• **Previous lecture:**
  - Objects are passed by reference to functions
  - Details on class definition (constructor, instance method)

• **Today’s lecture:**
  - Overloading methods
  - Array of objects
  - Class reuse

• **Announcements:**
  - Welcome back from wellness break!
  - Prelim 2 will be graded this weekend
    - Exam posted on website – practice re-doing problems, trying out solutions in Matlab
  - Tomorrow’s lab builds on today’s examples
  - Project 5
function Inter = overlap(self, other)
% Inter is overlapped Interval between self
% and the other Interval. If no overlap then
% Inter is empty array of class 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

% 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
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 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

Error!

• Interval constructor we have so far requires two parameters:
  
  \[
  \text{function Inter} = \text{Interval}(lt, rt)
  \]

• 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 \( \Rightarrow \text{Error!} \)
Function overloading I

Problem: “default construction” passes 0 args, but our constructor has 2 input params

Want a function that performs the same task for different numbers of inputs

- **MATLAB’s solution**: accept all possible arguments, then ask how many we got

Examples

- `rand()`, `rand(2)`, `rand(1, 3)`
- `max(4, 3)`, `max([6 7 5])`
- `plot(x, y)`, `plot(x, y, 'm-*')`, `plot(x, y, v, w)`
- `Interval(4, 6)`, `Interval()`
Constructor that handles variable number of args

- When used inside a function, `nargin` evaluates to the number of arguments that were passed

```matlab
classdef Interval < handle

properties
    left
    right
end

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

    ...
end
end
```
Constructor that handles variable number of args

- When used inside a function, `nargin` evaluates to the number of arguments that were passed.
- If `nargin` $\neq 2$, 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
```
Default values

• Default property value: empty double array \([\ ]\)
• Within an array:
  – Default double: 0
  – Default char: null (char(0), but looks like a space in MATLAB)
  – Default cell: empty cell \({}\}
  – Default object: call constructor with no arguments
    • Advantage of bundling behavior with data

Later: customizing default property values in objects
Function overloading II (arguably “overriding”)

Want to customize an existing function for new classes

- MATLAB’s solution: define a method in the class with the same function name

Examples

- disp
- plot
- Operators!

DEMO
Overload disp in Interval.m
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.
  - Caution: accessing properties of an entire array produces “comma-separated lists” – an advanced topic
Write a function to create an array of random intervals
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)
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
```

An independent function, not an instance method. See `intervalArray.m`
Write a function to return the widest interval in an array
A function to find the widest \texttt{Interval} in an array

\begin{verbatim}
function inter = widestInterval(A)
  \% inter is the widest \texttt{Interval} (by width) in
  \% A, an array of \texttt{Intervals}
\end{verbatim}

An independent function, not an instance method. See \texttt{widestInterval.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

inter= A(1); % widest Interval so far
for k= 2:length(A)
    if A(k).right - A(k).left > ...
        inter.right - inter.left
        inter= A(k);
    end
end

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

inter= A(1);  % widest Interval so far
for k= 2:length(A)
    if A(k).getWidth() > inter.getWidth()
        inter= A(k);
    end
end

An independent function, not an instance method. See widestInterval.m
Poll: Functions returning objects

\[ v = \text{[Interval}(2, 4) \text{ Interval}(3, 7)]; \]
\[ w = \text{widestInterval}(v); \]
\[ w.\text{scale}(2); \]
\[ \text{disp}(v(2).\text{right}) \]

What is displayed?

A: 7
Intervals were copied when passed into function, so original does not change

B: 7
New Interval was created when returned from function, so original does not change

C: 11
Original is modified through returned value
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
### Syracuse weather data

#### Monthly temperature and precipitation

<table>
<thead>
<tr>
<th>Low</th>
<th>High</th>
<th>Precip</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>31</td>
<td>3.07</td>
</tr>
<tr>
<td>16</td>
<td>33</td>
<td>2.96</td>
</tr>
<tr>
<td>23</td>
<td>42</td>
<td>3.09</td>
</tr>
<tr>
<td>34</td>
<td>55</td>
<td>3.91</td>
</tr>
<tr>
<td>43</td>
<td>67</td>
<td>3.86</td>
</tr>
<tr>
<td>52</td>
<td>76</td>
<td>4.27</td>
</tr>
<tr>
<td>58</td>
<td>80</td>
<td>4.03</td>
</tr>
<tr>
<td>56</td>
<td>79</td>
<td>3.95</td>
</tr>
<tr>
<td>48</td>
<td>70</td>
<td>3.79</td>
</tr>
<tr>
<td>42</td>
<td>58</td>
<td>3.44</td>
</tr>
<tr>
<td>31</td>
<td>47</td>
<td>3.19</td>
</tr>
<tr>
<td>21</td>
<td>36</td>
<td>2.82</td>
</tr>
</tbody>
</table>

Units: English

Class LocalWeather should be able to construct an object from such data files, given the known file format.
//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>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>31</td>
<td>2.08</td>
</tr>
<tr>
<td>17</td>
<td>34</td>
<td>2.06</td>
</tr>
<tr>
<td>23</td>
<td>42</td>
<td>2.64</td>
</tr>
<tr>
<td>34</td>
<td>56</td>
<td>3.29</td>
</tr>
<tr>
<td>44</td>
<td>67</td>
<td>3.19</td>
</tr>
<tr>
<td>53</td>
<td>76</td>
<td>3.99</td>
</tr>
<tr>
<td>58</td>
<td>80</td>
<td>3.83</td>
</tr>
<tr>
<td>56</td>
<td>79</td>
<td>3.63</td>
</tr>
<tr>
<td>49</td>
<td>71</td>
<td>3.69</td>
</tr>
<tr>
<td>NaN</td>
<td>59</td>
<td>NaN</td>
</tr>
<tr>
<td>32</td>
<td>48</td>
<td>3.16</td>
</tr>
<tr>
<td>22</td>
<td>36</td>
<td>2.40</td>
</tr>
</tbody>
</table>

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

See LocalWeather.m for complete code including use of nargin

//Ithaca
//Monthly temperature and precipitation
//Lows (cols 4-8), Highs (col 12-16), precip (col 20-24), units: English
//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
49  71  3.69
NaN 59  NaN
32  48  3.16
22  36  2.40
Method to show data of a month of `LocalWeather`

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

end
```

Should display which month, the high and low temperatures, and precipitation.
Method 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\n', mo{m})
fprintf('Temperature range: ')
disp(self.temps(m))
fprintf('Average precipitation: %.2f\n', ...
    self.precip(m))
end
```

See `LocalWeather.m`