More About Classes

Object Oriented Programming

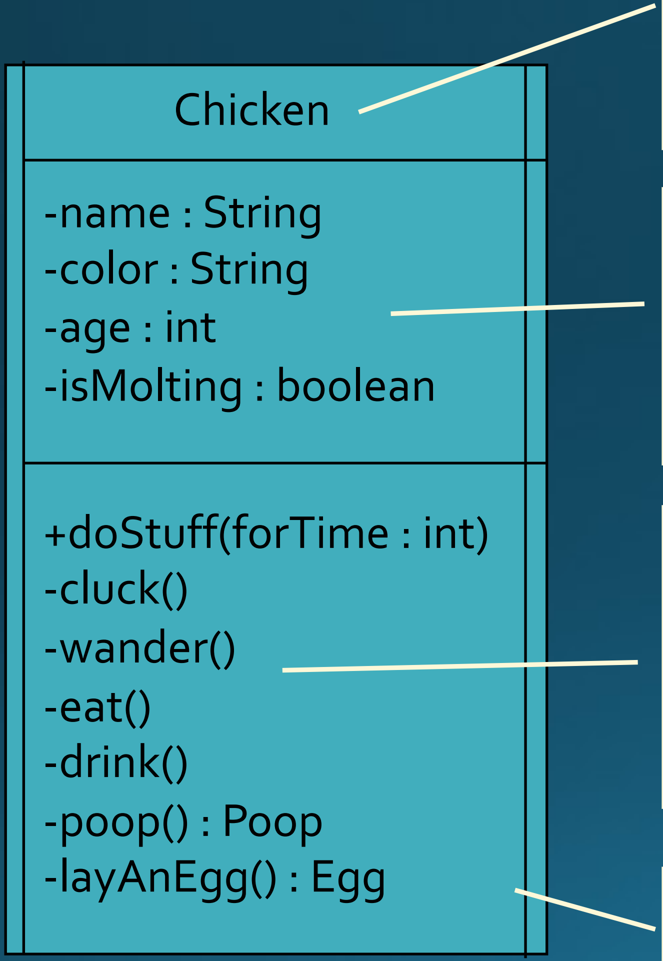
Edison Lascano

ESPE

Based on the slides of

Stephen Clyde Ph.D., Utah State University

Review of UML Class Notation



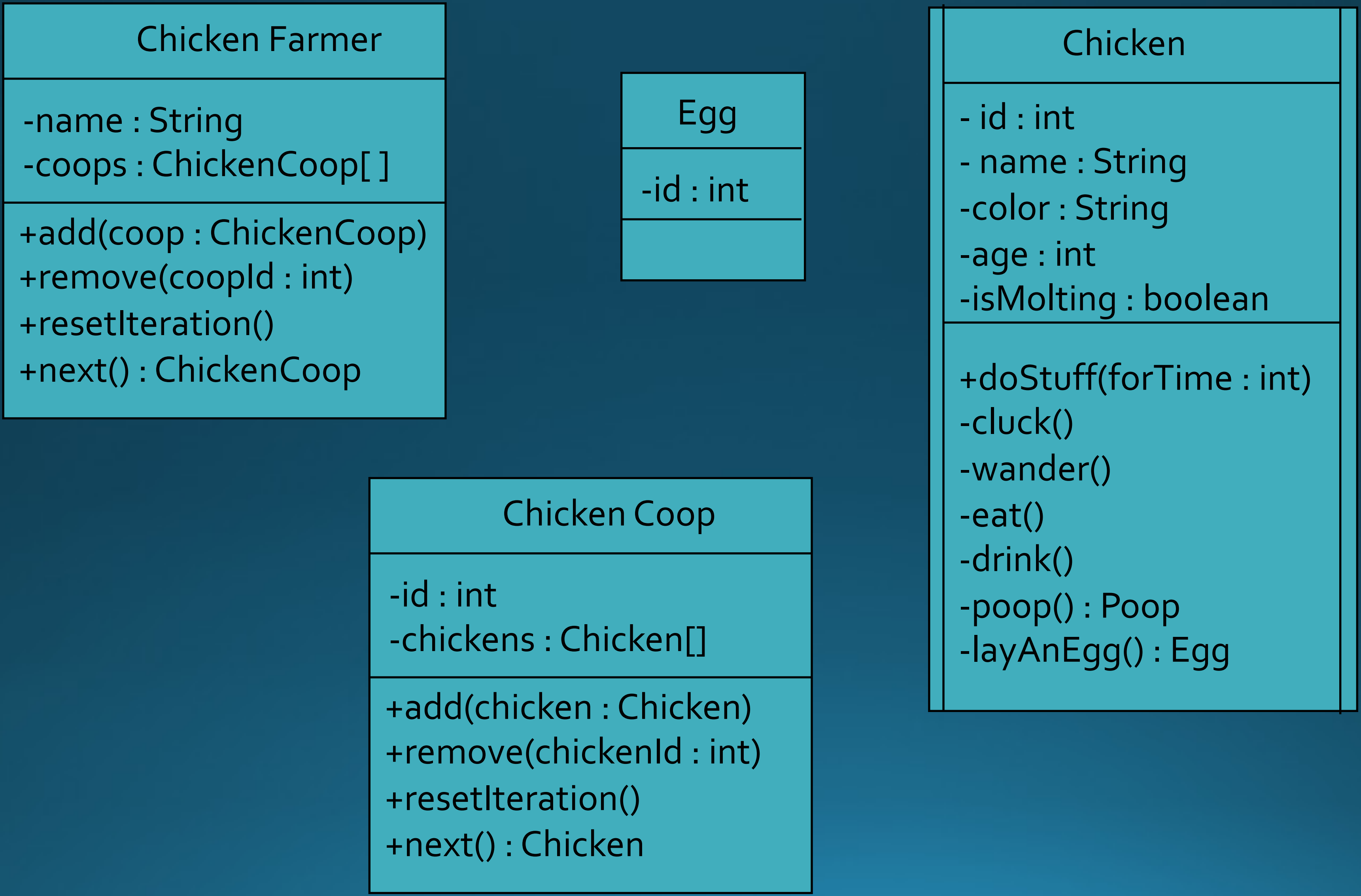
Class Name
(not underlined)
Use a noun or noun phrase that accurately represents a typical object in the class

Class Variables
(visibility label : data type)
The visibility should also be private, indicated by a “-”. The label should be a noun or noun phrase that describes the property the data member will capture. The data type can be any standard data type or the name of another class.

Methods
(visibility signature)
The visibility can be public “+” or private ”-”. Later we’ll introduce a protected visibility “#”. The signature includes a method name, an options list of parameters inside of ()’s and a a return type.

Active Indicator
(double left/right edge)

Examples



```
Chicken Farmer
-name : String
-coops : ChickenCoop[ ]
+add(coop : ChickenCoop)
+remove(coopId : int)
+resetIteration()
+next() : ChickenCoop
```

```
Egg
-id : int
```

```
Chicken
- id : int
- name : String
-color : String
-age : int
-isMolting : boolean
+doStuff(forTime : int)
-cluck()
-wander()
-eat()
-drink()
-poop() : Poop
-layAnEgg() : Egg
```

```
Chicken Coop
-id : int
-chickens : Chicken[]
+add(chicken : Chicken)
+remove(chickenId : int)
+resetIteration()
+next() : Chicken
```

Dependencies

A dependency indicates that one class's definition or implementation depends on another class

Represented by a dashed line with an arrowhead pointing to the second class

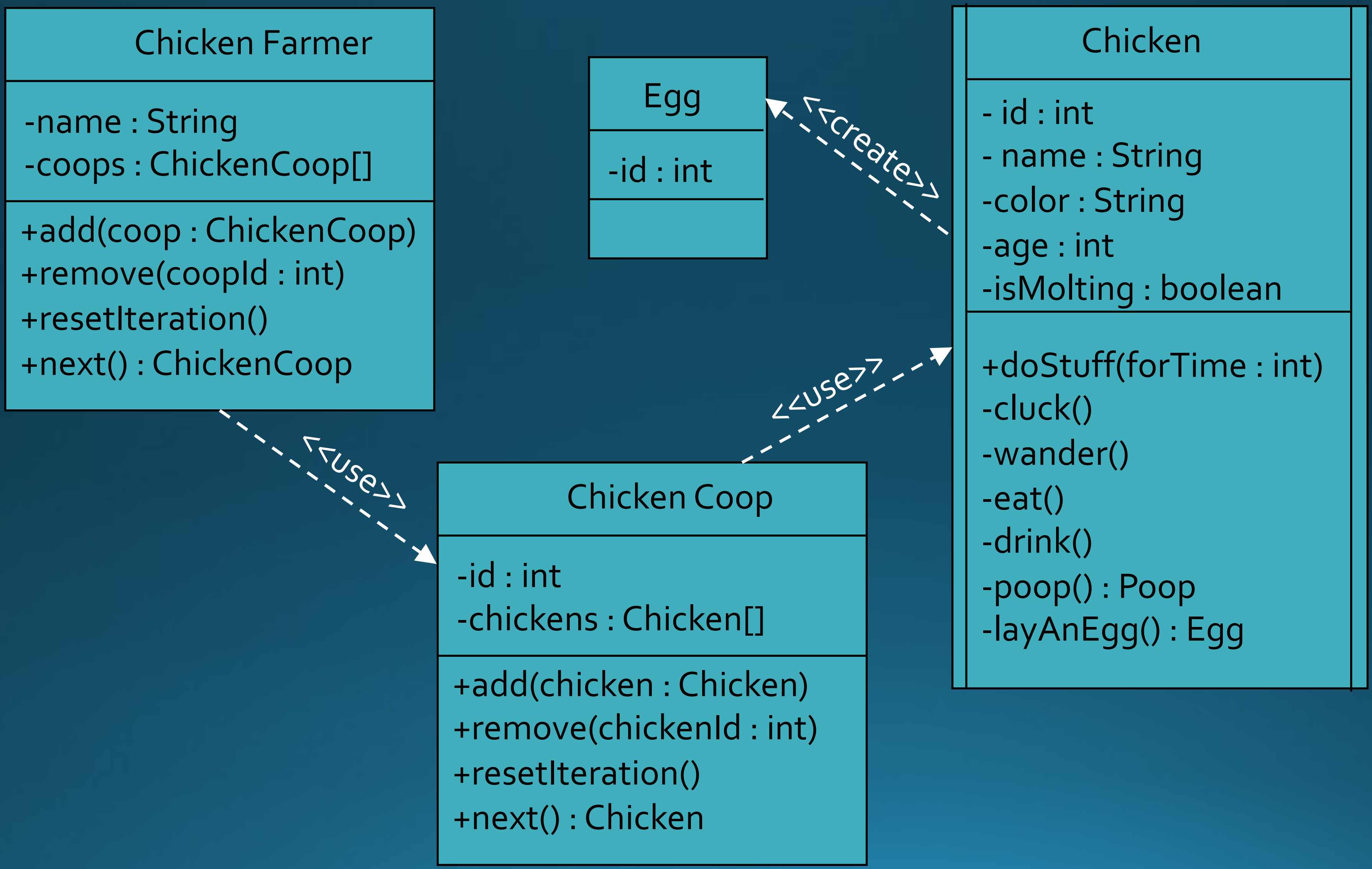
Dependencies can have an optional label with a name that explains how the first class uses the second class, e.g.

- calls some method m( )
- creates
- deletes

Special labels for dependency:

- <<use>> - means the 1st class uses the 2nd class in a data member definition or as method parameter
- <<create>> means that the 1st class instantiates (dynamically allocates) objects of the 2nd class.
- <<delete>> means that the 1st class deletes objects of the 2nd class.

Dependency


Chicken Farmer
-name : String
-coops : ChickenCoop[]
+add(coop : ChickenCoop)
+remove(coopId : int)
+resetIteration()
+next() : ChickenCoop
Egg
-id : int
<<create>>
Chicken
- id : int
- name : String
-color : String
-age : int
-isMolting : boolean
+doStuff(forTime : int)
-cluck()
-wander()
-eat()
-drink()
-poop() : Poop
-layAnEgg() : Egg
<<use>>
<<use>>
Chicken Coop
-id : int
-chickens : Chicken[]
+add(chicken : Chicken)
+remove(chickenId : int)
+resetIteration()
+next() : Chicken

Basic UML Classes → Java Classes

The UML class name becomes in the Java class name

- Remove spaces
- Each word, begins with a capital letter (UpperCamelCase)

The UML data members become the Java class data members (attributes)

- declared as private in the Java class
- remove spaces from label; use lowerCamelCase
  - depending on your naming convention
- use the data type in the declaration of the Java class variable

The UML methods become the Java class methods

- declared as public or private according to the design
- use the same signature as in the design (lowerCamelCase)

WHAT IS NEXT…

Modeling Practices