- Previous lecture:
  - Review cell and struct arrays
  - File i/o and built-in function `sort`
- Today's lecture:
  - Introduction to objects and classes
- Announcements:
  - Project 5 due Friday at 11pm
  - Discussion in computer lab Upson B7 this week
  - Prelim 2 on Tues, Apr 21 at 7:30pm
  - Prelim 2 topics: end with Project 5 and Lecture and discussion of previous week; will NOT include OOP

Different kinds of abstraction

- Packaging procedures (program instructions) into a function
  - A program is a set of functions executed in the specified order
  - Data is passed to (and from) each function
- Packaging data into a structure
  - Elevates thinking
  - Reduces the number of variables being passed to and from functions
- Packaging data, and the instructions that work on those data, into an object
  - A program is the interaction among objects
  - Object-oriented programming (OOP) focuses on the design of data-instructions groupings

A card game, developed in two ways

- Develop the algorithm—the logic—of the card game:
  - Set up a deck as an array of cards. (First, choose representation of cards.)
  - Shuffle the cards
  - Deal cards to players
  - Evaluate each player's hand to determine winner

Procedural programming: focus on the algorithm, i.e., the procedures, necessary for solving a problem

- Identify "objects" in the game and define each:
  - Card
    - Properties: suit, rank
    - Actions: compare, show
  - Deck
    - Property: array of Cards
    - Actions: shuffle, deal, get #cards left
  - Hand …
  - Player …
- Then write the game—the algorithm—using objects of the above "classes"

A card game, developed in two ways

- Develop the algorithm—the logic—of the card game:
  - Set up a deck as an array of cards. (First, choose representation of cards.)
  - Shuffle the cards
  - Deal cards to players
  - Evaluate each player's hand to determine winner

Procedural programming: focus on the algorithm, i.e., the procedures, necessary for solving a problem

- Identify "objects" in the game and define each:
  - Card
    - Properties: suit, rank
    - Actions: compare, show
  - Deck
    - Property: array of Cards
    - Actions: shuffle, deal, get #cards left
  - Hand …
  - Player …
- T
  al
  th

Object-oriented programming: focus on the design of the objects (data + actions) necessary for solving a problem

Notice the two steps involved in OOP?

- Define the classes (of the objects)
  - Identify the properties (data) and actions (methods, i.e., functions) of each class

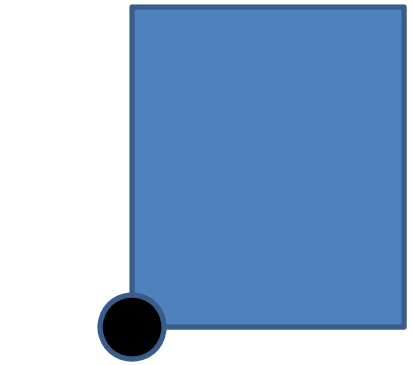
- Create the objects (from the classes) that are then used—that interact with one another

Defining a class ≠ creating an object

- A class is a specification
  - E.g., a cookie cutter specifies the shape of a cookie

- An object is a concrete instance of the class
  - Need to apply the cookie cutter to get a cookie (an instance, the object)
  - Many instances (cookies) can be made using the class (cookie cutter)
  - Instances do not interfere with one another. E.g., biting the head off one cookie doesn't remove the heads of the other cookies

Example class: Rectangle



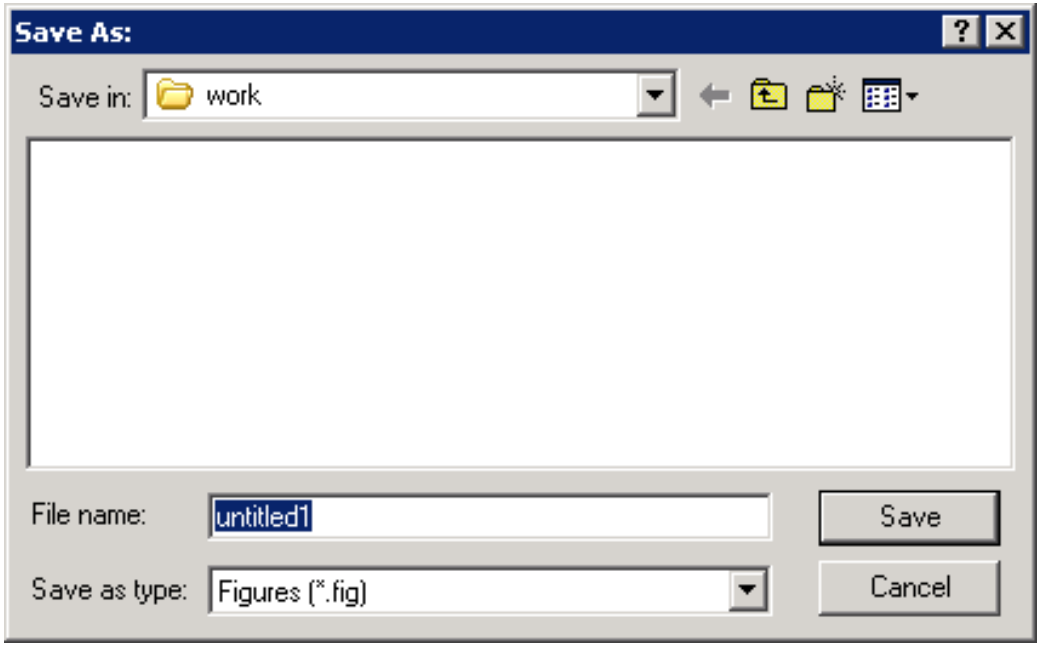
- Properties:
  - xLL, yLL, width, height
- Methods (actions):
  - Calculate area
  - Calculate perimeter
  - Draw
  - Intersect (the intersection between two rectangles is a rectangle!)

Example class: Time

- Properties:
  - Hour, minute, second
- Methods (actions):
  - Show (e.g., display in hh:mm:ss format)
  - Advance (e.g., advance current time by some amount)

Example class: Window (e.g., dialog box)

- Properties:
  - Title, option buttons, input dialog …
- Methods (actions):
  - Show
  - Resize
  - …



Many such useful classes have been predefined!

Matlab supports procedural and object-oriented programming

- We have been writing procedural programs—focusing on the algorithm, implemented as a set of functions
- We have used objects in Matlab as well, e.g., graphics
- A plot is a "*handle graphics*" object
  - Can produce plots without knowing about objects
  - Knowing about objects gives more possibilities

The `plot` handle graphics object in Matlab

```
x=…; y=…;
plot(x,y)
```
creates a graphics object

- In the past we focused on the visual produced by that command. If we want the visual to look different we make *another* plot.
- We can actually "hold on" to the graphics object—store its "*handle*"—so that we can later make changes to *that* object.

Objects of the same class have the same properties

```
x= 1:10;
% Two separate graphics objects:
plot(x, sin(x), 'k-')
plot(x(1:5), 2.^x, 'm-*')
```

- Both objects have some x-data, some y-data, some line style, and some marker style. These are the properties of one kind, or class, of the objects (plots)
- The values of the properties are different for the individual objects

See **demoPlotObj.m**

Object-Oriented Programming

- First design and define the classes (of the objects)
  - Identify the properties (data) and actions (methods, i.e., functions) of each class

- Then create the objects (from the classes) that are then used, that interact with one another

Class Interval

- An interval has two properties:
  - left, right
- Actions—methods—of an interval include
  - Scale, i.e., expand
  - Shift
  - Add one interval to another
  - Check if one interval is in another
  - Check if one interval overlaps with another

See **`demoInterval0.m`**

Class Interval

- An interval has two prope
  - left, right
- Actions—methods—of an
  - Scale, i.e., expand
  - Shift
  - Add one interval to anothe
  - Check if one interval is in a
  - Check if one interval overla

To specify the properties and actions of an object is to define its class

```
classdef Interval < handle

  properties
    left
    right
  end

  methods
    function scale(self, f)
      …
    end

    function shift(self, s)
      …
    end

    function Inter = overlap(self, other)
      …
    end

    function Inter = add(self, other)
      …
    end

    …

  end

end
```

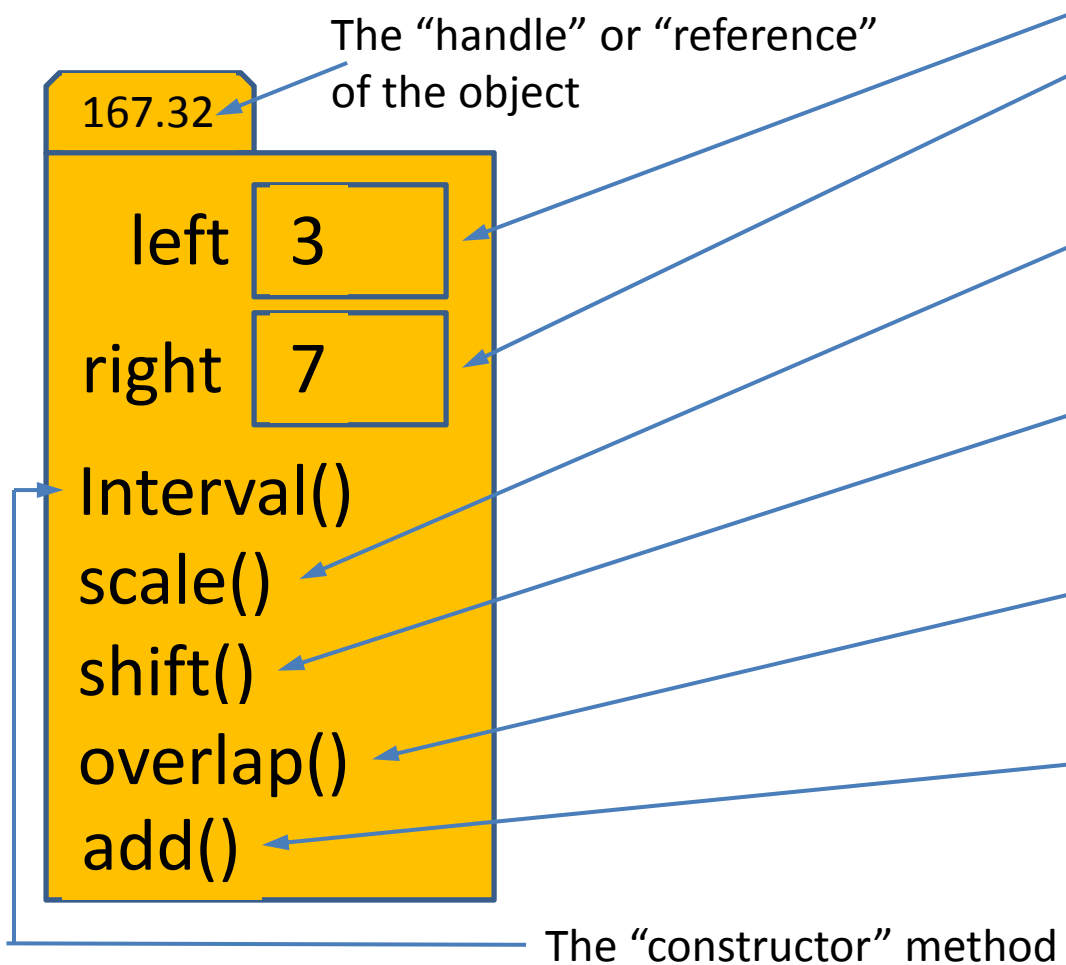
Given class Interval (file Interval.m) …

```
% Create 2 Intervals, call them A, B
A= Interval(2,4.5)
B= Interval(-3,1)

% Assign another right end point to A
A.right= 14

% Half the width of A (scale by .5)
A.scale(.5)

% See the result
disp(A.right) % show value in right property in A
disp(A)       % show all property values in A
disp(B)
```

Observations:

- Each object is referenced by a name.
- Two objects of same class has same properties (and methods).
- To access a property value, you have to specify whose property (which object's property) using the dot notation.
- Changing the property values of one object doesn't affect the property values of another object.

An Interval object




```
classdef Interval < handle

  properties
    left
    right
  end

  methods
    function scale(self, f)
      ...
    end

    function shift(self, s)
      ...
    end

    function Inter = overlap(self, other)
      ...
    end

    function Inter = add(self, other)
      ...
    end

    ...

  end

end
```

An object is also called an "instance" of a class. It contains every property, "instance variable," and every "instance method" defined in the class.