• Previous lecture:
  – Introduction to objects and classes
• Today’s lecture:
  – Defining a class
    • Properties
    • Constructor and instance methods
  – Objects are passed by reference to functions
• Announcements:
  – Discussion this week in the lab. At the beginning of your discussion section, hand in (on paper) your answer to Problem 3 of last week’s discussion exercise.
  – 2nd optional review session: W 8-9:30pm in Hollister B14
  – Prelim 2: Thurs at 7:30pm

Object-Oriented Programming
• First design and define the classes (of the objects)
  – Define the properties (data) and actions (methods, i.e., functions) of each class in a “class definition file”
• 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

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 Interval A
A.right= 14

% Half the width of A (scale by 0.5)
A.scale(.5)

% See the result
disp(A) % show all property values in A
disp(B)
An Interval object

The "constructor" or "reference" of the object

```
classdef Interval < handle
    properties
        left
        right
    end
    methods
        function Inter = Interval(lt, rt)
            Inter.left = lt;
            Inter.right = rt;
        end
        function scale(self, f)
            w = self.right - self.left;
            self.right = self.left + w*f;
        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.

Multiple Interval objects

```
classdef Interval < handle
    properties
        left
        right
    end
    methods
        function Inter = overlap(self, other)
            % overlap
        end
        function Inter = add(self, other)
            % add
        end
    end
end
```

Every object (instance) contains every "instance variable" and every "instance method" defined in the class. Every object has its own handle.

The constructor method

To create an Interval object, use its class name as a function call: `p = Interval(3,7)`

```
classdef Interval < handle
    % An Interval has a left end and a right end
    properties
        left
        right
    end
    methods
        function Inter = Interval(lt, rt)
            Inter.left = lt;
            Inter.right = rt;
        end
        function scale(self, f)
            w = self.right - self.left;
            self.right = self.left + w*f;
        end
    end
end
```

Constructor, a special method with these jobs:
- compute the handle of the new object,
- execute the function code (to assign values to properties), and
- return the handle of the object.

Constructor is the only method that has the name of the class.

What is the effect of referencing?

```
p = Interval(3,7);  % p references an Interval object
s = p;              % s stores the same reference as p
s.left = 2;         % change value inside object
disp(p.left)        % 2 is displayed
```

The object is not copied—no new object is created! `s` and `p` both reference the same object.

```
p = Interval(3,7);  % p references an Interval object
s = p;              % s stores the same reference as p
s.left = 2;         % change value inside object
disp(p.left)        % 2 is displayed
clear p             % get rid of p from memory
```

The object still can be accessed through `s`.

A handle object is referenced by its handle

```
p = Interval(3,7);
r = Interval(4,6);
```

```
p 167.32  % p references an Interval object
r 177.54 % r references an Interval object
```

What is the effect of referencing?

A handle, also called a reference, is like an address; it indicates the memory location where the object is stored.
In contrast, structs are stored by value …

**Syntax for calling an instance method**

```matlab
r = Interval(4,6);
r.scale(5);
```
Non-objects are passed to a function by value

Syntax for calling an instance method:

```matlab
<reference>.<method><arguments for 2nd thru last parameters>
```

p = Interval(3,7);
```
Better!
```

r = Interval(4,6);
```
yesno = p.isIn(r);
% Explicitly call
% p’s isIn method
```
yesno = isIn(p,r);
% Matlab chooses the % isIn method of one % of the parameters.
```

Objects are passed to a function by reference

```matlab
function scale2(v,f)
% Scale v by a factor f
v = v*f;
end
end
```

```matlab
v = [2 4 1];
scale2(v,5)
disp(v) %???
```

```matlab
function scale2(v,f)
% Scale v by a factor f
v = v*f;
end
end
```

```matlab
v = [2 4 1];
scale2(v,5)
disp(v) %NOCHANGE
```

```matlab
r = Interval(4,6);
scale(r,5)
disp(r.right) % updated value
```

```matlab
r = Interval(4,6);
scale(r,5)
disp(r.right) % NOCHANGE
```

```matlab
classdef Interval < handle
  properties
    left
    right
  end
  methods
    function Inter = Interval(lt,rt)
      Inter.left = lt;
      Inter.right = rt;
    end
    function scale(self,f)
      w = self.right - self.left;
      self.right = self.left + w*f;
    end
    function tf = isIn(self,other)
      tf = self.left >= other.left && ...
          self.right <= other.right;
    end
  end
end
```

```matlab
classdef syntax summary
A class file has the name of the class and begins with keyword classdef:
```
```matlab
classdef Interval < handle
  % An Interval has a left end and a right end
  properties
    left
    right
  end
  methods
    function Inter = Interval(lt,rt)
      Inter.left = lt;
      Inter.right = rt;
    end
    function scale(self,f)
      w = self.right - self.left;
      self.right = self.left + w*f;
    end
    function tf = isIn(self,other)
      tf = self.left >= other.left && ...
          self.right <= other.right;
    end
  end
end
```

```matlab
classdef syntax summary
```

```matlab`
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```