• Previous lecture:
  – Objects are passed by reference to functions
  – Details on class definition
    • Constructor
    • Methods
• Today’s lecture:
  – Overloading methods
  – Array of objects
  – Methods that handle variable numbers of arguments
• Announcements:
  – Discussion this week in Upson B7 lab
  – Prelim 2 tonight at 7:30pm
    • Last names A-L in Uris Hall Room G01
    • Last names M-Z in Baker Lab Room 200

Overloading 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 “overloading”
• A typical built-in function to overload is disp
  – Specify which properties to display, and how, when
    the argument to disp is an object
  – Matlab calls disp when there’s no semi-colon at the
    end of an assignment statement

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, `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`).

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
end
```

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
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 → 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

```matlab
classdef LocalWeather < handle
    properties
        city      % string
        temps  % array of Intervals
        precip % numeric vector
    end
    methods
        function lw = LocalWeather(fname)
            ... end
end
```

Weather data file

```plaintext
//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.

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

```matlab
classdef LocalWeather < handle
    properties
        city ;
        temps= Interval.empty();
        precip end
    methods
        function lw = LocalWeather(name)
            ... end
end
```
//Ithaca
//Monthly temperature and
//Lows (cols 4-8), Highs (cols 12-14)  U
//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

function lw = LocalWeather(fname)
    fid = fopen(fname, 'r');
    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

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

OOP ideas

- Aggregate variables/methods into an abstraction (a class) that makes their relationship to one another explicit
- Objects (instances of a class) are self-governing (protect and manage themselves)
- Hide details from client, and restrict client’s use of the services
- Provide clients with the services they need so that they can create/manipulate as many objects as they need