• Previous lecture:
  – Objects are passed by reference to functions
  – Details on class definition
    • Constructor
    • Methods
  – Overloading methods
• Today’s lecture:
  – Array of objects
  – Methods that handle variable numbers of arguments
  – Attributes for properties and methods
• Announcements:
  – Discussion this week in Upson B7 lab
  – Prelim 2 to be returned at end of lecture. Unclaimed papers (and those on which student didn’t indicate the lecture time) can be picked up after 5pm today during consulting hours (Su-R 5-10p) at ACCEL Green Rm (Carpenter Hall)
An “array of objects” is really an …

array of references to objects

$$\begin{array}{c}
>> A = \text{Interval}(3,7); \\
>> A(2) = \text{Interval}(4,6); \\
>> A(3) = \text{Interval}(1,9);
\end{array}$$
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

• **An overloaded `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.
Constructor that handles variable number of args

- When used inside a function, \texttt{nargin} returns the number of arguments that were passed
Constructor that handles variable number of args

- When used inside a function, `nargin` returns the number of arguments that were passed.
- If `nargin` is not 2, the 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`).

```matlab
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
```
A function to create an array of **Intervals**

```matlab
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
```

See `intervalArray.m`
A function to find the biggest Interval in an array

function inter = biggestInterval(A)
% inter is the biggest Interval (by width) in A, an array of Intervals
A function to find the biggest *Interval* in an array

```matlab
function inter = biggestInterval(A)
    % inter is the biggest Interval (by width) in A, an array of Intervals

    inter= A(1);  % biggest Interval so far
    for k= 2:length(A)
        if A(k).getWidth() > inter.getWidth()
            inter= A(k);
        end
    end
end
```
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 $\rightarrow$ an `Interval`
    • For a year $\rightarrow$ length 12 array of Intervals
  – Precipitation: a scalar value
    • For a year $\rightarrow$ length 12 numeric vector
  – Include the city name: a string

```matlab
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

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>31</td>
<td>2.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>34</td>
<td>2.06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>42</td>
<td>2.64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>56</td>
<td>3.29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>67</td>
<td>3.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>76</td>
<td>3.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>80</td>
<td>3.83</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>79</td>
<td>3.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>71</td>
<td>3.69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NaN</td>
<td>59</td>
<td>NaN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>48</td>
<td>3.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>36</td>
<td>2.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

    function showCityName(self)
        disp(self.city)
    end
end %methods
end %classdef
Function to show data of a month of `LocalWeather`:

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

Should display which month, the high and low temperatures, and precipitation.
Function to show data of a month of **LocalWeather**

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

    mo= {'Jan','Feb','Mar','Apr','May','June',...  
         'July','Aug','Sep','Oct','Nov','Dec'};
    fprintf('%s Data
', mo{m})
    fprintf('Temperature range: ')disp(self.temps(m))
    fprintf('Average precipitation: %.2f
', ...
            self.precip(m))
end
```

See *LocalWeather.m*
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