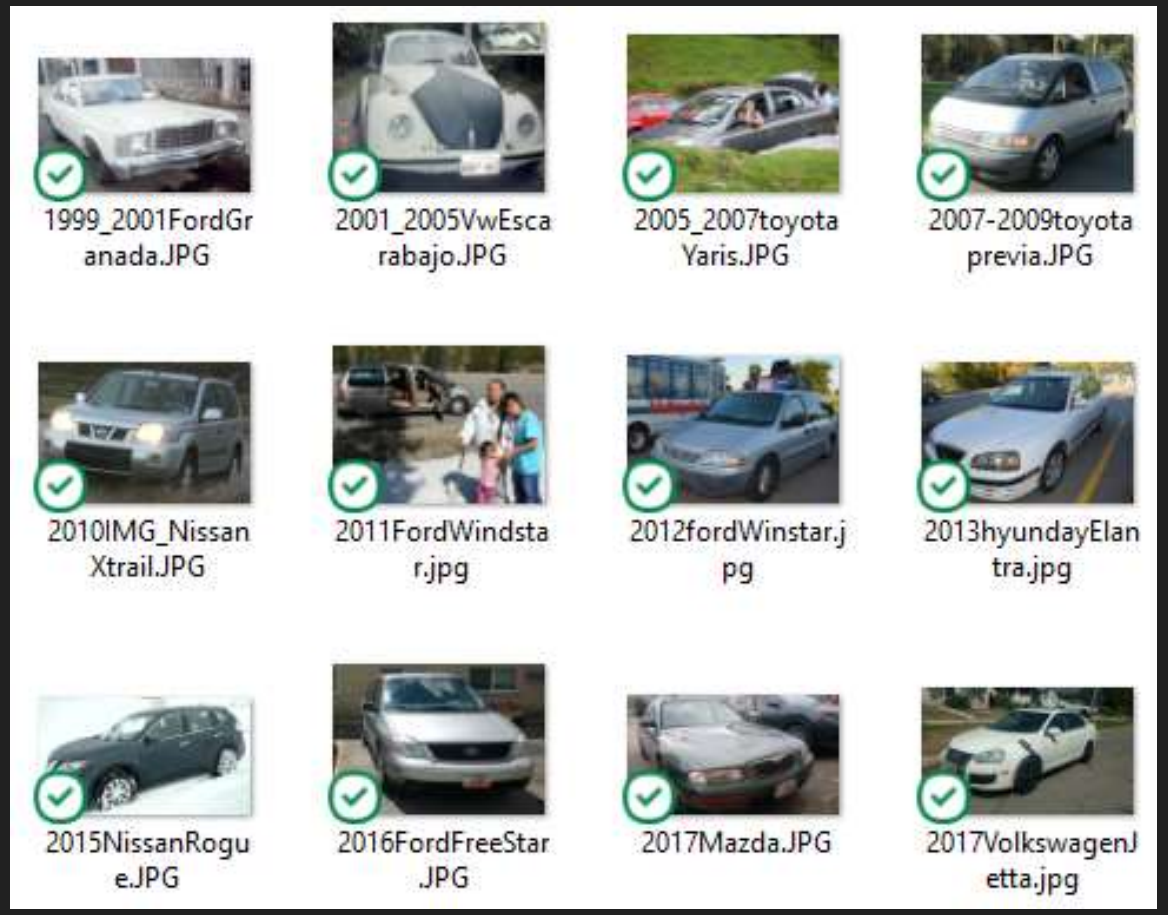
Associations and Multiplicity Constraints

Jorge Edison Lascano, Object Oriented Programming – Universidad de Las Fuerzas Armadas ESPE

Based on the slides of

Stephen Clyde Ph.D., Utah State University

Associations



Links Among Objects

A software object can include, reference , or point to an object. For example,

Vehicle #12543643 was built by Ford

Vehicle #24636325 was built by Honda

Vehicle #74845726 was built by Ford

Sammy's Used Cars currently owns Vehicle #24636325

Susan Smith currently owns Vehicle #24636325

Jonathan Johnson currently owns Vehicle #74845726

Logically, links can be navigated or "queried". For example,

What manufacturer built Vehicle #12543643 ?

- Ford
- What were vehicles built by Ford?
  - Vehicle #12543643 and Vehicle #74845726
- Who currently owns Vehicle #24636325?
  - Susan Smith
- Who currently own vehicles built Ford?
  - Sammy's Used Cars and Jonathan Johnson

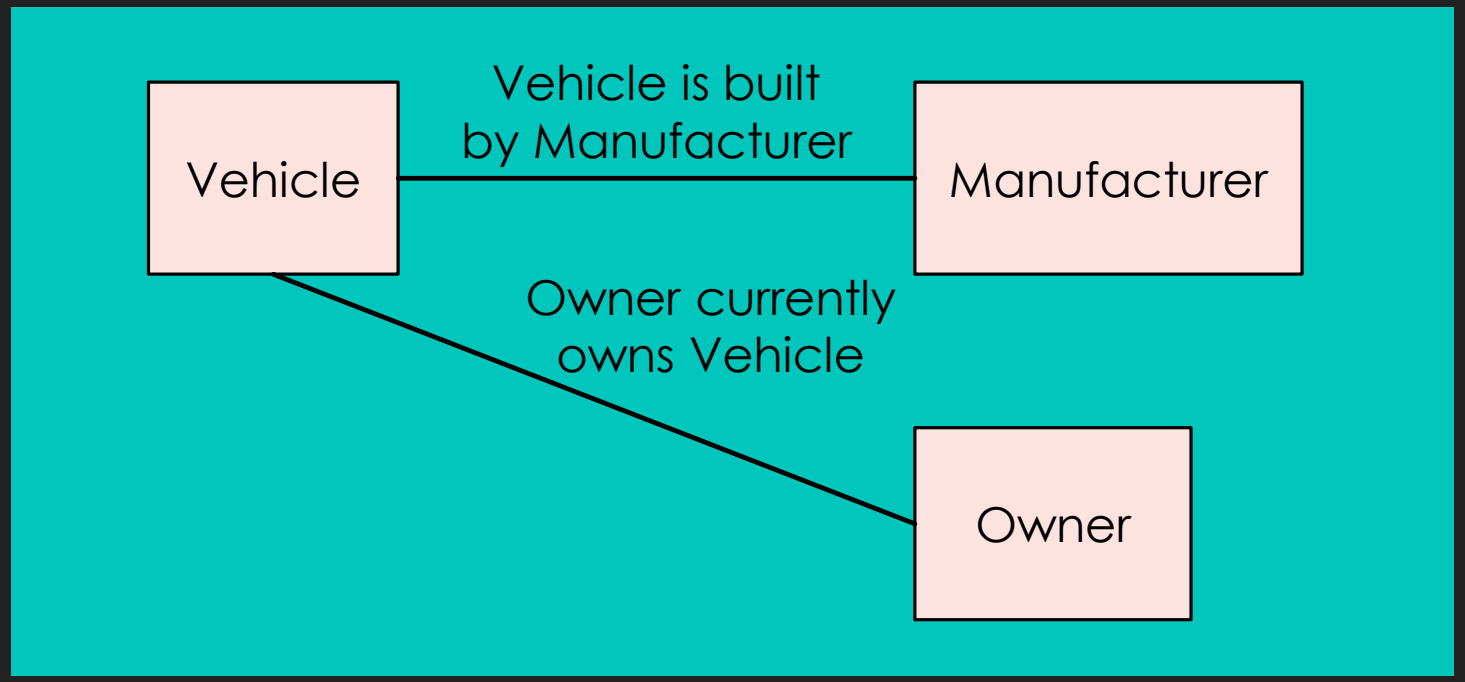
Associations in UML

- Associations model sets of links between classes of objects
- Specifically, a binary association connects two classes, e.g., Vehicle and Manufacturer, and describes a set of links such as each link connects an object from one of the classes with an object from the other

- An association is represented by a solid line that connects the classes

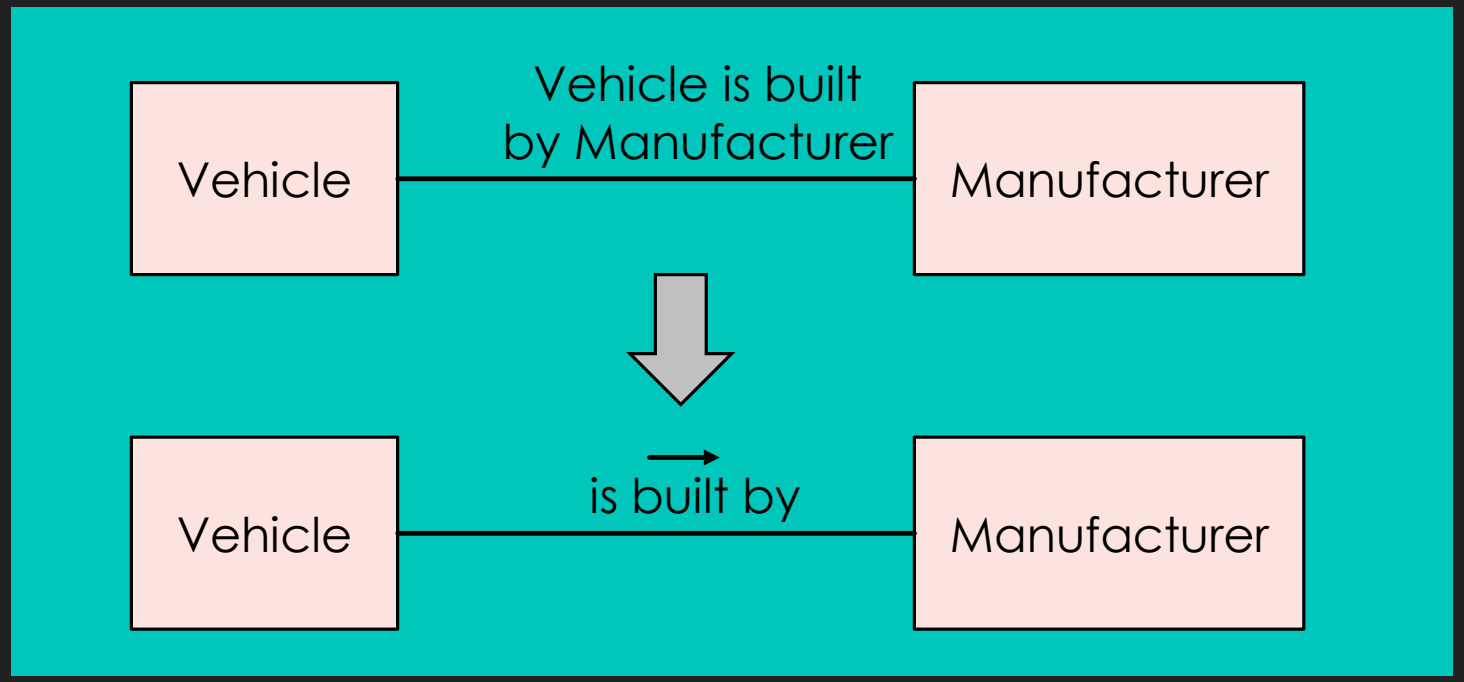
Associations in UML

- Every association has a name that describes the meaning of an individual link
  - Fully written out, the name includes the connected class names and forms an understandable sentence or phrase, e.g. "Vehicle is built by Manufacturer".

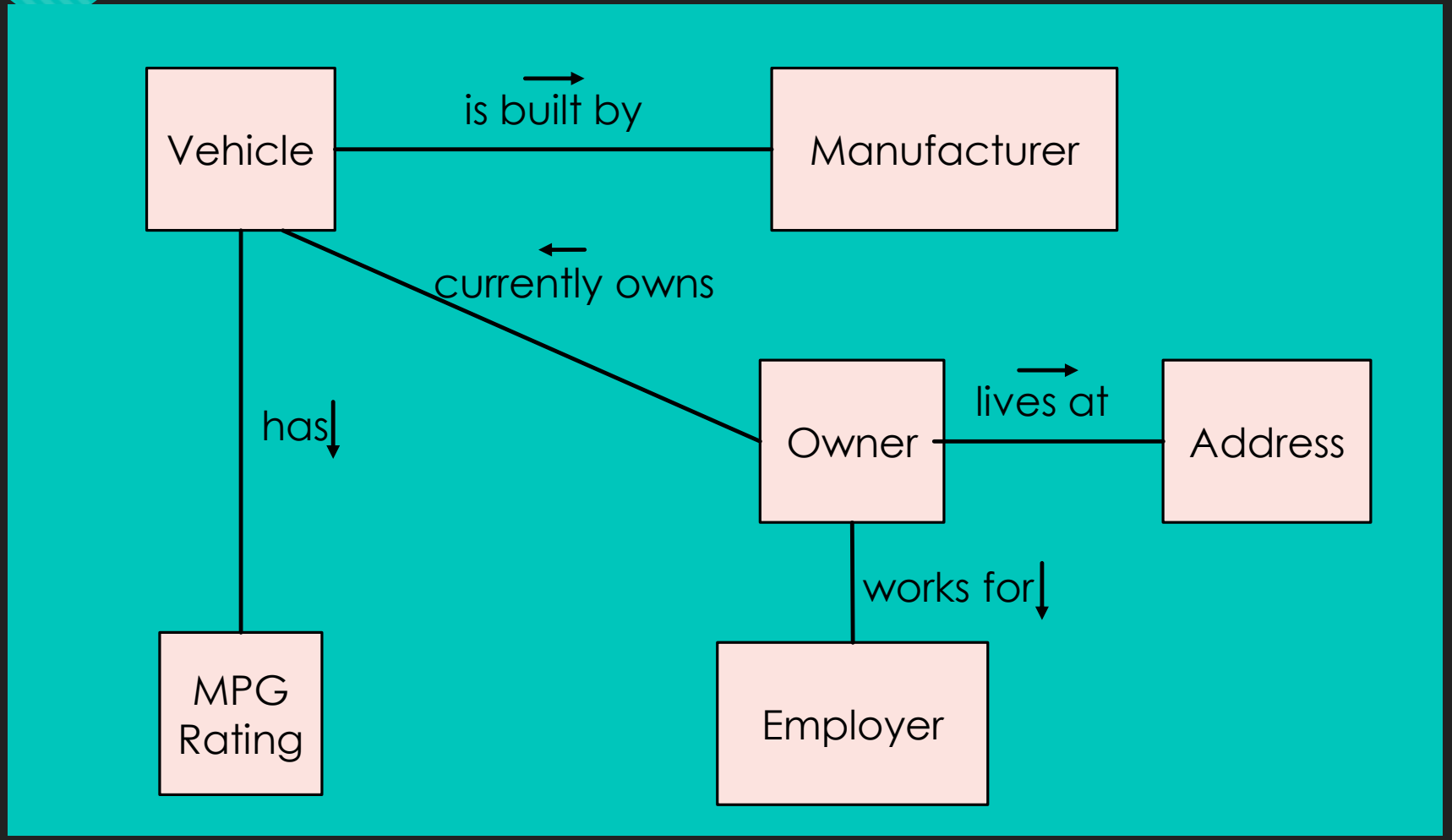


Associations in UML

- The association name can be abbreviated to a verb or a prepositional phrase by dropping the class names and using an arrow by the name to indicate how it is to be read

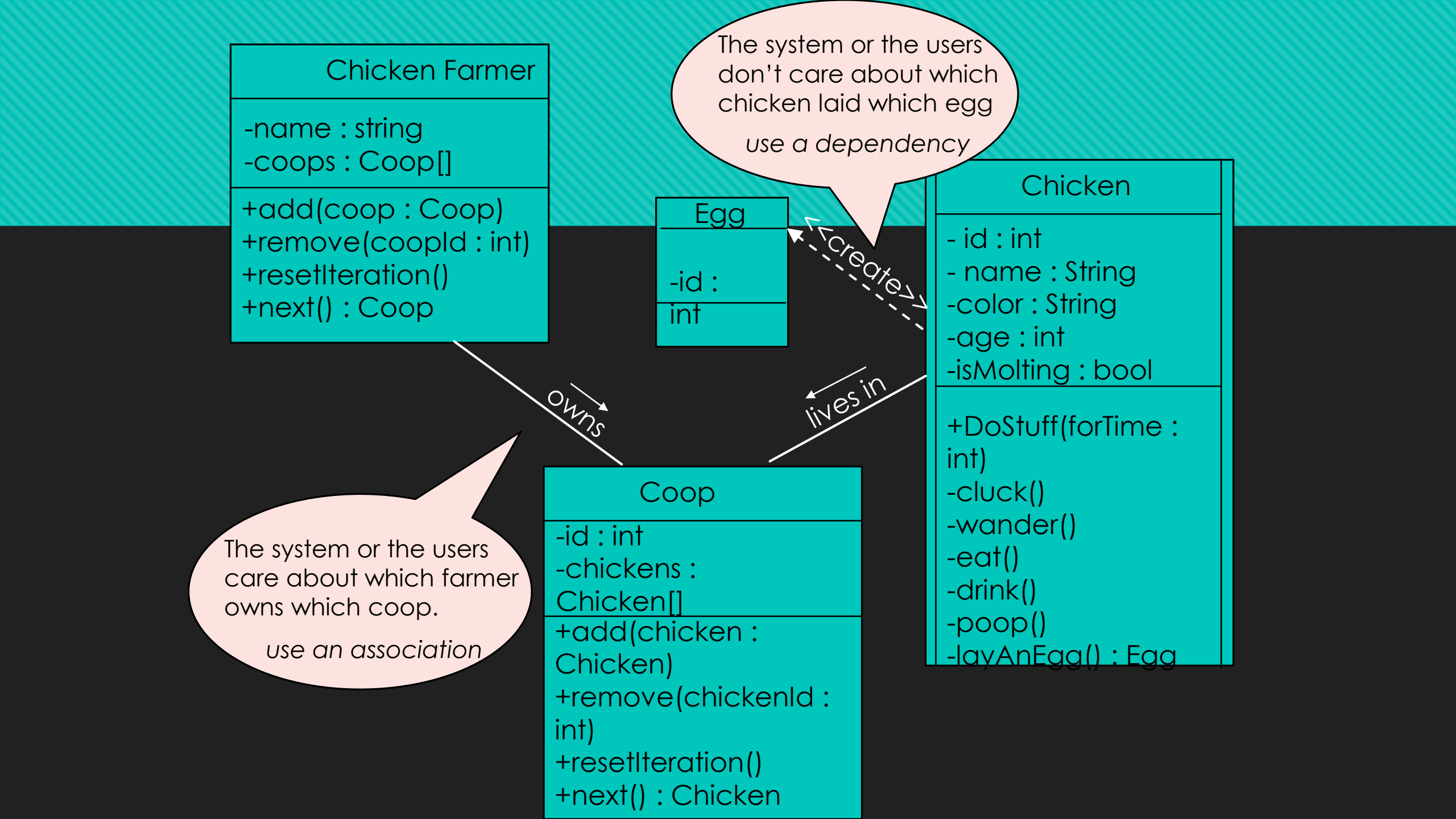


More Examples



Differences Between Associations and Dependencies

- Dependencies communicate that a class relies on another class in some way, e.g.,
  - A class's definition uses another class's definition
  - The implementation of a method in a class uses or creates objects from another class
- Associations explain that objects from a class have some kind of link to objects from another class, e.g.,
  - Farmer John owns Chicken Coop #234
  - Renan is registered for NRC4023, Section 1, Period 201950
  - Daniel has an annual income of $14,000
- A dependency provides insight into the static definition of a class; an association tells us something about the structure of the runtime data


Chicken Farmer
-name : string
-coops : Coop[]
+add(coop : Coop)
+remove(coopId : int)
+resetIteration()
+next() : Coop
The system or the users don't care about which chicken laid which egg
use a dependency
Egg
-id : int
create
Chicken
- id : int
- name : String
-color : String
-age : int
-isMolting : bool
+DoStuff(forTime : int)
-cluck()
-wander()
-eat()
-drink()
-poop()
-layAnEgg() : Egg
owns
lives in
The system or the users care about which farmer owns which coop.
use an association
Coop
-id : int
-chickens : Chicken[]
+add(chicken : Chicken)
+remove(chickenId : int)
+resetIteration()
+next() : Chicken

Discovering Associations

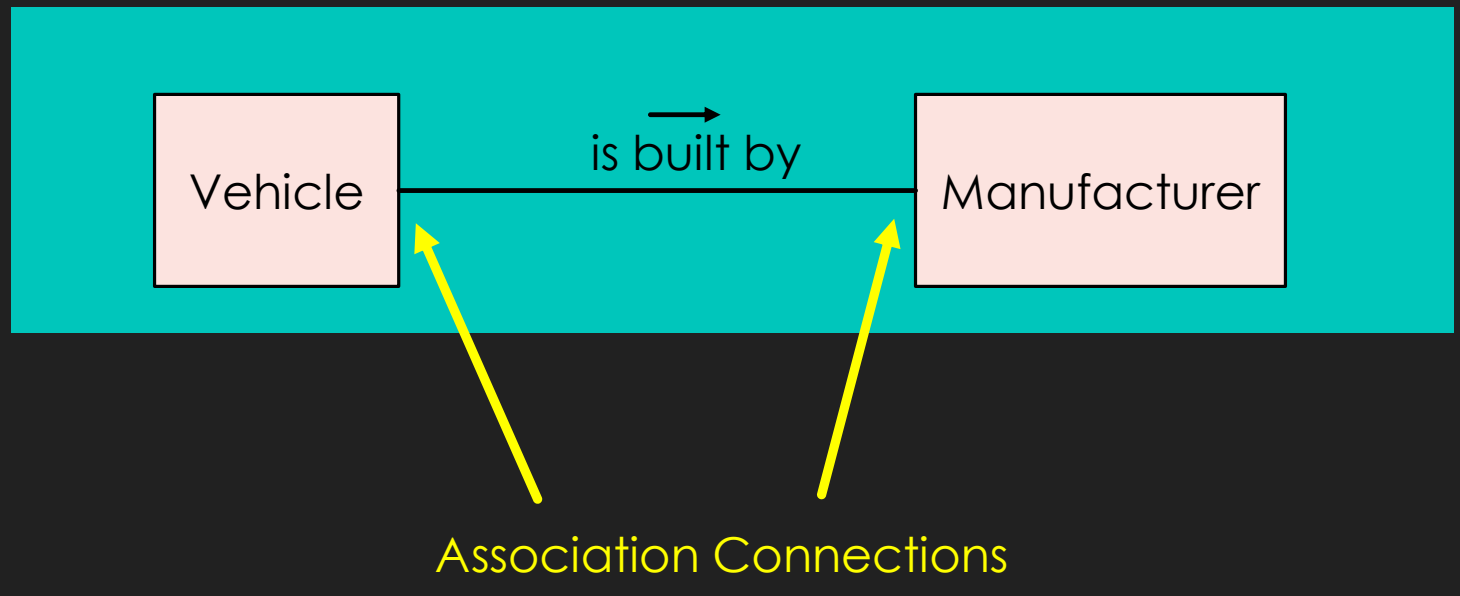
Document a description of the system from the user's or customer's prescriptive and look for

- Interesting classes (nouns and noun phrases)
- connections between objects of those classes (verbs, verb phrases, or prepositional phases)

Consider those that represent information or facts the system will need to navigate through or query, not just actions

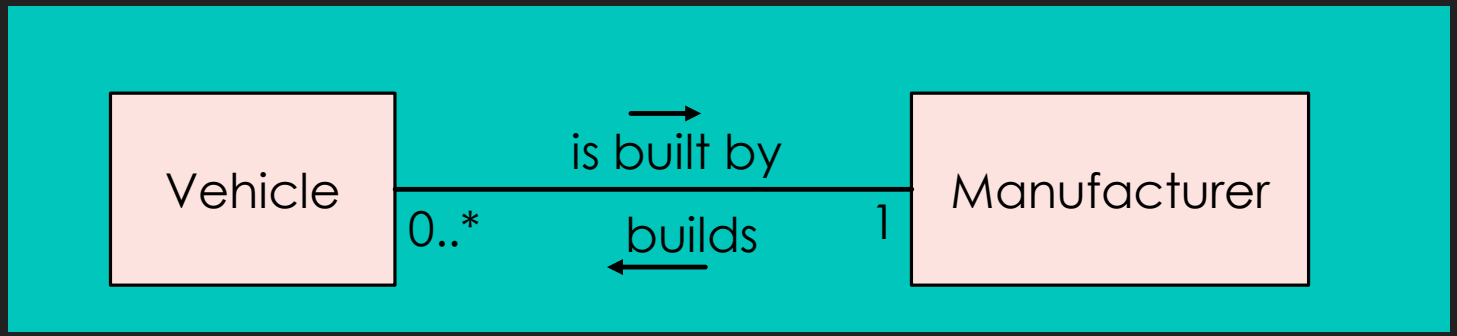
Association Connections

- An association connection is the class's connection to the association
- It can be adorned with additional descriptive information
  - Multiplicity Constraints
  - Roles



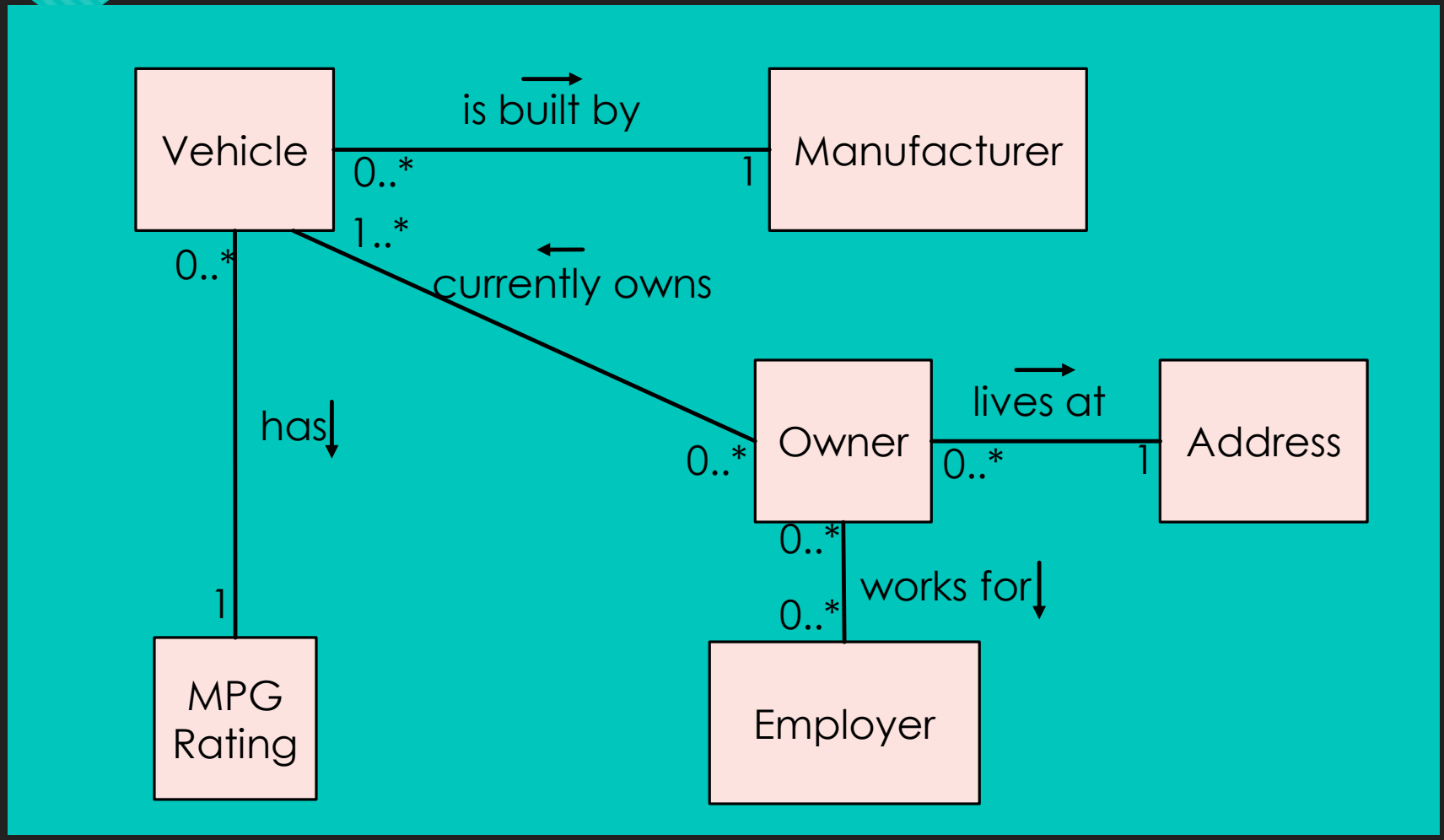
Multiplicity Constraints

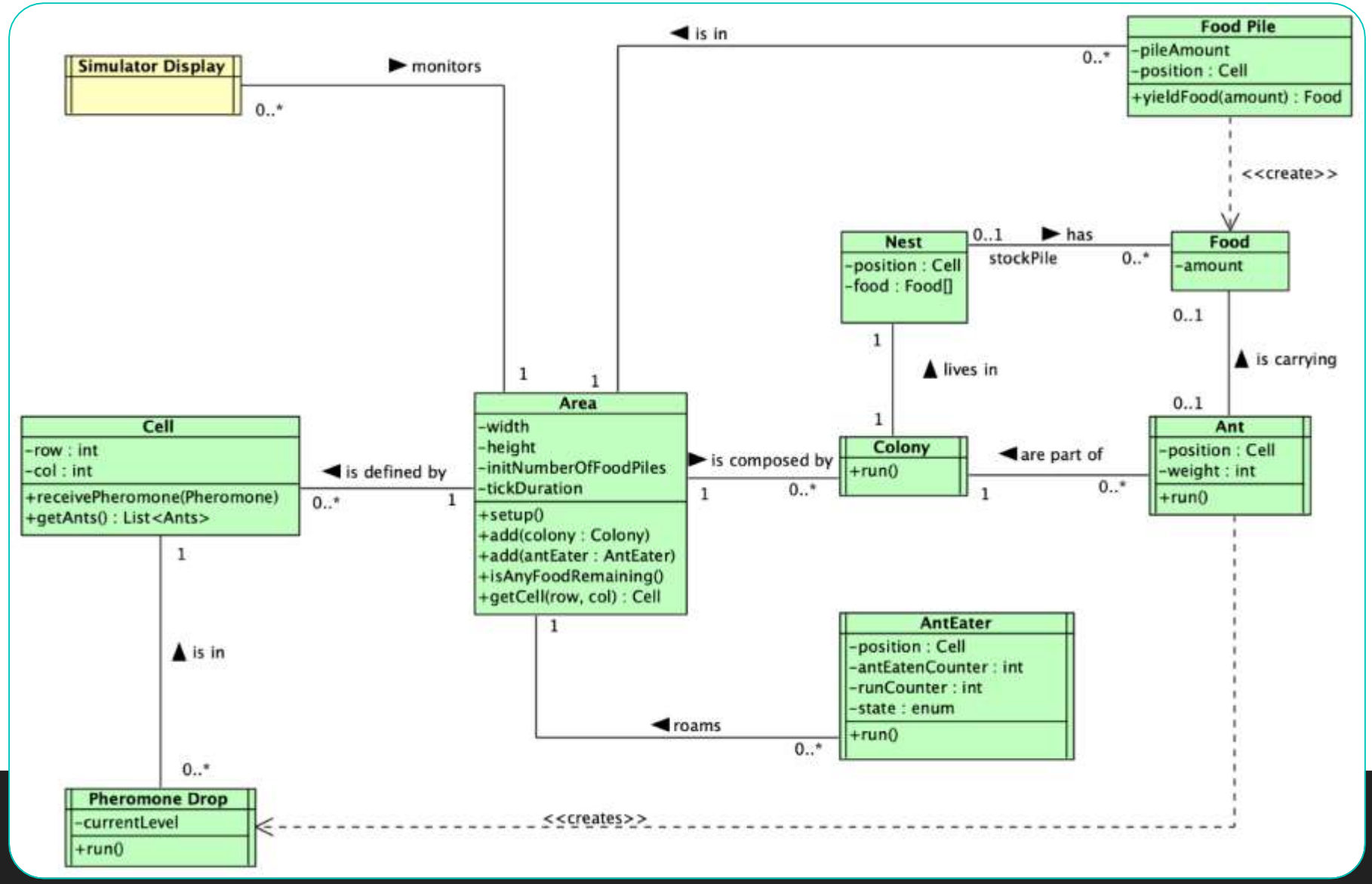
- A multiplicity constraint tells us how many times an object from one class can be linked to objects from another classes, e.g.
  - A vehicle is built by 1 manufacturer
  - A manufacturer built zero or more vehicles



- To read a multiplicity constraint,
  - read the name of a first class and the abbreviated relationship name
  - Then jump across the limit
  - Read the multiple constraint
  - Finally, read the name of the second class

Examples


Vehicle
is built by
0..*
1
Manufacturer
1..*
0..*
currently owns
has
0..*
Owner
lives at
0..*
1
Address
0..*
works for
0..*
1
MPG Rating
Employer


Simulator Display
monitors
0..*
1
1
is in
0..*
Food Pile
-pileAmount
-position : Cell
+yieldFood(amount) : Food
<<create>>
Nest
-position : Cell
-food : Food[]
0..1
has
stockPile
0..*
Food
-amount
0..1
1
lives in
1
is carrying
0..1
Area
-width
-height
-initNumberOfFoodPiles
-tickDuration
+setup()
+add(colony : Colony)
+add(antEater : AntEater)
+isAnyFoodRemaining()
+getCell(row, col) : Cell
Cell
-row : int
-col : int
+receivePheromone(Pheromone)
+getAnts() : List<Ants>
is defined by
0..*
1
is composed by
1
0..*
Colony
+run()
are part of
1
0..*
Ant
-position : Cell
-weight : int
+run()
1
is in
0..*
1
AntEater
-position : Cell
-antEatenCounter : int
-runCounter : int
-state : enum
+run()
roams
0..*
Pheromone Drop
-currentLevel
+run()
<<creates>>

Examples

Roles

- Roles on an association connection provide an alternate name for a set of objects from the connected objects perspective

WHAT IS NEXT..

Associations and Attributes