CS 1112 Introduction to Computing Using MATLAB

Instructor: Dominic Diaz

Website:
https://www.cs.cornell.edu/courses/cs1112/2022fa/

Today: object-oriented programming

Agenda and announcements

- Last time
  - Object-oriented programming
- Today
  - Object-oriented programming
    - More practice with objects and classes
- Announcements
  - Ex 12 due Tuesday Nov 15th (should be relatively easy)
  - Project 5 due Monday 11/14
  - Prelim 2 TONIGHT!

Constructor should be able to handle call with no inputs

```
A = Interval(3,7);
A(2) = Interval(4,6);
A(3) = Interval(1,9);
A(5) = Interval(2,5);
```

Error!

Why is there an error? The Interval constructor requires two input parameters

The user specified 2 inputs for `A(5)`, but…

MATLAB will implicitly call
`A(4) = Interval()` → Error!

```
    properties
        left
        right
    end

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

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

Constructor that handles variable number of input args

- When used inside a function, `nargin` returns the number of input arguments that were passed
- If `nargin ~= 2`, constructor ends without executing the assignment statements. Then `Inter.left` and `Inter.right` get any default values (`[ ]` (type `double`)).

This is called function overloading

```
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
        % object
            if nargin == 2
                Inter.left = lt;
                Inter.right = rt;
            end
        end

        ...
    end
end
```

What is nargin doing?

```
function Inter = Interval(lt, rt)
     % constructor: construct an Interval
     % object
          if nargin == 2
               Inter.left = lt;
               Inter.right = rt;
          end
end
```

Without the if statement, during the implicit call `Intervals(2) = Interval()`, the code tries to access `lt` and `rt`, which don’t exist.

```
Intervals(1) = Interval(3,5);     % creates array of intervals, with the first one
                                  % being in interval from 3 to 5
Intervals(3) = Interval(8, 10);   % implicitly calls Intervals(2) = Interval()
                                  %       since nargin == 0, sets Intervals(2).left and
                                  %       Intervals(2).right to default values []
                                  % Creates interval from 8 to 10 at third position in
                                  % in Intervals array
```

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);









end
```

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
```

How do we call `intervalArray`?

```
inters = n.intervalArray();

inters = intervalArray(n);
```

An independent function, not an instance method.

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
```

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
```

If this is an independent function, how do we call this function?

```
inter = widestInterval(intervals);
```

If this is a class method (written in the `classdef`), how do we call this function?

```
inter = intervals.widestInterval();
```

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
    - City name: 1D `char` array

```
classdef LocalWeather < handle
    properties
        city
        temps
        precip
    end

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

        ...
    end
end
```

Default property vals

In general, you should set a default value for each of the properties that are not the default MATLAB type (double or array of doubles).

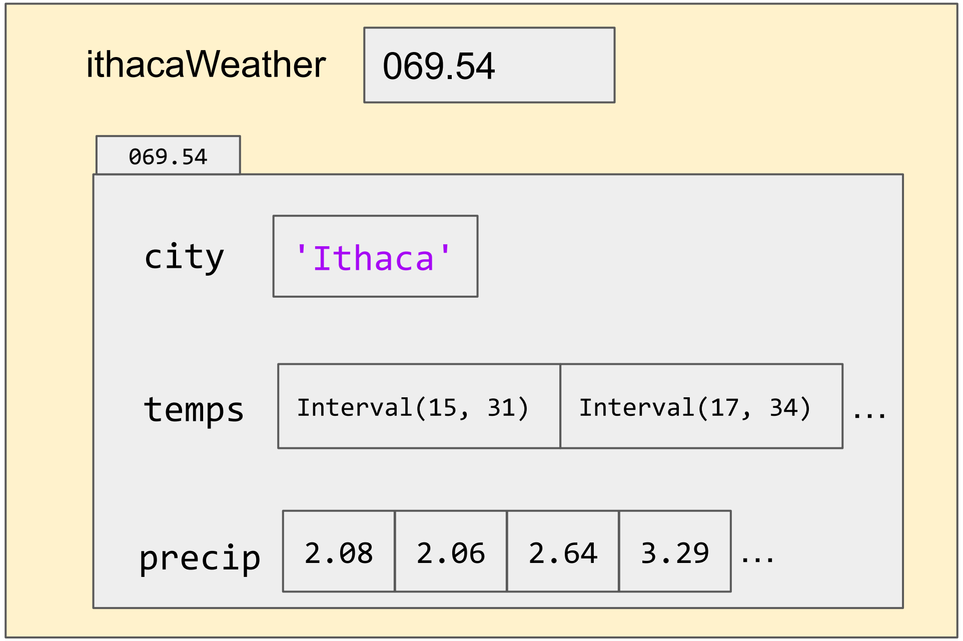
- city will be an array of `chars` → set default value to empty `char` array
- temps will be an array of `Intervals` → set default value to empty array of `Intervals`
- precip will be an array of doubles → default value will be set correctly

```
classdef LocalWeather < handle
    properties
        city = '';
        temps = Interval.empty();
        precip
    end

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

        ...
    end
end
```

Weather data file

```
// file ithWeather.txt
// Ithaca
// Monthly temperature and precipitation
// Lows (cols 4-8), Highs (cols 12-16), precip (20-24)

   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
```

```
ithacaWeather = LocalWeather('ithWeather.txt');
```



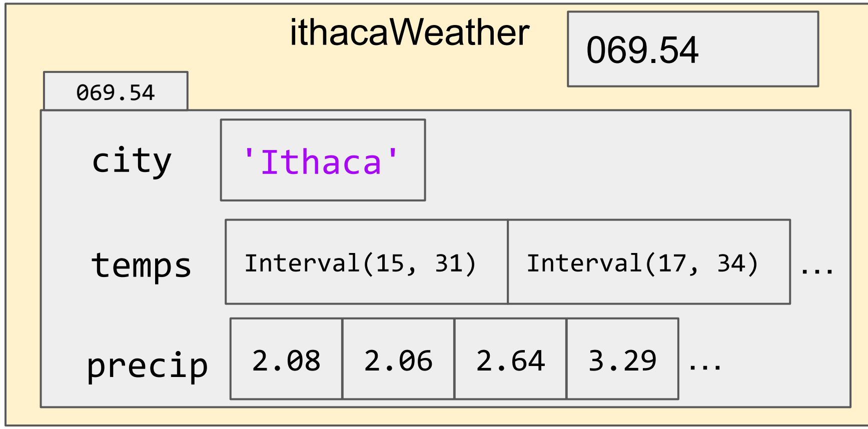
Fill in the blank

```
classdef LocalWeather < handle
    properties
        city = '';
        temps = Interval.empty();
        precip
    end

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

        function showCityName(self)

            ______________
        end
    end
end
```

Fill in the blank

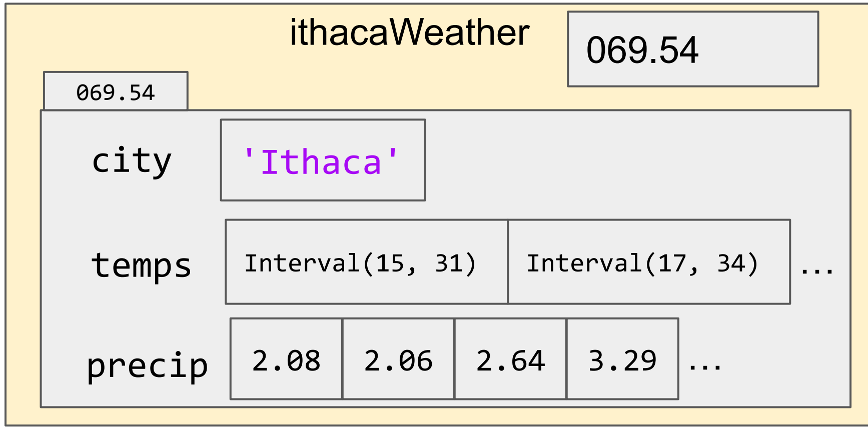
```
classdef LocalWeather < handle
     properties
          city = '';
          temps = Interval.empty();
          precip
     end

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

          function showCityName(self)

               disp(self.city)
          end
     end
end
```

Function to show data of a month of LocalWeather



```
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', ________)  % display [month] data
fprintf('Temperature range: ')
disp(_____________)              % display the temp interval
fprintf('Average precipitation: %.2f\n', _______________)
end
```

Function to show data of a month of LocalWeather



```
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
```

Definitions you should know

Try to fill in the blanks before you go to the next slide

- ______________: The template that specified a custom MATLAB type.
  - Defines _____________ and ____________ for that class.
- _____________: Specific instance of a class.
- _____________: A type of programming that focuses on creating objects and writing methods that act on those objects
- _________________: special method that returns the handle to a newly allocated object
- _________________: unique identifier of an object generated by MATLAB
- _________________________: change the behavior of a built-in function for an object of a class
- _________________________: writing functions that take variable number of input arguments
  - _________________: returns the number of function input arguments given in the call to the currently executing function

Definitions you should know

- class: The template that specified a custom MATLAB type.
  - Defines properties and methods for that class.
- object: Specific instance of a class.
- OOP: A type of programming that focuses on creating objects and writing methods that act on those objects
- constructor: special method that returns the handle to a newly allocated object
- handle: unique identifier of an object generated by MATLAB
- Overriding functions: change the behavior of a built-in function for an object of a class
- Overloading functions: writing functions that take variable number of input arguments
  - nargin: returns the number of function input arguments given in the call to the currently executing function