• Previous lecture:
  – Introduction to objects and classes
• Today’s lecture:
  – Defining a class
    • Properties
    • Constructor and other methods
  – Objects are passed by reference to functions
• Announcements:
  – Project 5 due Friday at 11pm
  – Optional review sessions on Saturday 2-3:30pm and Monday 5:30-7pm. Location TBA.
  – Prelim 2 on Tues at 7:30pm

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

A handle object is referenced by its handle
\[ p = \text{Interval}(3,7); \quad r = \text{Interval}(4,6); \]

What is the effect of referencing?

What is the effect of referencing?

A handle, also called a reference, is like an address; it indicates the memory location where the object is stored.
In contrast, structs are stored by value …

\[
P.x = 5; \quad P.y = 0; \quad \% \text{ A point struct } P
\]
\[
Q = P; \quad \% \text{ Q gets a copy of } P \quad \% \text{ Q is ANOTHER}
\]
\[
Q.y = 9; \quad \% \text{ Changes Q’s copy only, not } P’s
\]
\[
disp(P.y) \quad \% \text{ What is displayed?}
\]

In fact, storing-by-value is true of all non-handle-object variables. You already know this from before …

\[
a = 5;
\]
\[
b = a + 1; \quad \% \text{ b stores the value 6, not }
\]
\[
a = 8; \quad \% \text{ Changing a does not change b}
\]
\[
disp(b) \quad \% \text{ 6 is displayed}
\]

Syntax for calling an instance method

```
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
```

Calling an object’s method (instance method)

```
r = Interval(4,6);
```
```
r.scale(5)
```

Executing an instance method

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

Function space of `scale`

```
self.right = self.left + w*f;
```

Calling an object’s method (instance method)

```
p = Interval(3,7);
r = Interval(4,6);
r.scale(5)
```

Executing an instance method

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

Function space of `scale`

```
self.right = self.left + w*f;
```
Object is passed to a function by reference

```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 with given left and right end
            Left = Lt;
            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
```

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

Objects are passed to a function 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.

```
function scale2(v,f)
    % Scale v by a factor f
    v = v * f;
end
```

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

Non-objects are passed to a function by value

```
function Inter = overlap(self, other)
    % Inter is overlapped Interval between self and the other Interval.
    % If no overlap then self is empty Interval.
end
```

```
function isIn(self, other)
    % tf is true if self is in other interval
    tf = self.left <= other.left & ... self.right <= other.right;
end
```

```
yesno = p.isIn(r);
% Explicitly call p's isIn method
yesno = isIn(p,r);
% Matlab chooses the % isIn method of one % of the parameters.
```

Syntax for calling an instance method:

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

```
p = Interval(3,7);
r = Interval(4,6);
yesno = p.isIn(r);
% Better!
```

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.
end
```

```
compare two intervals
```

```
1
2 redRight < bluelight
3
4
5
6 blueRight < redRight
```

April 5, 2007  Lecture 22
```
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)
', X.left,X.right)
end

Overriding 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 “overriding” (called “overloading” in Matlab documentation)
- A typical built-in function to override is `disp`
  - Specify which properties to display, and how, when the argument to `disp` is (a reference to) an object
  - Matlab calls `disp` when there’s no semi-colon at the end of an assignment statement