Classes

Objects and Primitives

- Our programs have worked with two types of data: ***primitives*** and ***objects***.
- Primitives are data types like **`int`**, **`double`**, **`char`**, and **`boolean`**.
  - They're built into Java – you can't define your own primitive types.
- Objects are types like `ArrayList`, `String`, `GPoint`, and `RandomGenerator`.
  - Where do these types come from?

Objects Revisited

- An object is a combination of
  - ***State*** – persistent information, and
  - ***Behavior*** – the ability to operate on that state.

- `GRect` state:
  - Position
  - Size
  - Color
  - Is filled?
  - etc.

- `GRect` behavior:
  - Move
  - Change color
  - Change fill state
  - Report position
  - etc.

Objects Revisited

- An object is a combination of
  - ***State*** – persistent information, and
  - ***Behavior*** – the ability to operate on that state.

- `GPoint` state:
  - Position
- `GPoint` behavior:
  - Move
  - Move by angle
  - Report *x* coordinate
  - Report *y* coordinate

Objects Revisited

- An object is a combination of
  - ***State*** – persistent information, and
  - ***Behavior*** – the ability to operate on that state.
    - `String` state:
      - Character sequence
    - `String` behavior:
      - Get characters
      - Produce substring
      - etc.

Classes and Objects

- Every object is an ***instance*** of a ***class***.
- The class determines
  - what state each instance maintains.
  - what behaviors each instance possesses.
- Each instance determines
  - the specific values for that state information.

class Dog

Has a fur color.
Has an energy level.
Has a level of cuteness.
Can be your friend.
Can sit.
Can stay.
Can bark.

Creating a Class

Creating our own Class

Image credit: http://store.controlconceptsusa.com/media/products/MC-Tally.jpg

Creating our own Class

- State:
  - The current number.

We use instance variables to keep track of state.

- Behavior:
  - Read the counter.
  - Increment the counter.

Creating our own Class

- State:
  - The current number.

- Behavior:
  - Read the counter.
  - Increment the counter.

We use instance variables to keep track of state.

We use methods to specify behavior.

Creating Objects

- Each object is an ***instance*** of a class.
- You can create an object that's an instance of a given type by writing

```
new Type(args)
```

- This is sometimes called ***instantiating*** the class.

Instance Variables Revisited

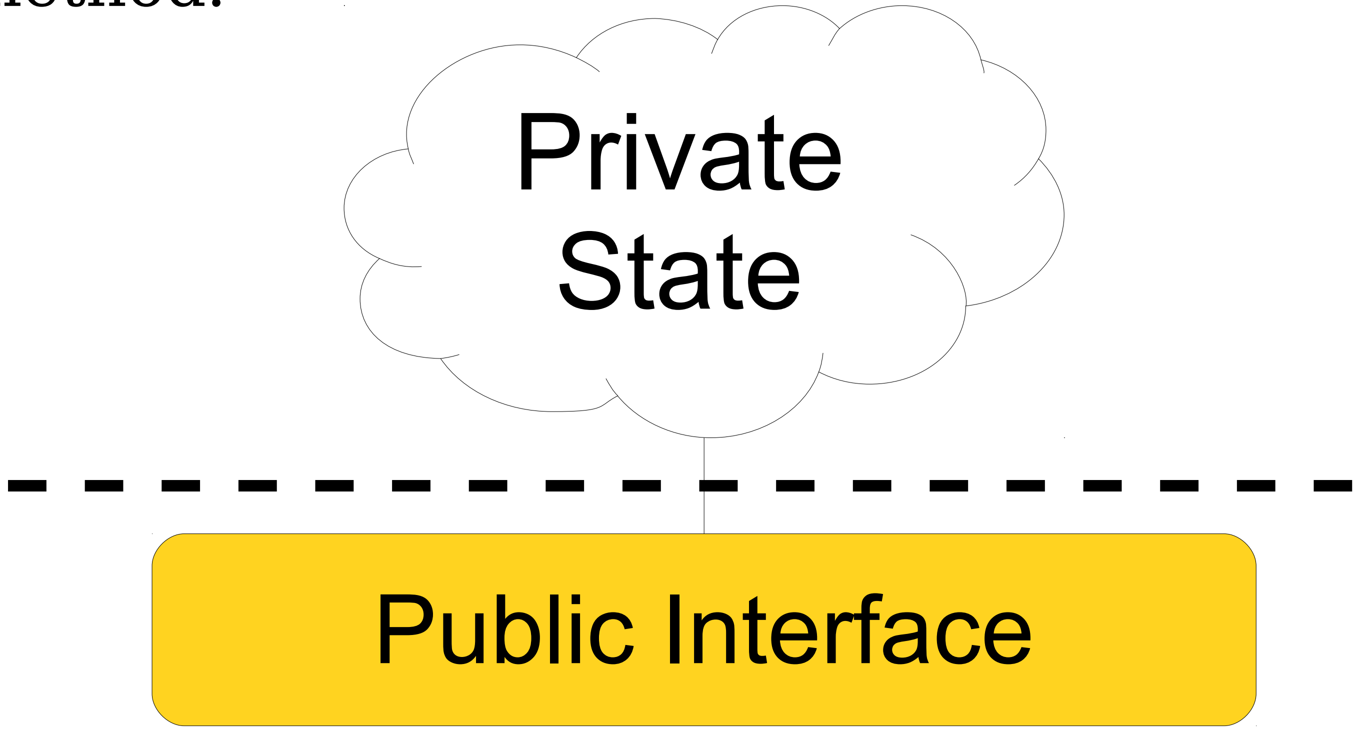
- Each instance of a class gets its own, unique copy of each instance variable.
- Each object's instance variables persist as long as the object exists.
- Different instances of the same object cannot read or write each others' instance variables.

`public` and `private`

- A method or instance variable declared **`public`** can be accessed from *anywhere*.
- A method or instance variable declared **`private`** can only be accessed by an instance of the class in the body of a method.

`public` and `private`

- A method or instance variable declared **`public`** can be accessed from *anywhere*.
- A method or instance variable declared **`private`** can only be accessed by an instance of the class in the body of a method.



Why Hide Information?

- Making instance variables private and mediating access through public methods has many advantages.
- Separates *what you can do* from *how it's done*:
  - We never talked about how `GOval` or `HashMap` actually work, but you can still use them.
- Prevents meaningless operations:
  - A counter may be *implemented* using an **`int`**, but it's *not* actually an **`int`** and not all operations on **`int`** make sense on a counter (or vice-versa).

Time-Out for Announcements!

Assignment 7

- Assignment 6 (Array Algorithms) due at 3:15PM today.
- Midterm regrades will be completed by Monday.
- Assignment 7 (**NameSurfer**) goes out today and is due Monday, March 9 at 3:15PM.
    - Play around with graphics, interactors, `HashMaps`, and classes!
    - See historical trends play out in baby name popularities!

Casual CS Dinner

- WiCS is holding their second Casual CS Dinner of the quarter next Wednesday at 6PM.
- Location info and RSVP link available in the email sent out yesterday.

Midterm Logistics

- Second midterm is next Tuesday from 7PM – 10PM.
- Same locations as last time – just go where you went before!
  - `Abb  - Jon:` Go to **Hewlett 200**
  - `Jun  - Mari:` Go to **Hewlett 201**
  - `Marq - Mik:` Go to **Hewlett 101**
  - `Mil  - Ogr:` Go to **Hewlett 102**
  - `Oke  - Pat:` Go to **Hewlett 103**
  - `Pau  - Tan:` Go to **Braun Auditorium**
  - `Tao  - Zuc:` Go to **320-105**
- Good luck!

Back to CS106A!

Modifying our Class

Constructors

- A ***constructor*** is a special method defined in a class that is responsible for setting up class's instance variables to appropriate values.
- Syntax:

```
public NameOfClass(arguments) {
        /* … body of constructor … */
}
```

- Inside a constructor:
  - Give initial values to instance variables.
  - Set up instance variables based on values specified in the parameters.
- Constructor called when instance created with `new`.

toString()

- To get a string representation of an object, Java uses a method

```
public String toString()
```

- If you define this method in your Java classes, you can customize what string will be produced.
- Otherwise, you get Icky Javaspeak string representations.