Objects & Classes !

CPSC 1181 – O.O.

Jeremy Hilliker

Summer 2017

Langara.
THE COLLEGE OF HIGHER LEARNING.

What is an Object?

- Without being technical: it's a thing.
- What's a thing?
  - This is actually a profoundly difficult question.
  - We could try saying "something given a noun."
    - Verbs, adverbs, and adjectives are out.
    - What about non-physical things like: love, pain, anger, fear?
  - We could try saying "a physical thing,"
    - but what about virtual things?
    - Money (in your hand) is a thing
    - But money in your bank account is not a thing
      - It's an attribute… more like an adjective.

Are these Objects?

A pen	The upper ¼ of a pen
	The clicky part
A keyboard	A key, the keys, the set of keys
	The air above the keyboard
A shoe	A shoe's size
	The colour blue
A desk	All desks in the world
	The set of all desks in the world
A rocket	The first stage of a rocket
	The fuel in the rocket
A String	A substring
	The string's length
A date, a time, a DOB, an account	

What is Object Oriented?

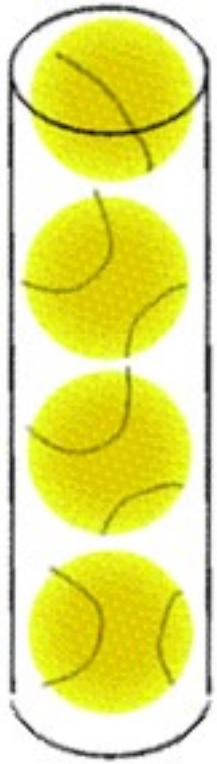
- A technique for system-modeling
- A model is an abstraction of something. It helps us understand and represent aspects of reality

What is an Object?

- An object has:
  - **Identity**
    - Acts as a single whole
    - One is “different” from another [or: not identically equal]
    - Has some kind of identifier / name (even “anonymous” ones)
  - **State**
    - Has properties / attributes that belong to the identity
    - May (mutable) or may not (immutable) change
  - **Behaviour**
    - Can do things
    - Can have things done to it

Tube of Tennis Balls

- Consider a tube of 4 yellow tennis balls
  - Is the tube of balls an object?
    - What is it's state?
    - What are it's behaviours?
    - Does it have an identity?
      - Bonus: If so, what's it's identifier?
  - Is each ball an object?
    - What is it's attributes?
    - What's one's identifier?
  - Are the two top balls an object?
  - Is the colour of the balls an object?
  - What about the colour of one ball?
  - Is you understanding of the balls an object?



Is the tube an object?

State:	Has 4 yellow balls, open/closed, capacity, etc
Behaviour:	Put ball, take ball, fill, empty
Identity:	[one is different from another]
Identity:	“The tube of balls on the previous slide”

Is each ball an object?

State:	Colour, wear, elasticity, used/new, chewed/not. Container?
Behaviour:	Bounce…
Identity:	Yes:
Identity:	“The 1st ball in the tube,” “The ball I am holding,” “The ball the dog ate”

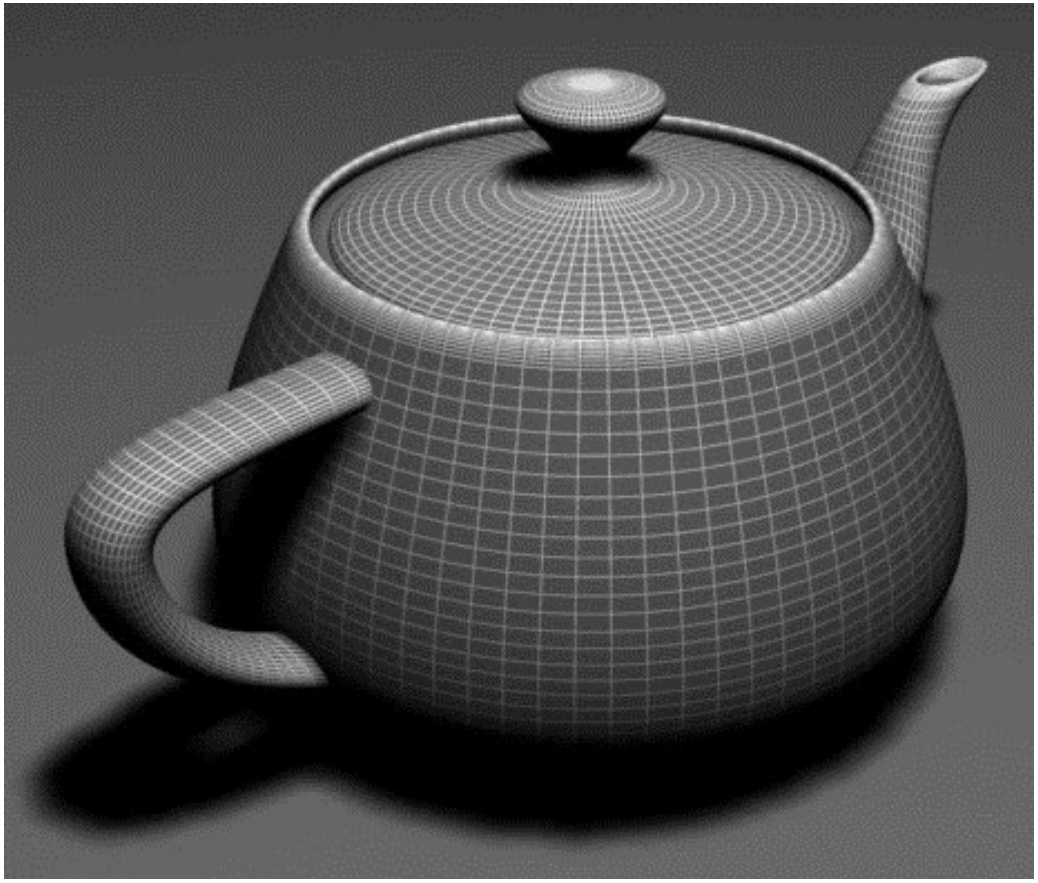
Others?

The top two balls:	Not ordinarily. But, if they were joined together by another object, then that object containing them would be. [A "pair"]
The colour of the balls: Or of one ball:	At the intro to programming level: "no, it's a property." At a higher level: it depends on if "colour" is a "first class citizen."
Is your "understanding" an object?	That's philosophical. Is it a "property" of your brain, or is it its own thing? What is "consciousness," "knowledge," or "understanding?"
Identity	Does it act as a whole? Maybe. Maybe not. Is it different from others? Yes. Does it have an identifier? Yes.
State	Yes… most likely.
Behaviour	Yes: Change, evaluate, decay, forget, confuse

Time

Identity	
Acts as a whole	11:45:00.000 PST
One is “different” from another	11:45 vs 10:30
Has some kind of identifier / name	11:45, lunch time, end of class
State	
Attributes	Hours, minutes, second, time zone
Behaviour	
	getHours, setMinutes, addMinutes isLocalized isBefore, equals timeBetween

A Teapot. Is it an object?



- A real teapot?
- The 3D rendering?
- This image of that rendering?
- What about that shadow?

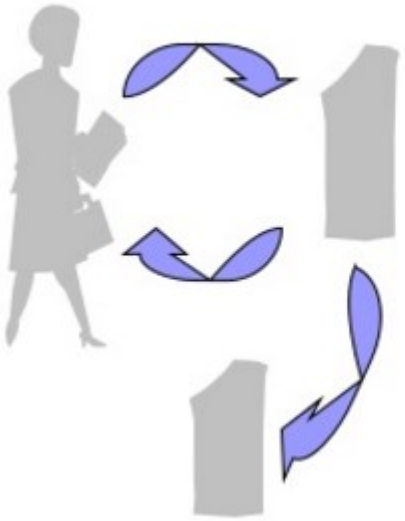
Object oriented programming

- Most programs do things that reflect some aspect of reality:
  - Eg: a bank **customer** can have **accounts**. The customer can deposit or withdraw *money* from an account, and can transfer money between accounts.
  - Note: “money” (here) is probably an attribute
    - not a “fist class citizen”
    - a first-class citizen (also type, object, entity, or value) in a given programming language is **an entity which supports all the operations generally available to other entities**. These operations typically include **being passed as an argument, returned from a function, and assigned to a variable**.

      -- Scott, Michael (2006). Programming Language Pragmatics. p. 140.

Procedural vs OO

- Procedural



Withdraw, deposit, transfer

- Object Oriented

Customer, money, account

- http://www.slideshare.net/smj/object-oriented-concept p. 13

Advantages of OO so far

- Natural: people think in terms of object
  - Model maps to reality [mostly]
  - Easier to develop [in most cases]
  - Easier to understand [in most cases]
- As an entity, an object “owns” its own state.
  - Exposes that state through its “contract.”
  - Makes debugging easier.

What is a Class?

- **Set of *objects* with the same set of *attributes* and *behaviours*.**
- That's a type!
  - set of values
    - set of (set of attributes)
  - operations on those values
    - the "behaviours" of the objects.

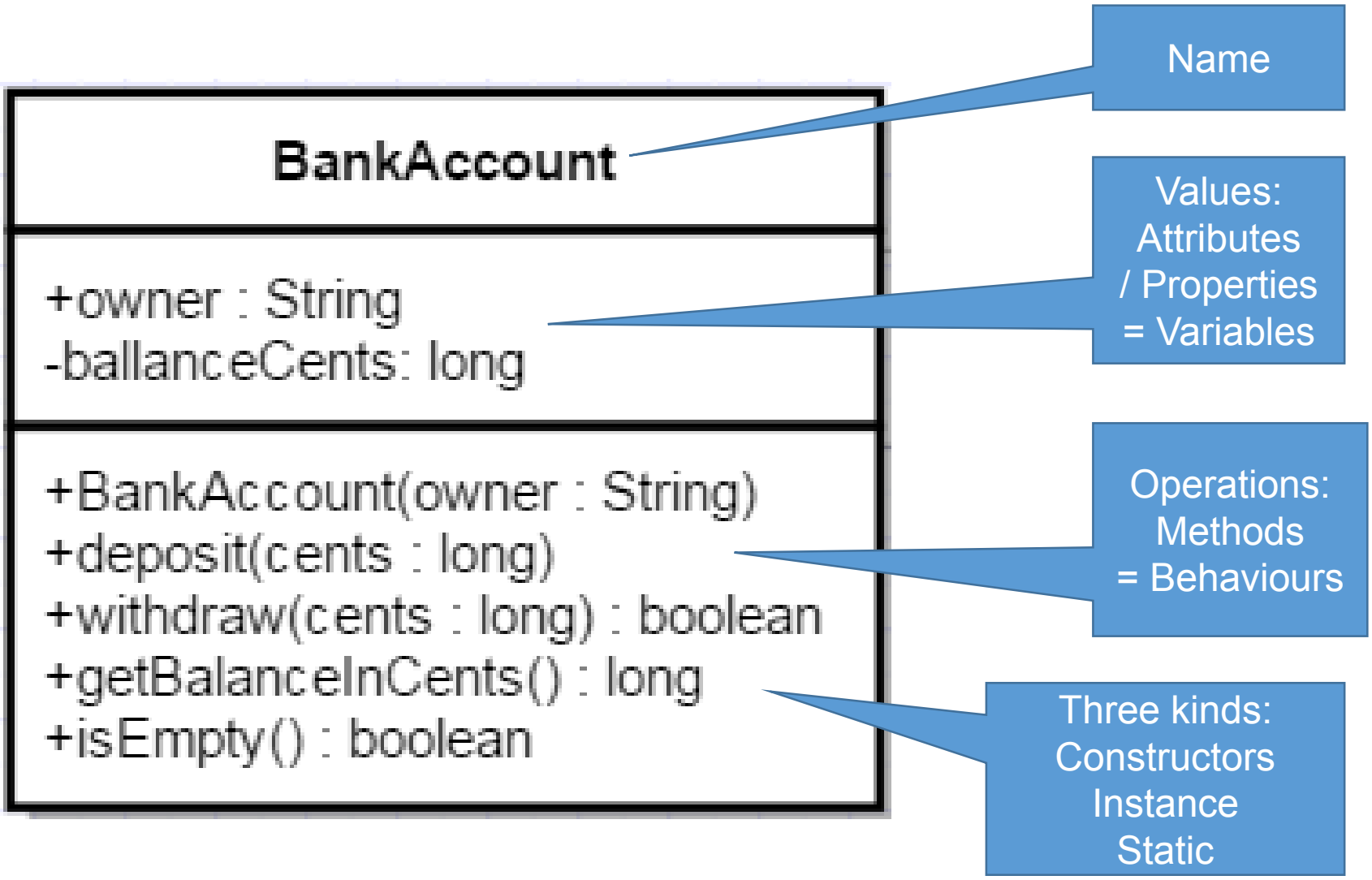
- Also sometimes described as
  - "A blueprint" for making objects (of that class)
  - A "contract" for the object's state and behaviours

Objects and Classes

- An object is an ***instance*** of a class
- Class:
  - Describes it's instances in general
  - Object a concrete instantiation of an element of the class

- Eg:
  - Class: String
  - Value: "a string"
  - Class: BankAccount
  - Value: Owner, balance

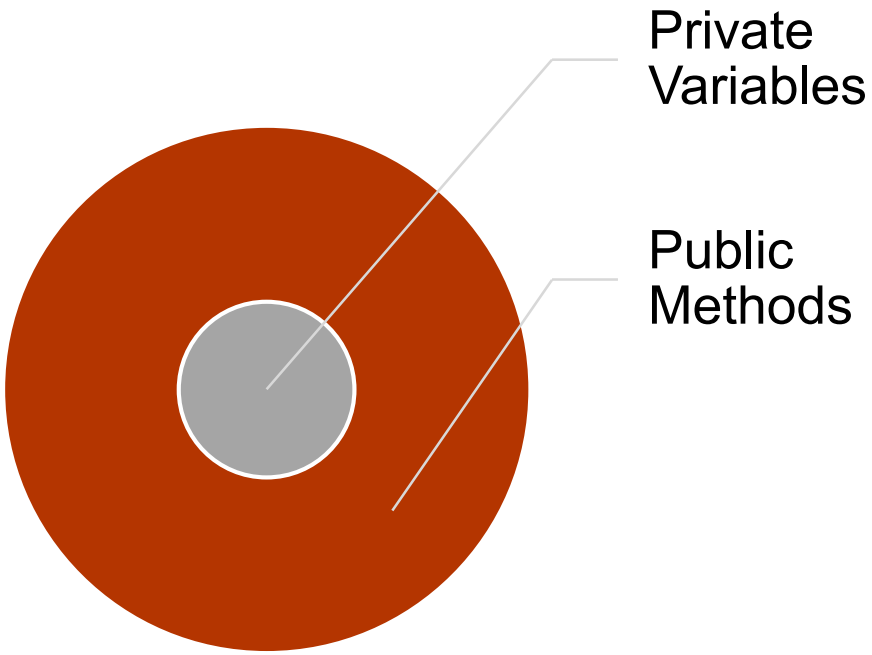
Elements of a Class:



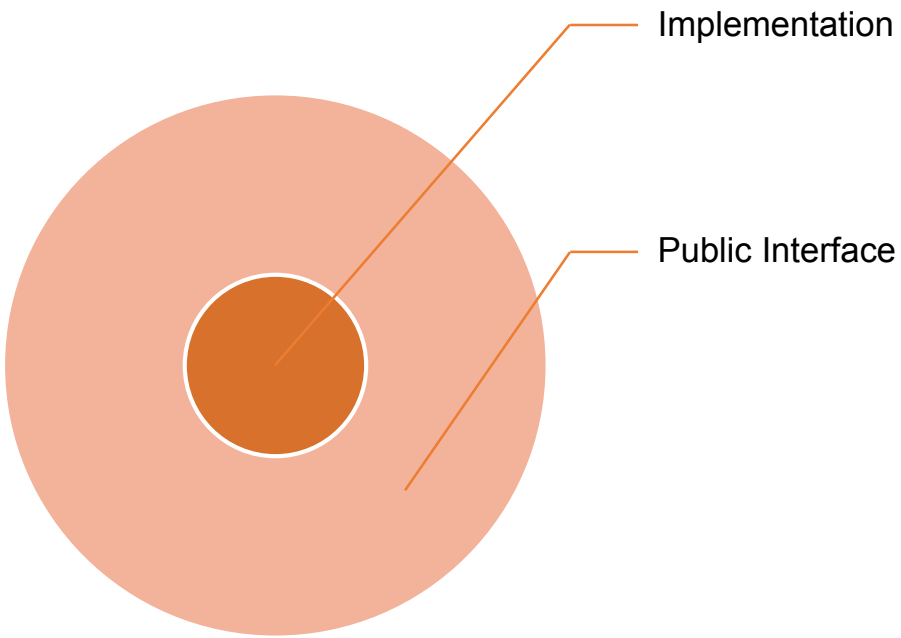
Three Big Ideas of O.O.

- **Encapsulation**:
  - restricting direct access to some of the object's components (ie: variables: data hiding)
  - bundling of data with the methods operating on that data (ie: a class)
- **Abstraction**:
  - Dealing with ideas rather than events
  - Providing functionality
  - Hiding implementation details
  - Know: what it does, not how it does it
  - “Design by Contract”
- **Polymorphism**
- [Inheritance (and to a lesser extent, composition)]

Encapsulation



Abstraction



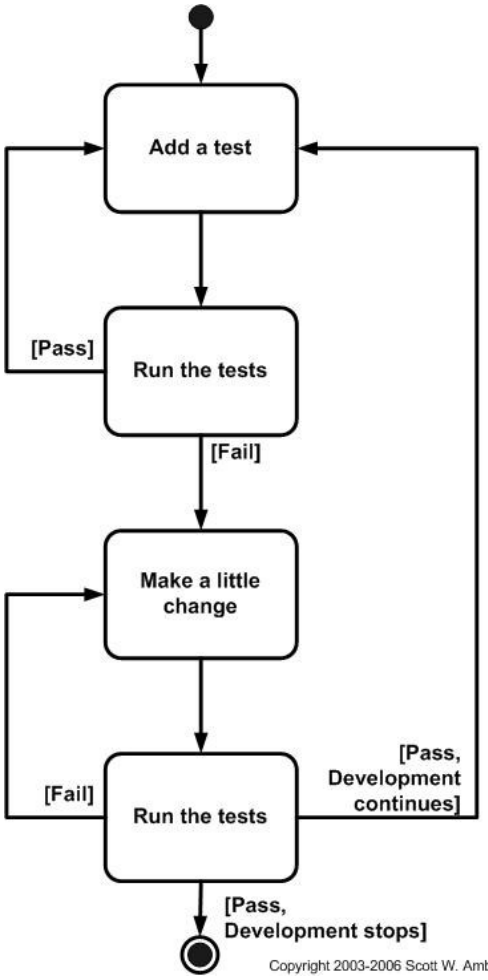
How to (Traditionally) Design a Class

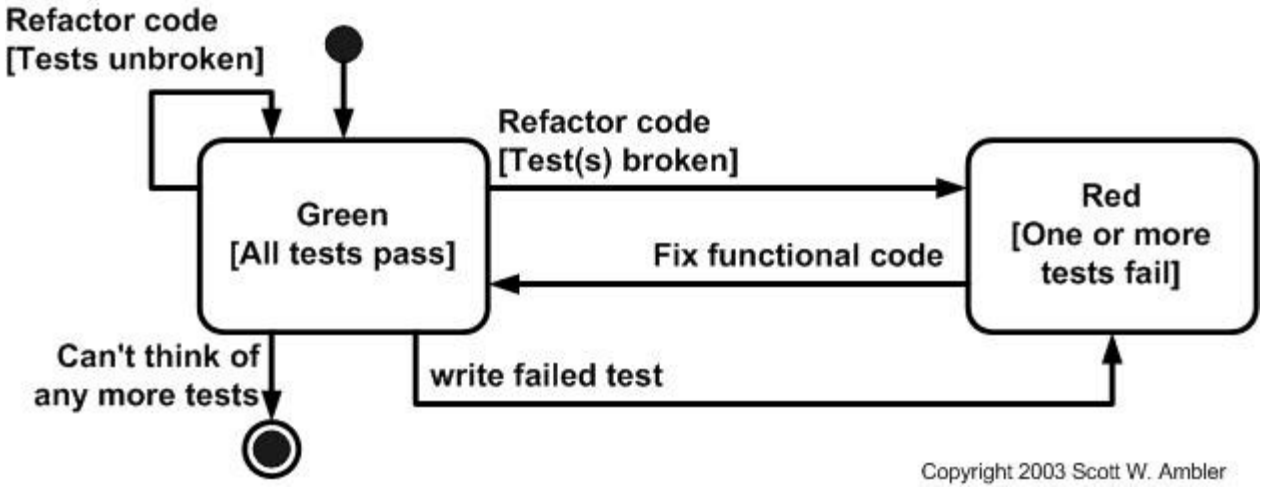
0. Think of a name for what you are making.
1. Find what (behaviours) you're asked to supply.
   - You don't have to model everything, only what is required.
2. Specify the public interface (the methods).
   - Determine method names, parameters, and return types.
3. Document your interface.
   - This actually matters. It makes you think about the interface and how it works and affects state. Plus you wont want to later.
4. Determine instance variables.
   - What does the object need to store?
   - Is that enough for each method (with parameters)?
5. Determine Constructors
6. Implement your methods
7. Test

Test Driven Development

If it's worth building, it's worth testing.

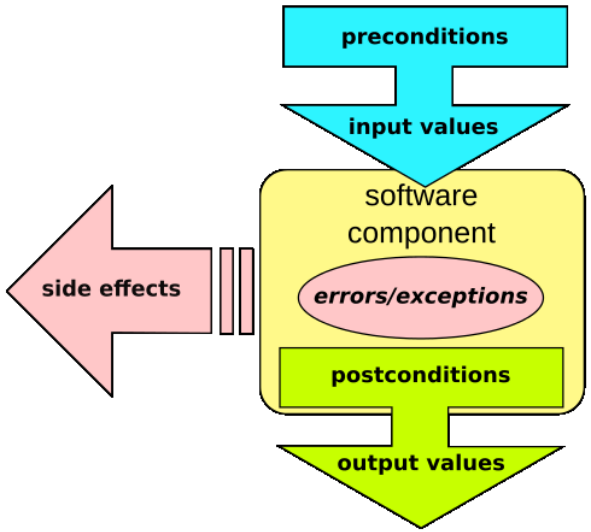
If it's not worth testing, why are you wasting your time on it?





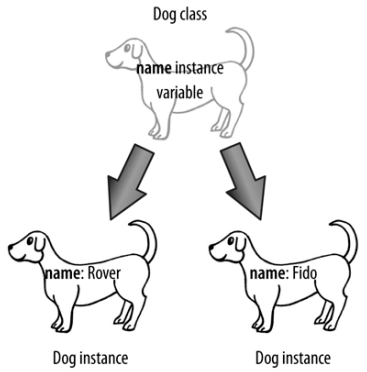
2) Public Interface

- Methods through which the object is manipulated
  - Encapsulation:
    - hiding the object's variables (private modifier)
      - Only* members of the class may access them directly
    - Grouping things together
  - Abstraction:
    - Hiding the details (the implementatio
  - You can (read: should) have private helper methods



4) Instance Variables

- Variables declared inside the class
- Each instance of an object will have their own set of them
- Private: can only be accessed by other instances of the same class
- For you, all instance variables should be private!



5) Constructors

- Method that *initializes* the instance variables of a newly constructed object
  - If you don't, sets them to defaults
    - numbers: 0
    - booleans: false
    - Objects: null
- Automatically called when object is first constructed
- Name: same name as the class
- Can be overloaded
  - If you don't provide any, the default constructor is provided
    - Default: no parameters