- Previous lecture:
  - Objects are passed by reference to functions
  - Details on class definition (constructor, instance method)
- Today's lecture:
  - More practice with instance methods
  - Overriding methods
  - Array of objects
  - Methods that handle variable numbers of arguments
- Announcements:
  - Prelim 2 tonight 7:30pm
    - CALS Auditorium in Kennedy Hall (Rm 116)
  - Lab exercise problem 2 to be submitted on CMS by Monday 11/16, at 11pm.

classdef syntax summary

A class file has the name of the class and begins with keyword classdef:

classdef *classname* < handle

The class specifies handle objects

Constructor returns a reference to the class object

Each instance method's first parameter must be a reference to the instance (object) itself

Use keyword end for classdef, properties, methods, function.

Properties

Constructor

Instance methods (functions)

```
classdef Interval < handle
% An Interval has a left end and a right end

  properties
    left
    right
  end

  methods
    function Inter = Interval(lt, rt)
    % Constructor: construct an Interval ob
      Inter.left= lt;
      Inter.right= rt;
    end

    function scale(self, f)
    % Scale the interval by a factor f
      w= self.right - self.left;
      self.right= self.left + w*f;
    end
  end
end
```

This file's name is Interval.m

Syntax for calling an instance method:

<reference>.<method>(<arguments for 2nd thru last parameters>)

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

Better!

```
yesno= p.isIn(r);
% Explicitly call
% p's isIn method

yesno= isIn(p,r);
% Matlab chooses the
% isIn method of one
% of the parameters.
```

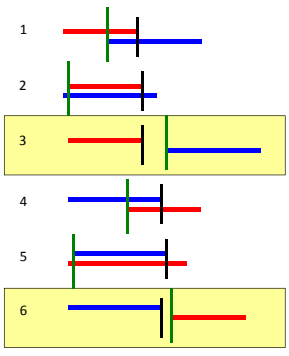
```
classdef Interval < handle
 :
 methods
 :
  function scale(self, f)
  % Scale self by a factor f
    w= self.right - self.left;
    self.right= self.left + w*f;
  end

  function tf = isIn(self, other)
  % tf is true if self is in other interval
    tf= self.left>=other.left && ...
        self.right<=other.right;
  end


 end
end
```

Method to find overlap between two Intervals

```
function Inter = overlap(self, other)
% Inter is overlapped Interval between self
% and the other Interval. If no overlap then
% self is empty Interval.
```



The overlap's left (OLeft) is the rightmost of the two original lefts

The overlap's right (ORight) is the leftmost of the two original rights

No overlap if OLeft > ORight


```
function Inter = overlap(self, other)
% Inter is overlapped Interval between self
% and the other Interval. If no overlap then
% self is empty Interval.

  Inter= Interval.empty();
  left= max(self.left, other.left);
  right= min(self.right, other.right);
  if right-left > 0
      Inter= Interval(left, right);
  end
end
```

Built-in function to create an empty array of the specified class

Built-in function isempty

```
% Example use of overlap function
A= Interval(3,7);
B= Interval(4,4+rand*5);
X= A.overlap(B);
if ~isempty(X)
    fprintf('(%f,%f)\n', X.left,X.right)
end
```

Overriding built-in functions

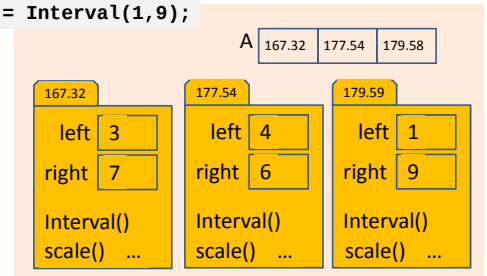
- You can change the behavior of a built-in function for an object of a class by implementing a function of the same name in the class definition
- Called "overriding" (called "overloading" in Matlab documentation)
- A typical built-in function to override is **`disp`**
  - Specify which properties to display, and how, when the argument to **`disp`** is (a reference to) an object
  - Matlab calls **`disp`** when there's no semi-colon at the end of an assignment statement

See `Interval.m`

An "array of objects" is really an …

array of references to objects

```
>> A= Interval(3,7);
>> A(2)= Interval(4,6);
>> A(3)= Interval(1,9);
```



MATLAB allows an array to be appended

```
v= [3 1 5 9]
v(7)= 4
```

- What happens to **`v(5)`** and **`v(6)`**?

3	1	5	9	0	0	4

- MATLAB assigns some "default value" to the skipped over components for simple, cell, and struct arrays
- For arrays of objects, you must implement the constructor to handle such a situation

Constructor needs to be able to handle a call with no arguments

```
>> A= Interval(3,7);     % Array of length 1
>> A(2)= Interval(4,6); % Array of length 2
>> A(3)= Interval(1,9); % Array of length 3
>> A(5)= Interval(2,5); % Array of length 5
```

Error!

- Interval constructor we have so far requires two parameters:
  `function Inter = Interval(lt, rt)`
- User specified two arguments as required for A(5), but…
- Matlab has to assign A(4) "on its own" by calling the constructor, but no arguments get passed → Error!

Constructor that handles variable number of args

- When used inside a function, **`nargin`** returns the number of arguments that were passed
- If **`nargin`**≠2, constructor ends without executing the assignment statements. Then **`Inter.left`** and **`Inter.right`** get any default values defined under properties. In this case the default property values are **`[]`** (type **`double`**)

```
classdef Interval < handle

  properties
    left
    right
  end

  methods
    function Inter = Interval(lt, rt)
      if nargin==2
        Inter.left= lt;
        Inter.right= rt;
      end
    end

    …

  end

end
```

If a class defines an object that may be used in an array…

- Constructor must be able handle a call that does not specify any arguments
  - Use built-in command **`nargin`**, which returns the number of function input arguments passed
- The overridden **`disp`** method, if implemented, should check for an input argument that is an array and handle that case explicitly. Details will be discussed next lecture.

A function to create an array of Intervals

```
function inters = intervalArray(n)
% Generate n random Intervals.  The left and
% right ends of each interval is in (0,1)

for k = 1:n
  randVals= rand(1,2);
  if randVals(1) > randVals(2)
     tmp= randVals(1);
     randVals(1)= randVals(2);
     randVals(2)= tmp;
  end
  inters(k)= Interval(randVals(1),randVals(2));
end
```

An independent function, not an instance method. See `intervalArray.m`

A function to find the widest Interval in an array

```
function inter = widestInterval(A)
% inter is the widest Interval (by width) in
% A, an array of Intervals
```

An independent function, not an instance method. See `widestInterval.m`

A weather object can make use of Intervals …

- Define a class LocalWeather to store the weather data of a city, including monthly high and low temperatures and precipitation
  - Temperature: low and high → an Interval
    - For a year → length 12 array of Intervals
  - Precipitation: a scalar value
    - For a year → length 12 numeric vector
  - Include the city name: a string

```
classdef LocalWeather < handle

 properties
   city     % string
   temps    % array of Intervals
   precip   % numeric vector
 end

 methods
   ...
 end

end
```

Weather data file

```
//Ithaca
//Monthly temperature and precipitation
//Lows (cols 4-8), Highs (col 12-16), precip (cols 20-24)
//Units:  English
      15       31    2.08
      17       34    2.06
      23       42    2.64
      34       56    3.29
      44       67    3.19
      53       76    3.99
      58       80    3.83
      56       79    3.63
      49       71    3.69
     NaN       59     NaN
      32       48    3.16
      22       36    2.40
```

Class `LocalWeather` should be able to construct an object from such data files, given the known file format.

See `ithacaWeather.txt`, `LocalWeather.m`

```
classdef LocalWeather < handle

 properties
   city= '';
   temps= Interval.empty();
   precip
 end

 methods
   function lw = LocalWeather(fname)
     ...
   end

   ...
 end

end
```

Set property variable that will store an array of objects to the correct type, either under properties or in the constructor

```
classdef LocalWeather < handle
 properties
   city=''; temps=Interval.empty(); precip=0;
 end
 methods
   function lw = LocalWeather(fname)
     fid= fopen(fname,'r');
     s= fgetl(fid);
     lw.city= s(3:length(s));
     for k= 1:3
        s= fgetl(fid);
     end
     for k=1:12
        s= fgetl(fid);
        lw.temps(k)= Interval(str2double(s(4:8), ...
                              str2double(s(12:16)));
        lw.precip(k)= str2double(s(20:24));
     end
     fclose(fid);
   end
   ...
 end  %methods
end  %classdef
```

```
//Ithaca
//Monthly temperature and p
//Lows (cols 4-8), Highs (c
//Units:  English
      15       31    2.08
      17       34    2.06
      23       42    2.64
      34       56    3.29
      44       67    3.19
      53       76    3.99
      58       80    3.83
      56       79    3.63
      49       71    3.69
     NaN       59     NaN
      32       48    3.16
      22       36    2.40
```

```
classdef LocalWeather < handle
  properties
    city=''; temps=Interval.empty();
    precip=0;
  end
  methods
    function lw = LocalWeather(fname)
      ...
    end

    function showCityName(self)

    end
    ...
  end  %methods
end  %classdef
```

Function to show data of a month of **`LocalWeather`**

```
function showMonthData(self, m)
% Show data for month m, 1<=m<=12.




end
```

Should display which month, the high and low temperatures, and precipitation

Observations about our class Interval

- We can use it (create Interval objects) anywhere
  - Within the Interval class, e.g., in method overlap
  - "on the fly" in the Command Window
  - In other function/script files – not class definition files
  - In another class definition
- Designing a class well means that it can be used in many different applications and situations