Classes

Some Quick Thoughts

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
      - Tell 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
      - Tell x
      - Tell y

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

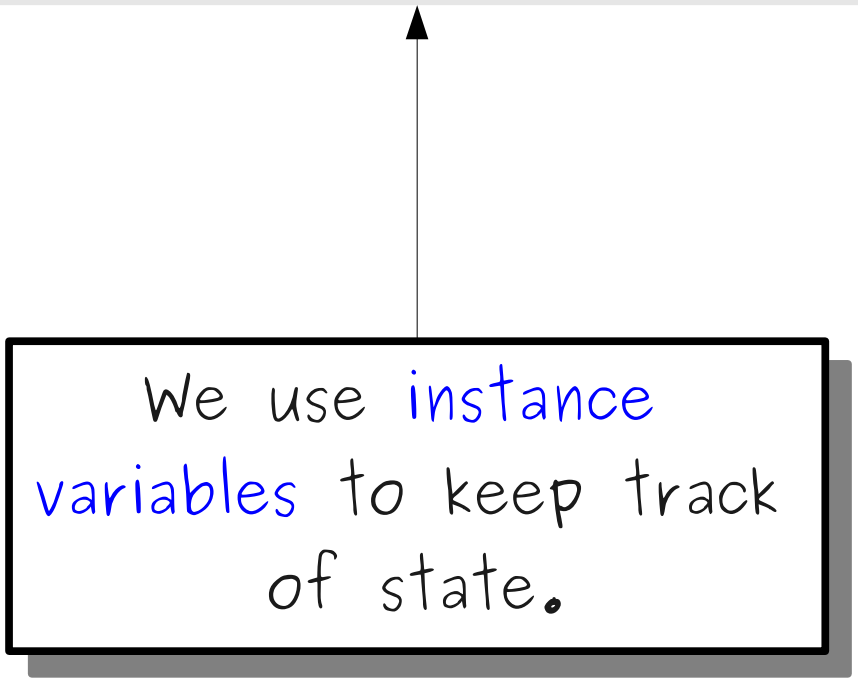
- Each object is an **instance** of some **class**.
- The class determines
  - what state each instance maintains.
  - what behaviors each instance possesses.
- Each instance determines
  - the specific values for that state information.

Creating our own Class

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

Creating our own Class

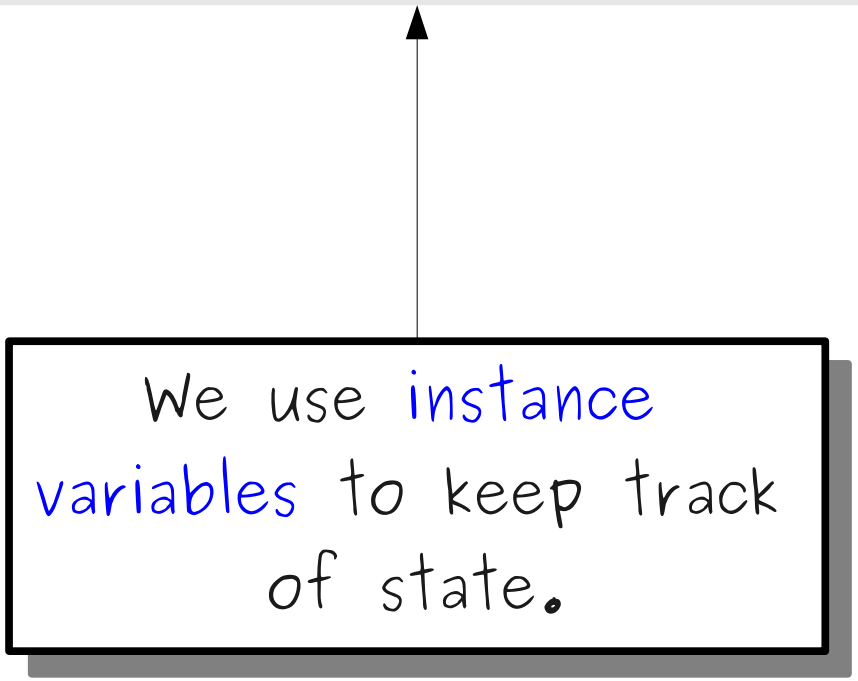
- State:
  - The current number.
- Behavior:
  - Read the counter.
  - Increment the counter.

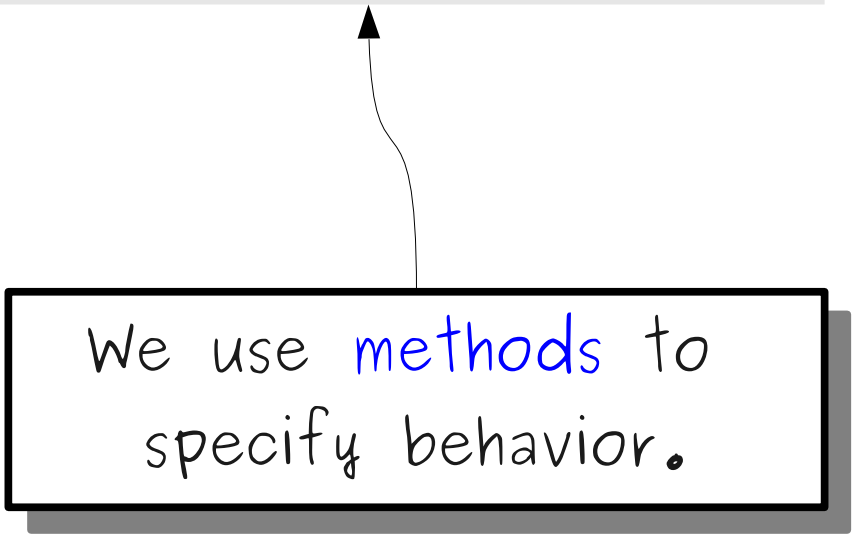


Creating our own Class

- State:
  - The current number.

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



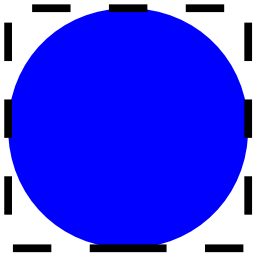


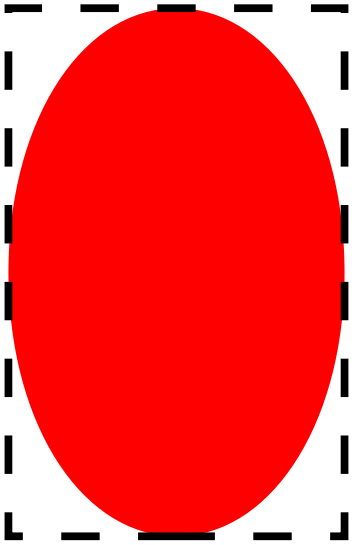
Instance Variables Revisited

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

Instance Variables Revisited

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





Time-Out for Announcements!

Midterm Logistics

- Midterm next **Wednesday, March 5** from 7PM – 10PM.
- Room locations divvied up by last name:
  - Abr - Che: Gates B01
  - Chi - Erd: Gates B03
  - Esp - Fre: Gates B12
  - Fu - Kea: SkillAud
  - Kel - Lim: HerrinT175
  - Lin - Oul: Hewlett201
  - Pad - Ren: 200-205
  - Rey - San: 380-380W
  - Sar - Sta: 380-380X
  - Ste - Tse: 380-380Y
  - Tsk - Zhu: 420-041

More Midterm Logistics

- Solutions to first practice exam released; second practice exam available.
- Review session this Sunday, March 2 from 1PM – 3PM in Hewlett 200.

Regrades Processed

- All exam regrades have been processed.
- Available for pickup later today in a specially-marked folder in the midterm return filing cabinet.

Assignment 6

- Assignment 5 due at 3:15PM today.
  - Due Monday with one late period, Wednesday with two, and Friday with three.
- Assignment 6 (**NameSurfer**) goes out today. It's due on Wednesday, March 12 at 3:15PM.
  - Play around with `HashMaps`, `ArrayLists`, arrays, file processing, graphics, interactors, and classes!
  - Assignment review session next Thursday from 5:30PM – 6:30PM in Hewlett 200.

Back to CS106A!

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(parameters) {
  /* … 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()
```

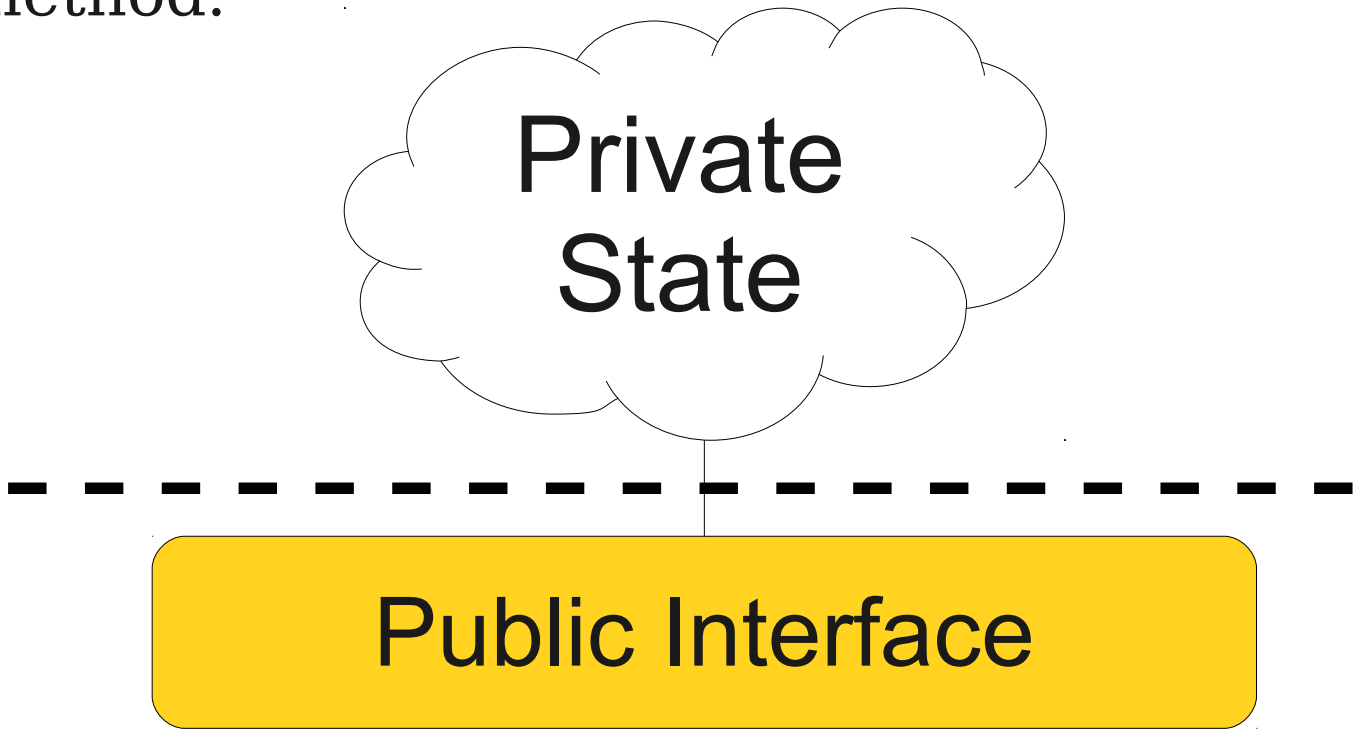
- If you define this method in your Java classes, you can customize what string will be produced.
- Otherwise, you get Icky Javaspeak string representations.

`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).