Objects and Classes

Object Oriented Programming

Edison Lascano
ESPE

Based on the slides of

Stephen Clyde Ph.D., Utah State University

Real-world Objects

In the real-world, what do you consider an object?

Examples?

Real-World Objects

- An object is anything that has a boundary in space or time
- Objects have state and behavior
  - An object's state includes anything and everything that describes it or its mode of operation
  - An object's behavior are the things it does in reaction to stimulus or on its own

My Chicken, Lucy

State
- Lucy
- brown & white
- 2 years old
- not molting

Behaviors
- cluck
- wander
- eat
- drink
- poop
- Lay an egg

Objects

- Every object is a unique entity in the universe, distinguished by its properties.
  - Many of those properties may not be observable or interesting to a software system
  - In a software system, even if two objects have the same known properties, they are not the "same" object
- Object may include other objects as part, e.g.
  - My car includes an engine and four distinct tires

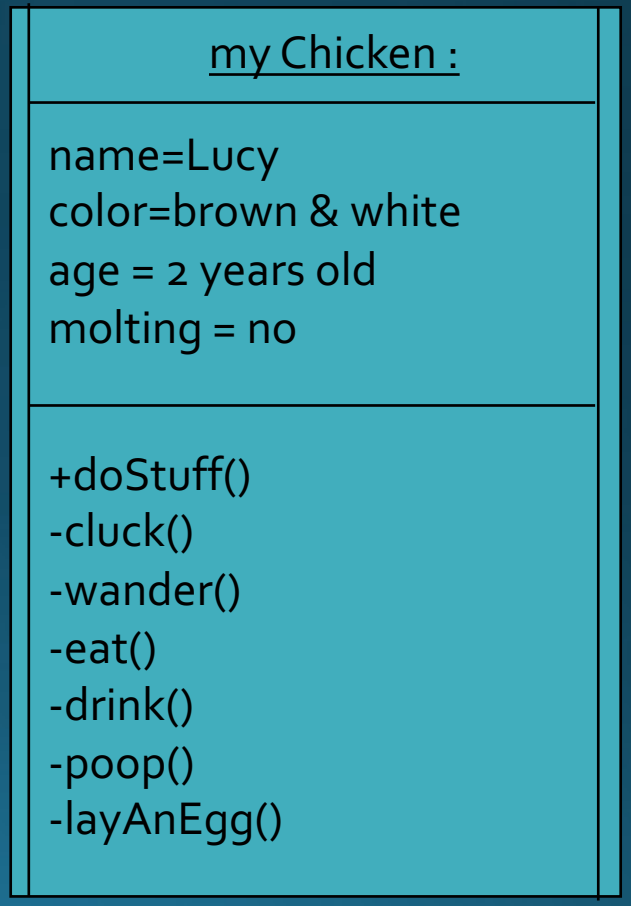
Software Objects

In software system, what is an object?

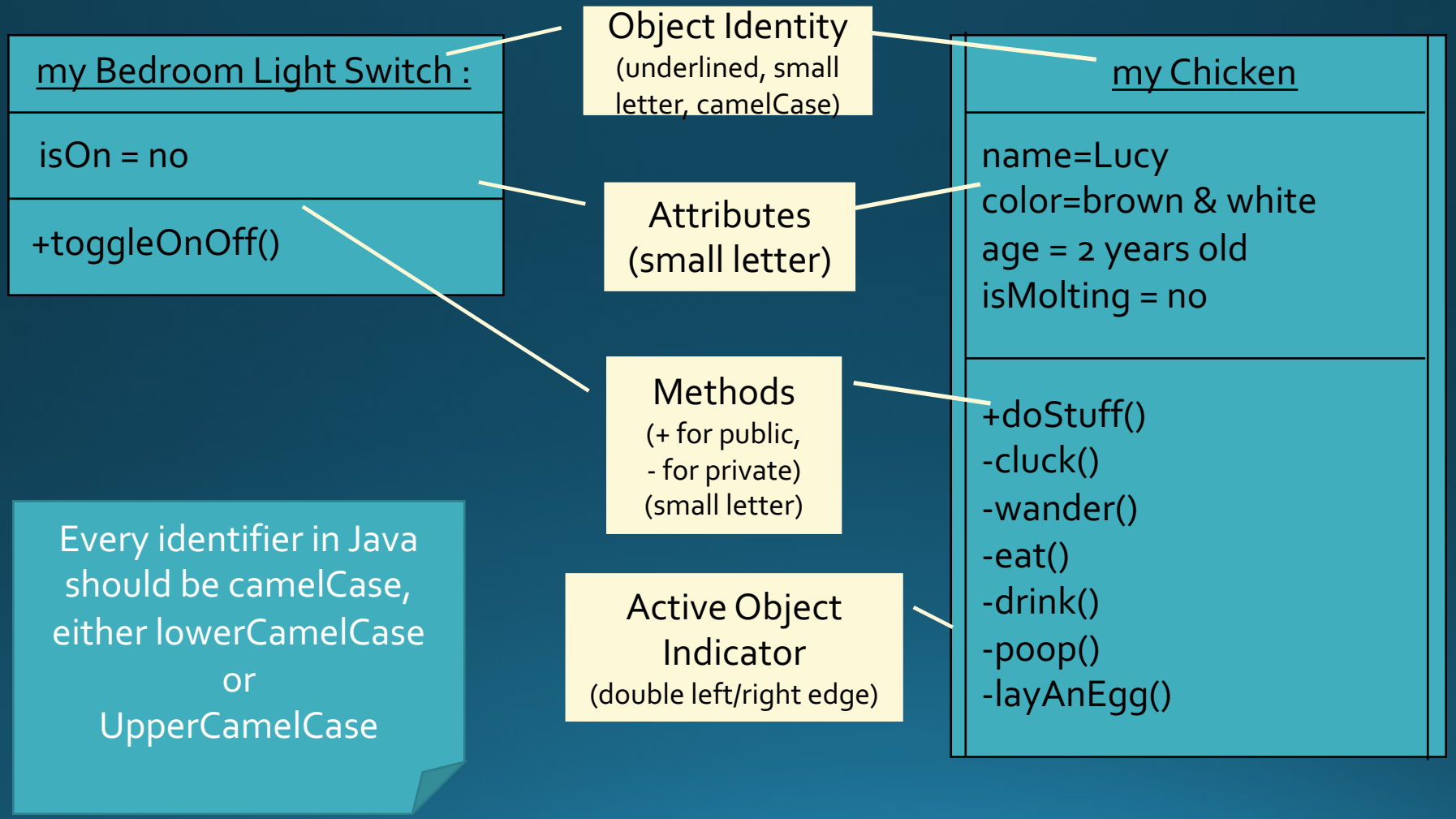
Examples?

Software Objects

- Conceptually, a software object is very similar to a real-world object
- It has
  - State, represented by data attributes (also called data members)
  - Behavior, represented by methods
- Software objects are loosely categorized as
  - Passive: only response to stimulus (e.g., method calls) from other objects
  - Reactive: in response to some stimulus, they stimulate behaviors in other objects
  - Active: do stuff (behavior) on their own



Software Objects



Software Objects

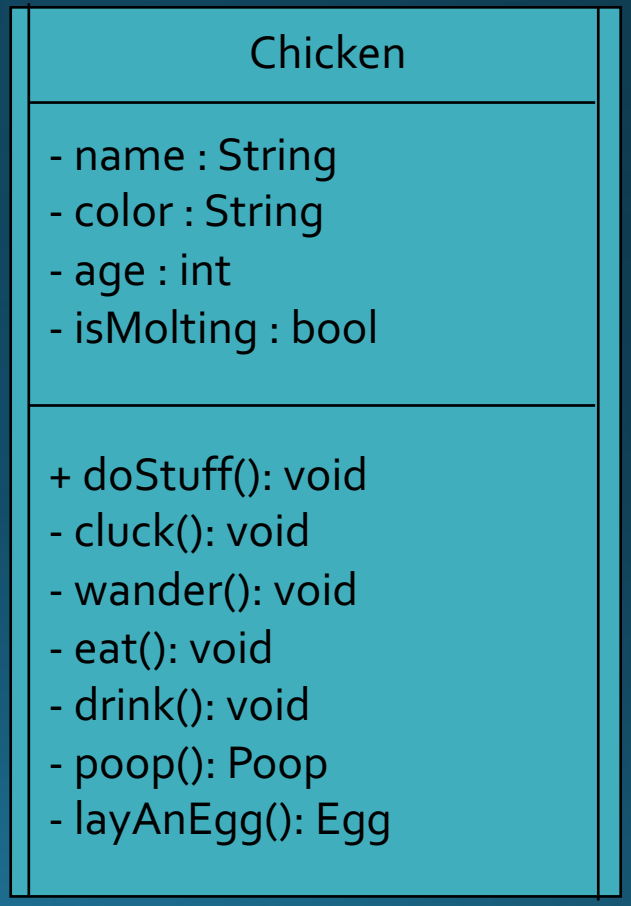
- How many objects can there be in a software system?
- Consider a system of managing your contacts
  - What would be some meaningful objects?
  - How many would the typical person have?
  - Do the object come and go?
- Consider a flight reservation system
  - What would be some meaningful objects?
  - At any given moment how many objects would the system need to know about?
- How can we manage complexity caused by having lots of software objects?

Classification

- Classification (the verb) is the process of grouping objects together into sets based on common properties
- A classification (the noun) or "class" is a set of objects that have the same kinds of attributes and methods
  - A class, e.g., Chicken, is a set of objects (instances)
  - All the objects in a class have certain types of attributes, e.g.,
    - All chickens have an "age" attribute
  - The objects in a class may have other attributes not defined by the class, e.g.,
    - My chicken has a name; not all chickens have a name
  - All the objects in a class have certain behaviors, e.g.
    - All chickens can eat
  - The objects in a class may have other methods not defined by the class, e.g.,
    - My chicken can wander
- In software, we use classification to better understand the problem and to help structure a solution

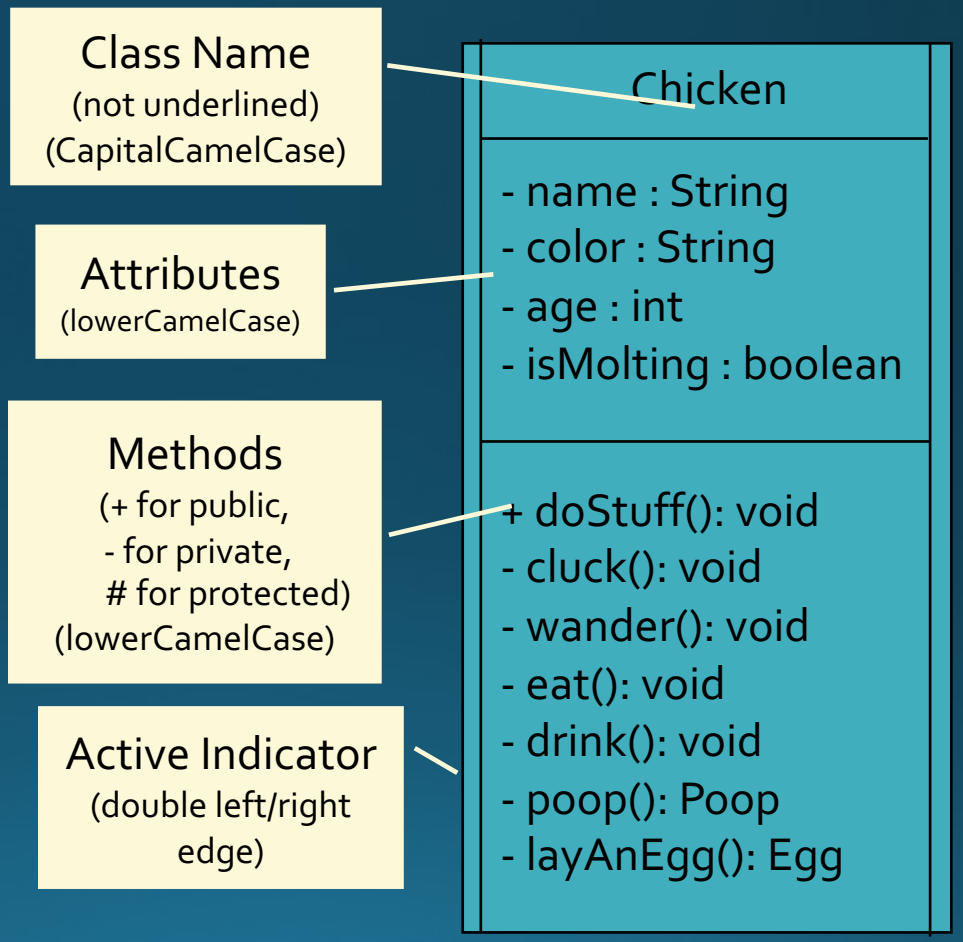
Software Classes

- In software, we use classification to better understand the problem and to help structure a solution
- We organization and manage objects with classes, such that
  - The class has a meaningful and cohesive purpose
  - The name of the class accurately and concisely represents a typical instance of the class
  - The attributes are closely related to the purpose of the class
  - The methods rely primarily on the attributes of the class



Software Classes

- Like objects, we can graphically represent classes with a box, divided into three sections
  - Name
  - class variables
  - Methods
- What is the difference between class variables for classes and attributes for objects?
- Since the methods apply to all objects in the class, they are defined and implemented at the class level



Java Classes

- The keyword "class" in Java comes from the notion of classification.
- Objects are instances of Java classes
- In Java, a class is defined by
  - The class variables that represent the common types of attributes for the instances of the class
  - The methods that can be applied to all instances of the class
- Java classes can be thought of as blueprints from which new instances of the class can be created

Finding and Designing Good Classes

- Finding potential classes via noun search
  - Document a high-level description of the system from a user's perspective
  - Look for key nouns and noun phrases and consider them as potential classes
  - Discard any that are outside the scope of work for the system you are building
- Finding potential classes via features or logic
  - Consider the requirements of the system
  - Look at the required features or chunks logically, and think about what is going to perform those operations
  - Group closely related features together
  - If they don't already belong to a class, consider whether they could represent a new class of objects whose responsibility is to perform those operations

Finding and Designing Good Classes

- Designing classes
  - For each class, consider the types of attributes (class variables) that are common to all instances of the class
  - Consider the behaviors (methods) that are common to all instances of the class
  - Think about the purpose of the class
    - If it is too complicated or diverse, consider splitting the class. The number of instances that will eventually be created for the class is not important. The number and complexity of the class variables and methods is important.)
    - If the class is trivial, consider merge it into a closely related class
  - Give the class a descriptive but concise name. If you are having trouble doing this, the purpose of the class may need to be refined

Finding and Designing Good Classes

- Refining classes so they become good classes
  - Review the class variables of each class, making sure that each pertains to every instance of that class. If necessary, move class variables to another class
  - Review the class variables of each class, making sure they have descriptive and concise names that are nouns or noun phrases
  - Review the class variables data types, making sure that each data member has an appropriate data type. Note that a data member's data type could be another class, meaning that the attributes (values of the data members for an object) will be an object of that other class.
  - Review the methods of each class, making sure that each relies primarily on the class variables of the class
  - Review the methods of each class, making sure they have descriptive and concise names that are verbs or verb phrases

Finding and Designing Good Classes

- Finding and designing good classes is a critical, but difficult-to-learn skill
- It takes practice!

WHAT IS NEXT…

More about Classes → Dependencies