• Previous lecture:
  – Introduction to objects and classes

• Today’s lecture:
  – Defining a class
    • Constructor
    • Methods
  – Objects are passed by reference to functions
  – Overloading methods

• Announcements:
  – Prelim 2 will be on Tues
Object-Oriented Programming

• First design and define the classes (of the objects)
  – Identify the properties (data) and actions (methods, i.e., functions) of each class

• Then create the objects (from the classes) that are then used, that interact with one another
Class Interval

- An interval has two properties:
  - left, right
- Actions—methods—of an interval include
  - Scale, i.e., expand
  - Shift
  - Add one interval to another
  - Check if one interval is in another
  - Check if one interval overlaps with another

See demoInterval1.m
Class Interval

• An interval has two properties:
  – left, right

• Actions—methods—of an interval include:
  – Scale, i.e., expand
  – Shift
  – Add one interval to another
  – Check if one interval is in another
  – Check if one interval overlaps with another

To specify the properties and actions of an object is to define its class.
An **Interval** object

- **left** 3
- **right** 7

The “handle” or “reference” of the object

---

**classdef** Interval < handle

**properties**
- left
- right

**methods**
- function scale(self, f)
  - ...
  - end
- function shift(self, s)
  - ...
  - end
- function Inter = overlap(self, other)
  - ...
  - end
- function Inter = add(self, other)
  - ...
  - end

---

An object is also called an “instance” of a class. It contains every property, “instance variable,” and every “instance method” defined in the class.
Multiple **Interval** objects

Every object (instance) contains every “instance variable” and every “instance method” defined in the class. Every object has its own handle.
Simplified Interval class

To create an Interval object, use its class name as a function call: \( p = \text{Interval}(3, 7) \)
The constructor method

To create an Interval object, use its class name as a function call: \( p = \text{Interval}(3,7) \)

The constructor is a special method whose main jobs are to

- compute the handle of the new object,
- execute the function code (to assign values to properties), and
- return the handle of the object.

Constructor has the name of the class.
Executing an instance method

```matlab
r = Interval(4,6);
r.scale(5)
disp(r.right) %What will it be?
```

```matlab
r
```

```
177.54
```

```
177.54
```

```
left
```

```
4
```

```
right
```

```
6
```

```
Interval()
```

```
scale()
```

```
Function space of scale
```

```
self
```

```
177.54
```

```
f
```

```
5
```

```
w
```

```
2
```

```matlab
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
        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
```
Object is passed to a function by reference

```matlab
r = Interval(4,6);
r.scale(5)
disp(r.right) % updated value
```

```
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
        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
```

Objects are passed to functions by reference. Changes to an object’s property values made through the local reference (self) stays in the object even after the local reference is deleted when the function ends.
Non-objects are passed to a function by value

```matlab
v = [2 4 1];
scale2(v,5)
disp(v) %???
```

```matlab
function scale2(v,f)
  % Scale v by a factor f
  v = v*f;
```
Non-objects are passed to a function by value

\begin{align*}
v &= \begin{bmatrix} 2 & 4 & 1 \end{bmatrix}; \\
\text{scale2}(v, 5) \\
\text{disp}(v) &\quad ???
\end{align*}

\begin{align*}
\text{function scale2}(v, f) \\
\% \text{ Scale } v \text{ by a factor } f \\
v &= v \times f;
\end{align*}
Non-objects are passed to a function by value.

```
v = [2 4 1];
scale2(v,5)
disp(v)  % NO CHANGE
```

```
function scale2(v,f)
    % Scale v by a factor f
v = v*f;
```
Objects are passed to a function by reference

```matlab
r = Interval(4,6);
r.scale(5)
disp(r.right) % updated value
```

```matlab
classdef Interval < handle
    :
    methods
    :
        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
```

Non-objects are passed to a function by value

```matlab
v= [2 4 1];
scale2(v,5)
disp(v) %NO CHANGE
```

```matlab
function scale2(v,f)
    % Scale v by a factor f
    v= v*f;
```
Syntax for calling an instance method:

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

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

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

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

1

2

3

4

5

6

redRight < blueRight

blueRight < redRight
The overlap’s left (OLeft) is the rightmost of the two original lefts.
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
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

% 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)’
    end

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

Built-in function isempty
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

```
Constructor: returns a reference to the class object
```

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

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

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

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

This file’s name is `Interval.m`