Introduction to Object-Oriented Programming in MATLAB

Loren Shure

Agenda

- Object-oriented programming
- Basic object-oriented programming syntax in MATLAB
- Classes in MATLAB

What is a program?

Data

```
x = 12
while (x < 100)
    x = x+1
    if (x == 23)
        disp('Hello')
    end
end
```

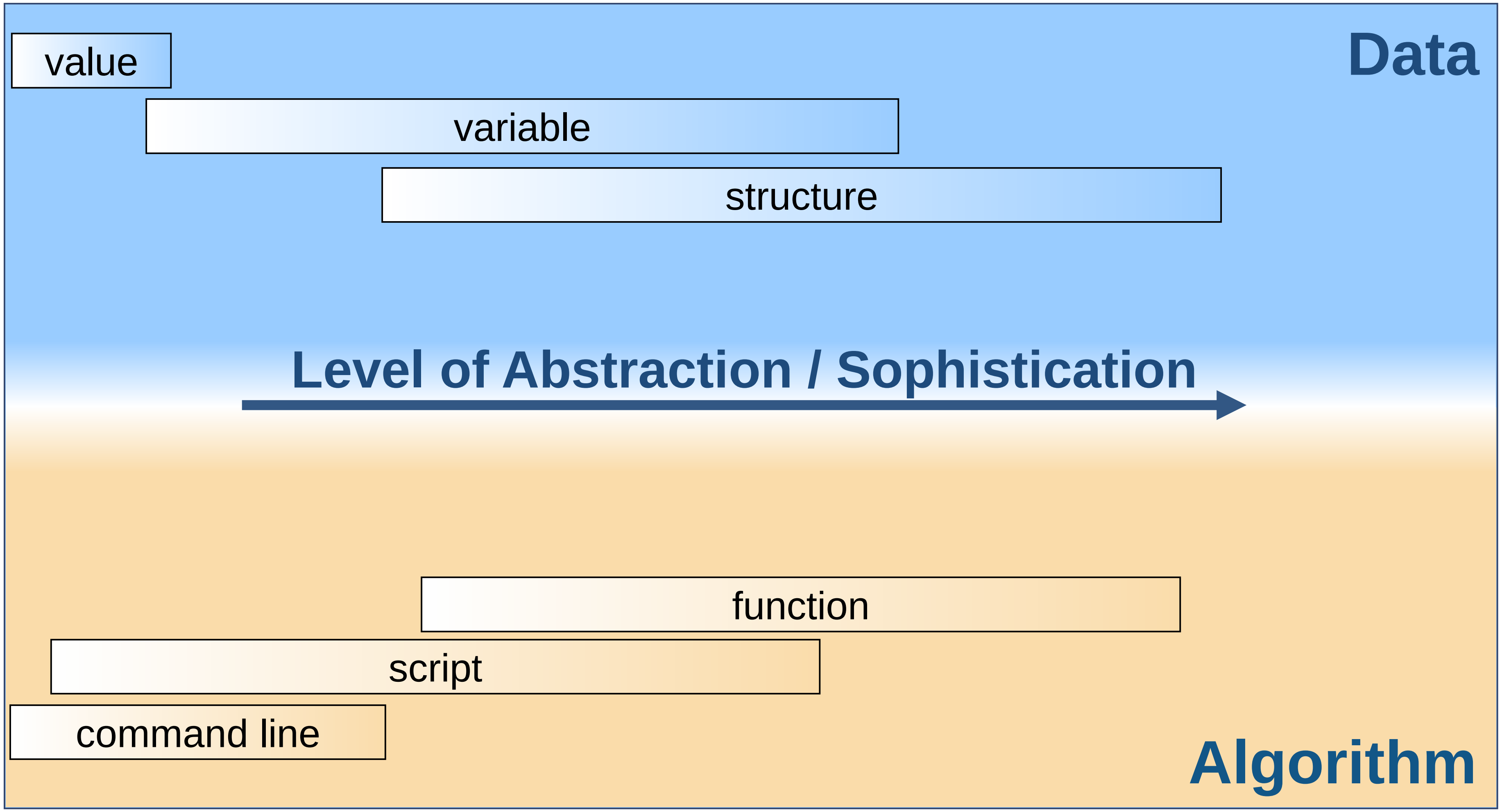
```
x = 12
while (x < 100)
    x = x+1
    if (x == 23)
        disp('Hello')
    end
end
```

Code

```
Assignment
Looping Test
    Increment
    Test to Act
        Take Action
    End
End
```

Actions

Progression of Programming Techniques

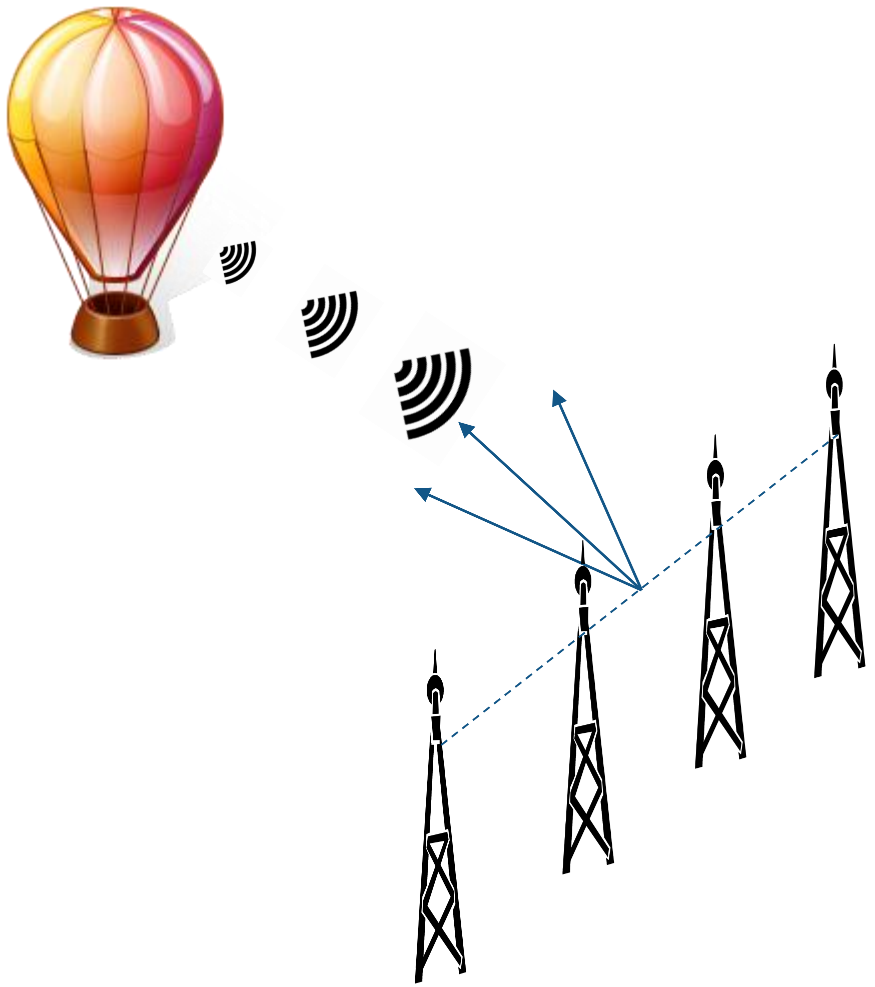


Example: Sensor Array

- Transmitting a signal from a weather balloon

- Locating the signal with a sensor array

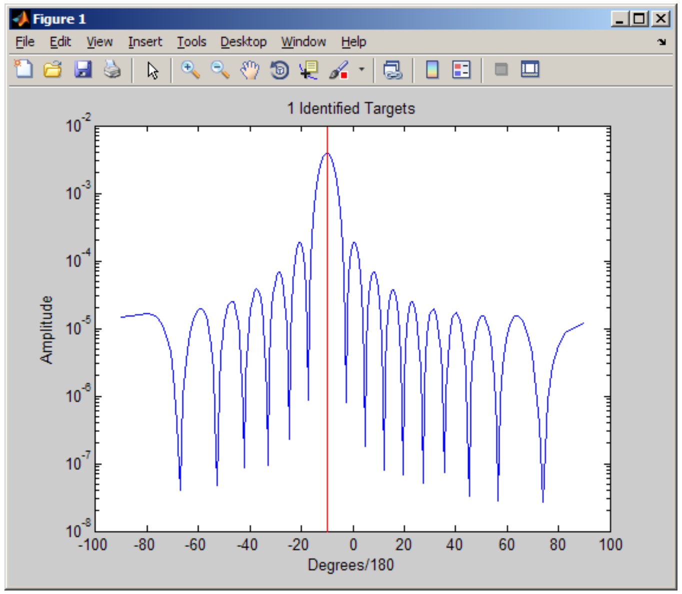
- Computing the angle of arrival (AoA) for the signal



Procedural Programming

- Easy to learn
- Minimal planning

- There is no formal relationship between data and functions.
- Every detail is exposed.



Data and Actions to Implement

Wavelength

Number

Spacing

Location

Reading

Frequency

Data

Determine peaks

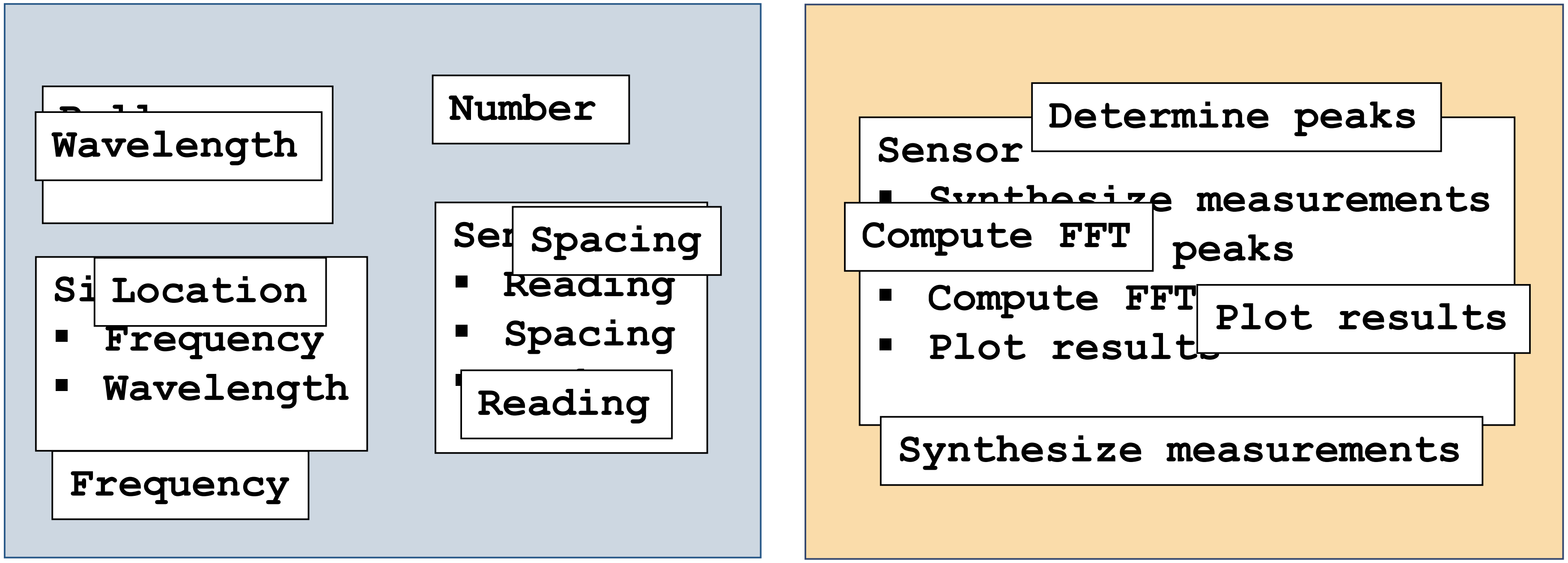
Compute FFT

Plot results

Synthesize measurements

Actions

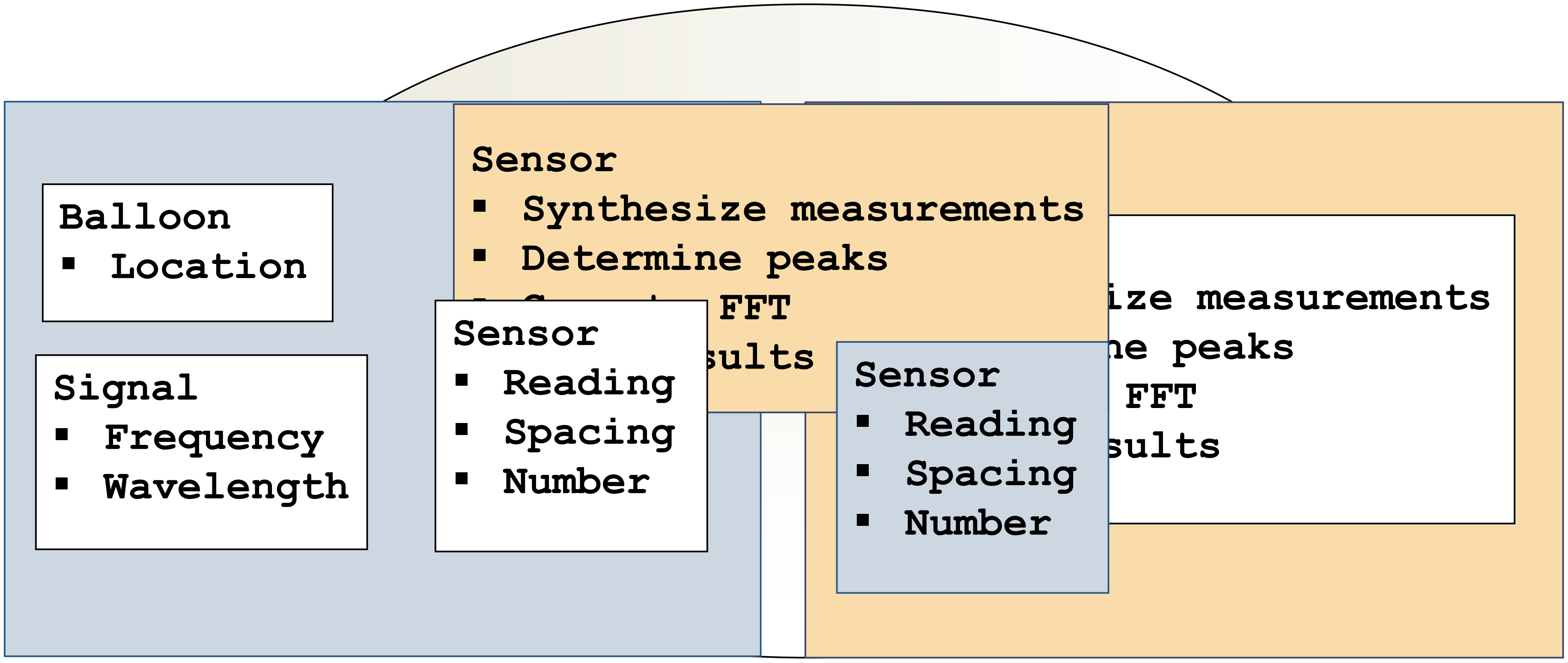
Related Data and Actions



Data

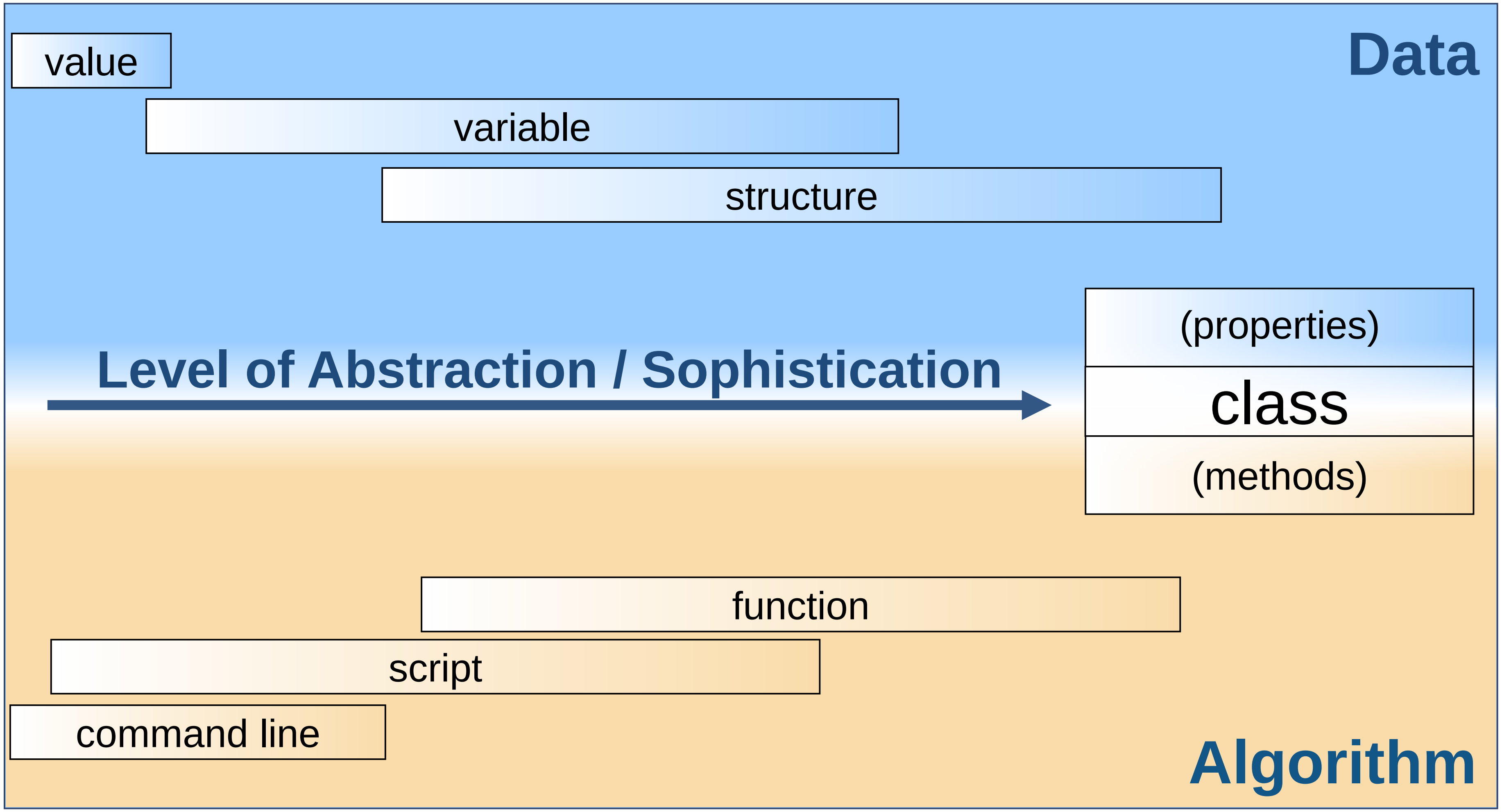
Actions

Grouping Related Data and Actions



Data Class Actions

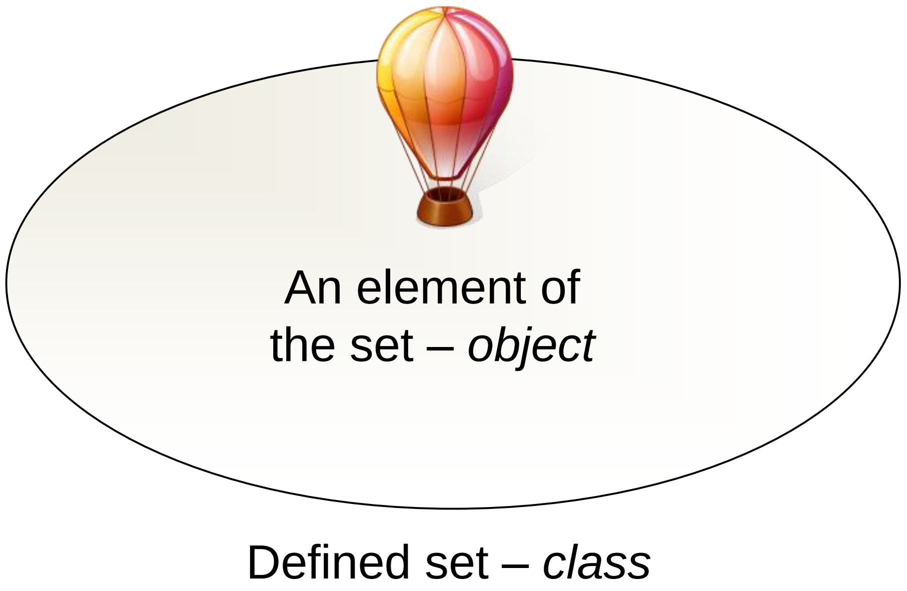
Progression of Programming Techniques



Object-Oriented Terminology

- Class
  - Outline of an idea
  - *Properties* (data)
  - *Methods* (algorithms)

- Object
  - Specific example of a *class*
  - *Instance*

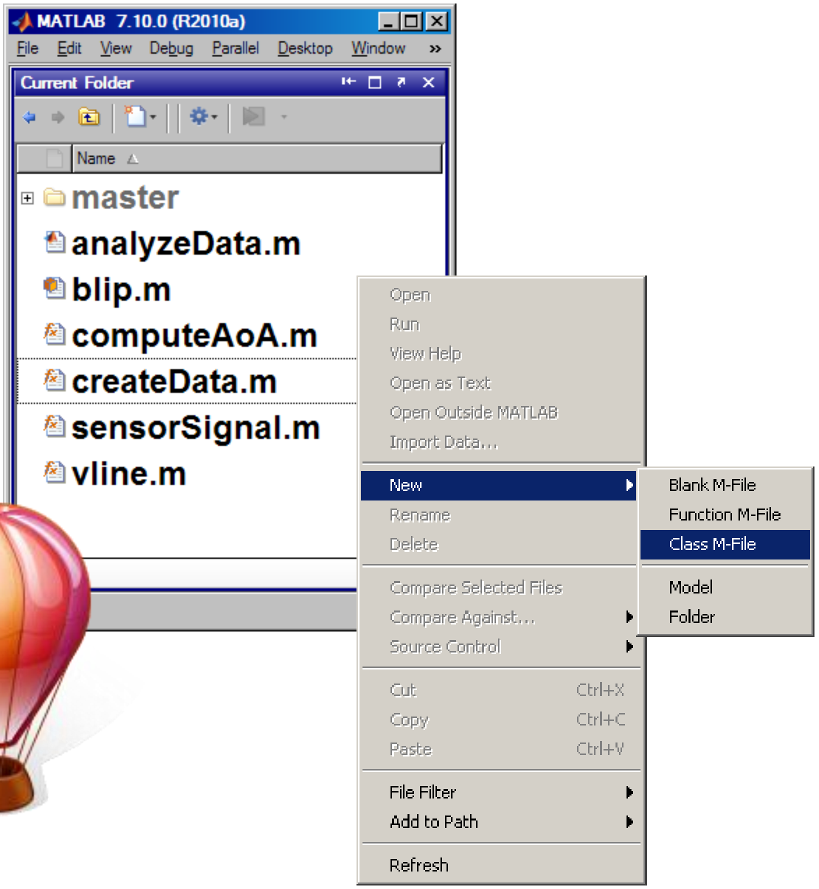


Agenda

- Object-oriented programming
- Basic object-oriented programming syntax in MATLAB
- Classes in MATLAB

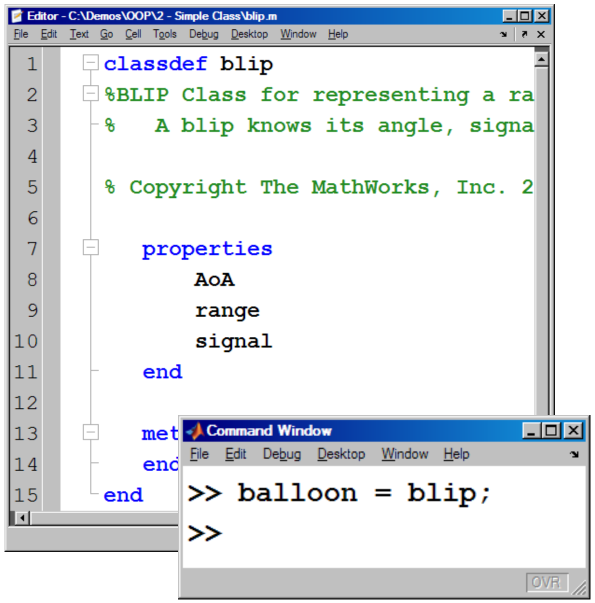
Demonstration: Building a Simple Class

- Define a class for our radar blips
- Create the weather balloon object
- Use the object in place of the structure



Objects

- Are easy to create
- Manage their own data
- Are interchangeable with a structure
  - No other code changes are required.
  - *Properties* behave similar to field names.
  - Fields can’t be added arbitrarily.



Editor - C:\Demos\OOP\2 - Simple Class\blip.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Command Window
File Edit Debug Desktop Window Help
OVR

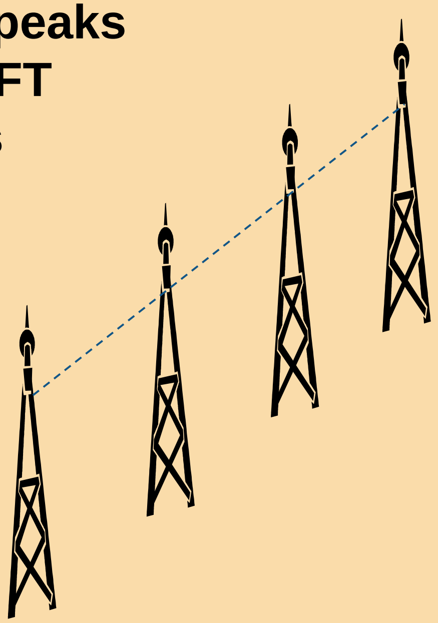
```
classdef blip
%BLIP Class for representing a ra
%   A blip knows its angle, signa

% Copyright The MathWorks, Inc. 2

    properties
        AoA
        range
        signal
    end

    met
    end
end
```

```
>> balloon = blip;
>>
```

Demonstration:
Adding Methods to a Class

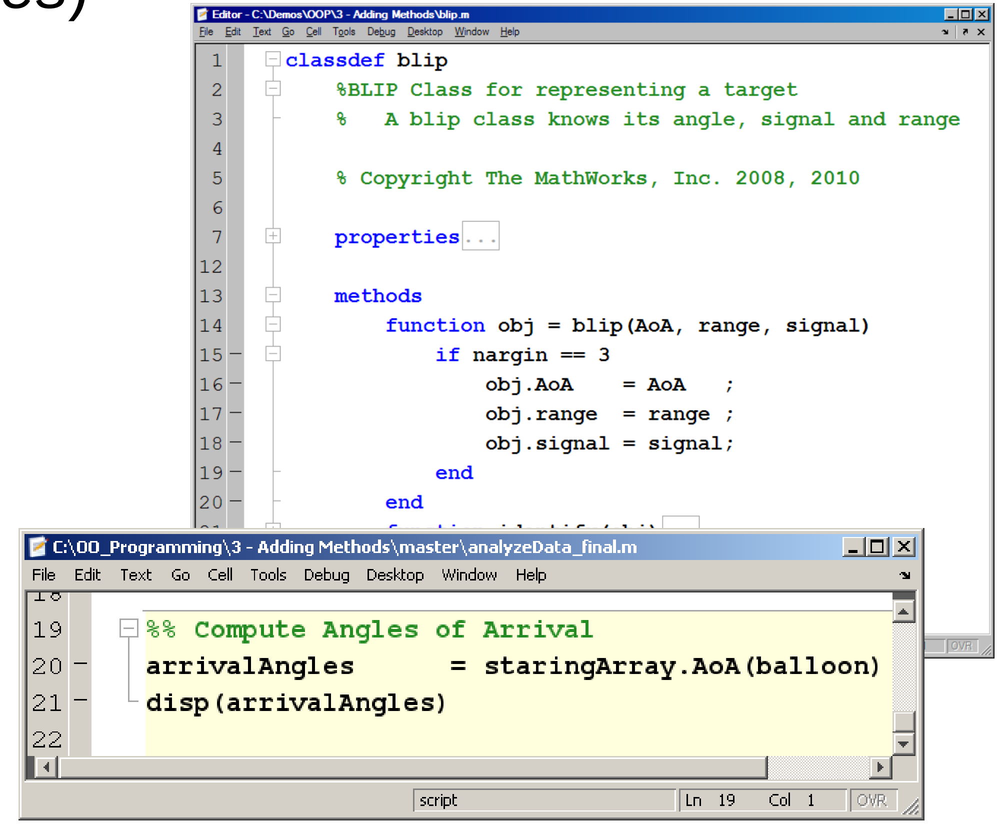
- Start from a sensor *class* with existing *properties*
- Add a *method* to compute angle of arrival (AoA)
- Integrate a sensor *object* into the existing code

Sensor
Synthesize measurements
Determine peaks
Compute FFT
Plot results



Objects with Methods

- Have immediate access to their own data (*properties*)
- Allow you to overload existing functions
- Allow you to perform custom actions at creation and deletion


Editor - C:\Demos\OOP\3 - Adding Methods\blip.m
File Edit Text Go Cell Tools Debug Desktop Window Help
C:\00_Programming\3 - Adding Methods\master\analyzeData_final.m
File Edit Text Go Cell Tools Debug Desktop Window Help
script
Ln 19 Col 1
OVR

```
classdef blip
    %BLIP Class for representing a target
    %   A blip class knows its angle, signal and range

    % Copyright The MathWorks, Inc. 2008, 2010

    properties...

    methods
        function obj = blip(AoA, range, signal)
            if nargin == 3
                obj.AoA     = AoA    ;
                obj.range   = range  ;
                obj.signal  = signal;
            end
        end
```

```
%% Compute Angles of Arrival
arrivalAngles       = staringArray.AoA(balloon)
disp(arrivalAngles)
```

Agenda

- Object-oriented programming
- Basic object-oriented programming syntax in MATLAB
- Classes in MATLAB

Classes in MATLAB

- Designed to ‘feel’ like MATLAB
  - Incorporates matrix indexing

    ```
    >> x = 2*anObject.itsProperty(1:end);
    ```

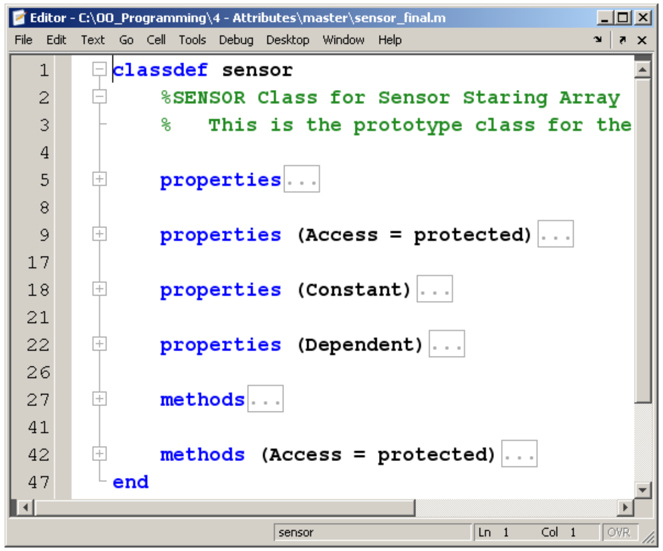
  - Inherently overloaded

    ```
    varargout = anObject.itsMethod(varargin)
    ```

- Works like an object-oriented language
  - Encapsulation, inheritance, etc.

Taking Methods and Properties Further

- Control access
- Create constants
- Make values interdependent
- Execute methods when properties change


Editor - C:\00_Programming\4 - Attributes\master\sensor_final.m
File Edit Text Go Cell Tools Debug Desktop Window Help

```
classdef sensor
    %SENSOR Class for Sensor Staring Array
    %   This is the prototype class for the

    properties...

    properties (Access = protected)...

    properties (Constant)...

    properties (Dependent)...

    methods...

    methods (Access = protected)...
end
```

Demonstration: Applying Attributes

- Control access

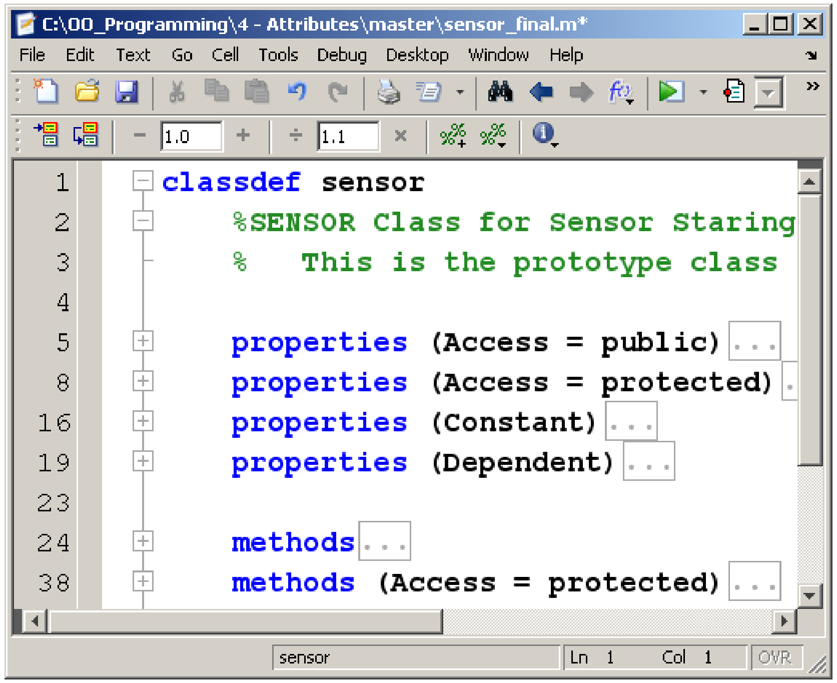
  `Access = public`

  `Access = protected`

- Restrict modification

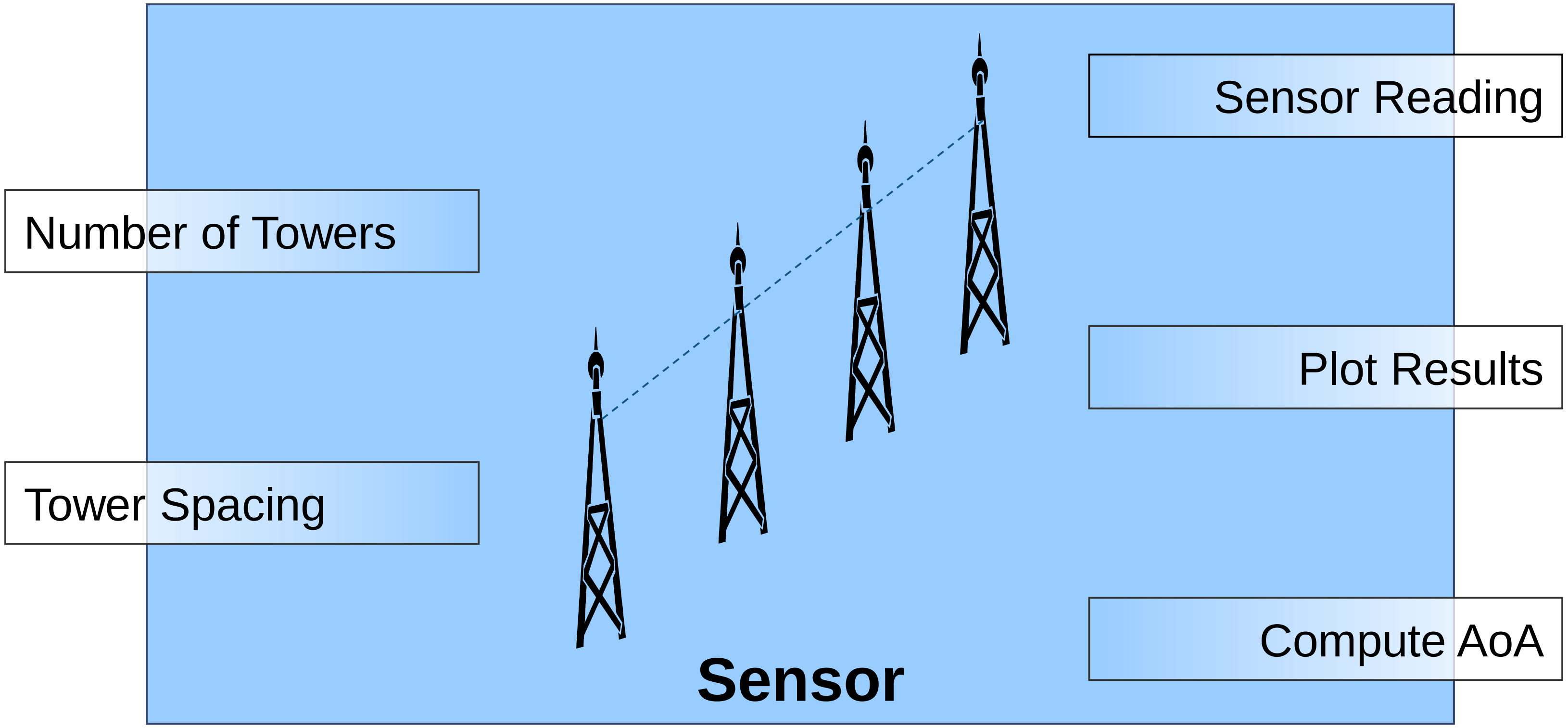
  `Constant`

  `Dependent`


C:\OO_Programming\4 - Attributes\master\sensor_final.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
sensor
Ln 1 Col 1 OVR

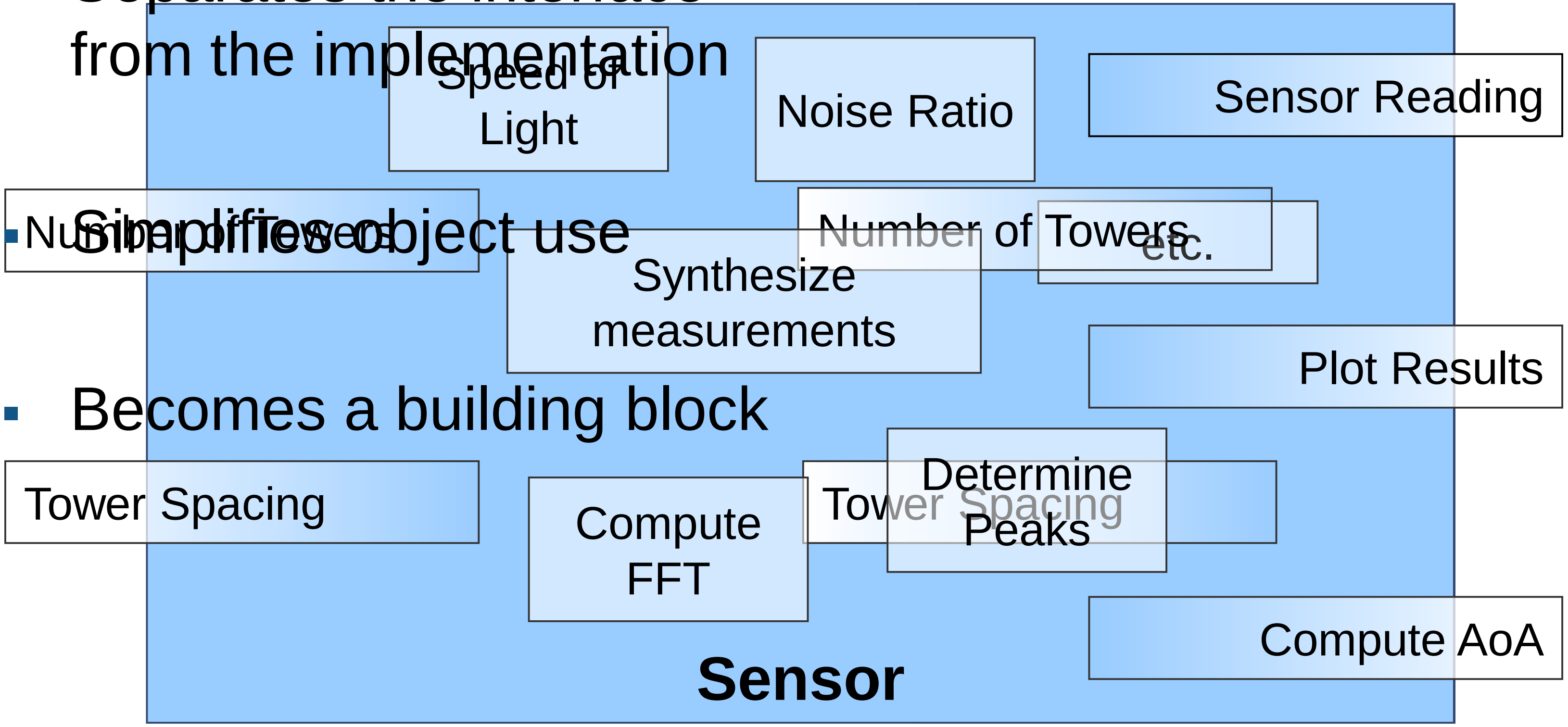
```
classdef sensor
    %SENSOR Class for Sensor Staring
    %   This is the prototype class

    properties (Access = public)...
    properties (Access = protected)...
    properties (Constant)...
    properties (Dependent)...

    methods...
    methods (Access = protected)...
```

Encapsulation

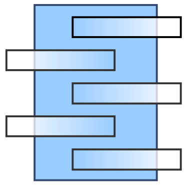


Encapsulation

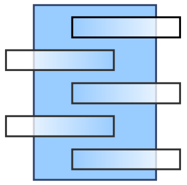
- Separates the interface from the implementation
- Simplifies object use
- Becomes a building block



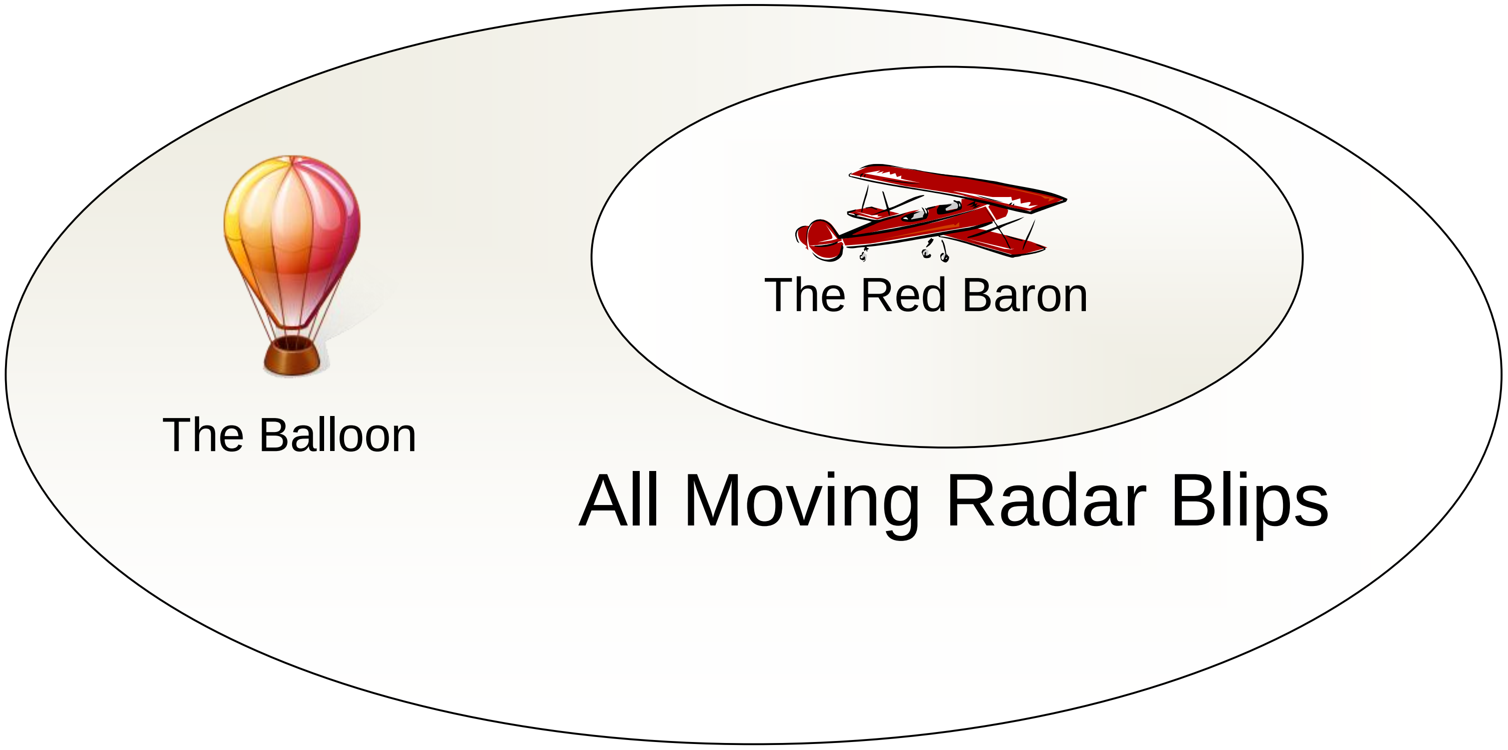
Using an Object as a Building Block



```
Assignment
Looping Test
    Increment
    Test to Act
        Take Action
    End
End
```



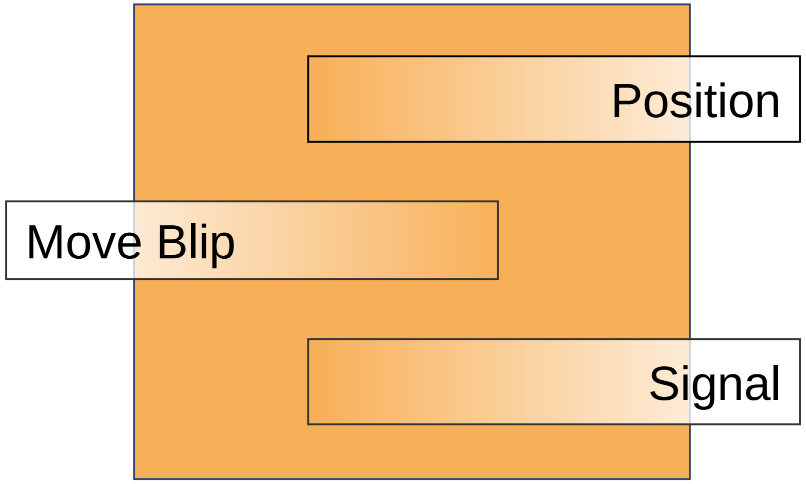
Using a Class as a Building Block



All Radar Blips

Demonstration: Creating a Moving Target

- Define a new *class* for moving blips
- *Inherit* from the existing *class* for blips
- Add a *method*
- Use the moving blip

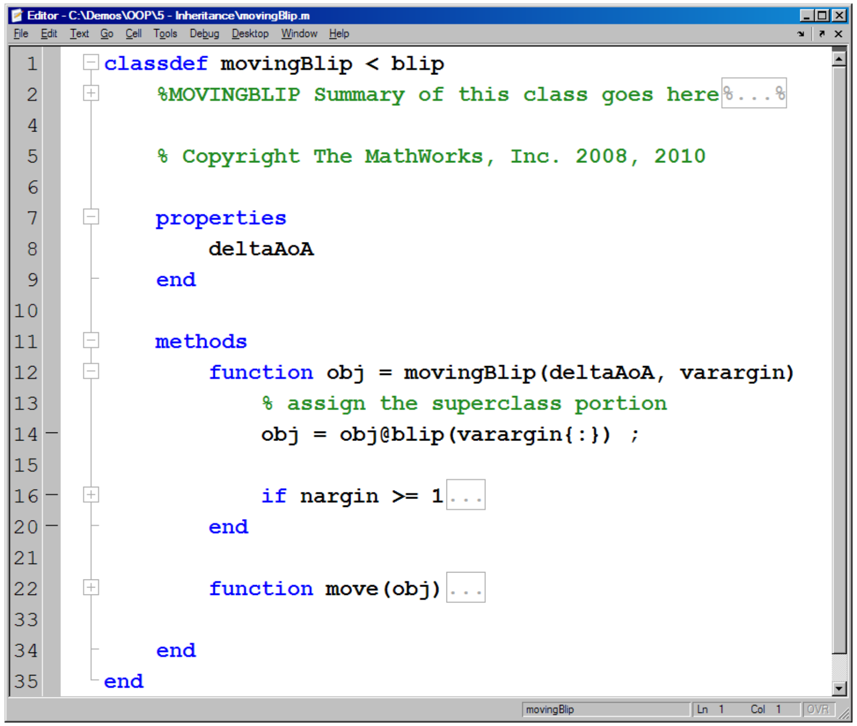


Inheritance

- *Subclass* substitutes for the *superclass*

- Allows re-envisioning and re-implementing the *superclass*

- Builds on proven code


Editor - C:\Demos\OOP\5 - Inheritance\movingBlip.m
File Edit Text Go Cell Tools Debug Desktop Window Help
movingBlip
Ln 1 Col 1 OVR

```
classdef movingBlip < blip
    %MOVINGBLIP Summary of this class goes here%...%

    % Copyright The MathWorks, Inc. 2008, 2010

    properties
        deltaAoA
    end

    methods
        function obj = movingBlip(deltaAoA, varargin)
            % assign the superclass portion
            obj = obj@blip(varargin{:}) ;

            if nargin >= 1...
        end

        function move(obj)...

    end
end
```

- Allows inheriting from the base MATLAB classes

How does ‘=’ work in MATLAB? *Round 1*

```
>> a = 10000;
>> b = a;
>> b = 20000;
>> disp(a)
```

[✓ a) 10,000]

b) 20,000

c) Something else

d) No idea

How does '=' work in MATLAB?
Round 2

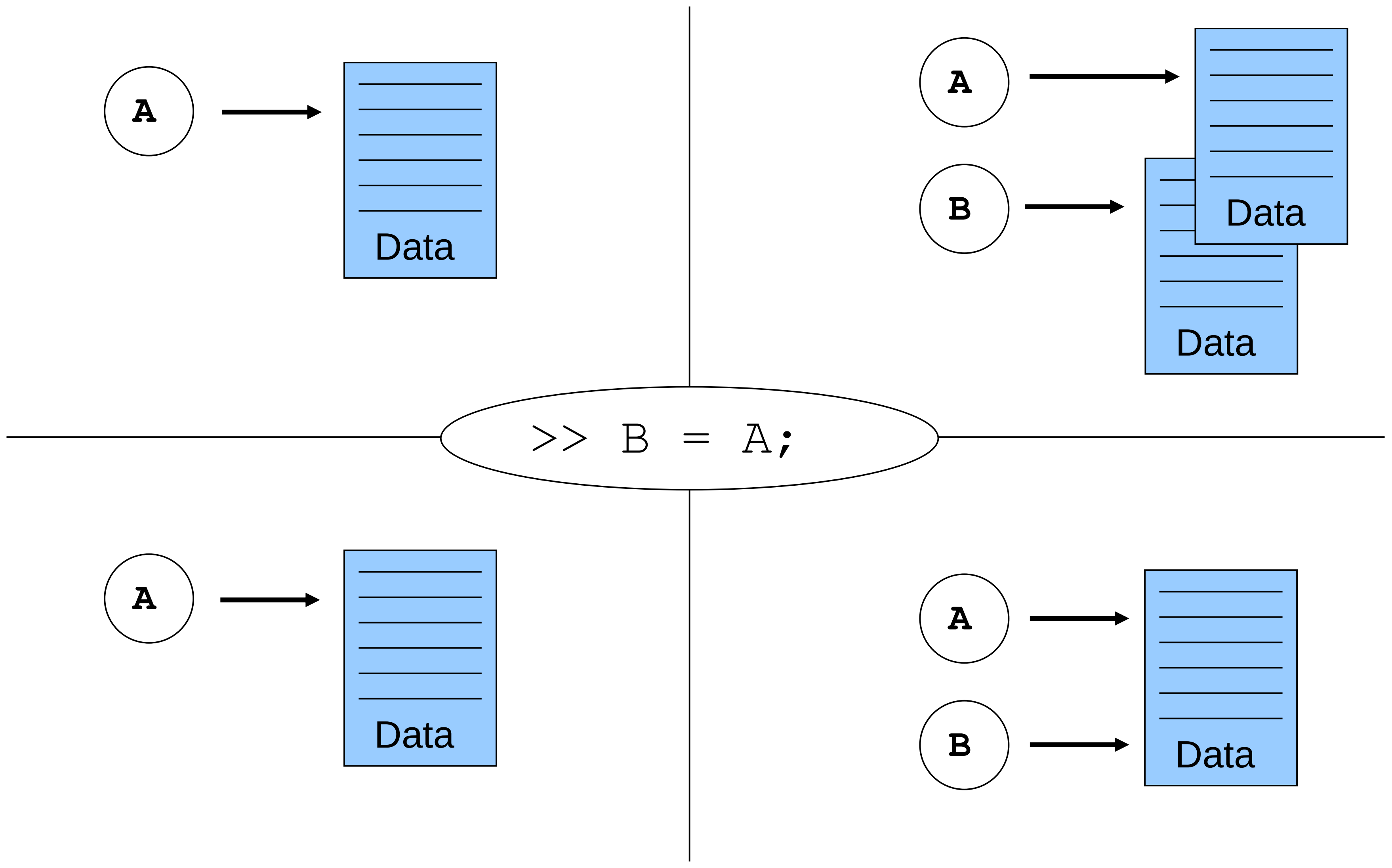
```
>> a = analoginput('winsound'); addchannel(a,1);
>> a.SampleRate = 10000;
>> b = a;
>> b.SampleRate = 20000;
>> disp(a.SampleRate)
```

a) 10,000

[✓ b) 20,000]

c) Something else

d) No idea

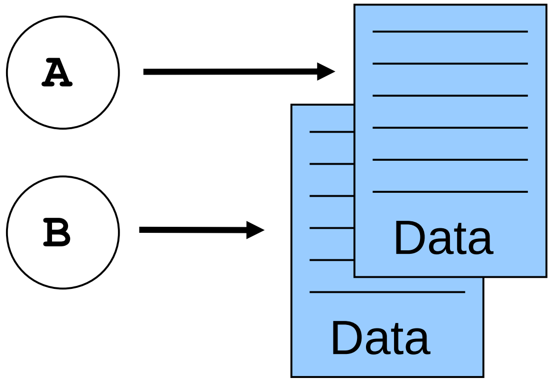

A
Data
A
B
Data
Data
>> B = A;
A
Data
A
B
Data

Value Class

MATLAB default

'=' *copies* data

data in workspace

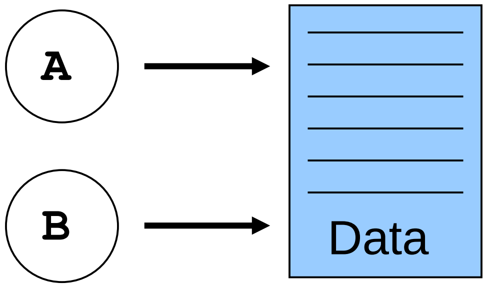


Handle Class

Use: `< handle`

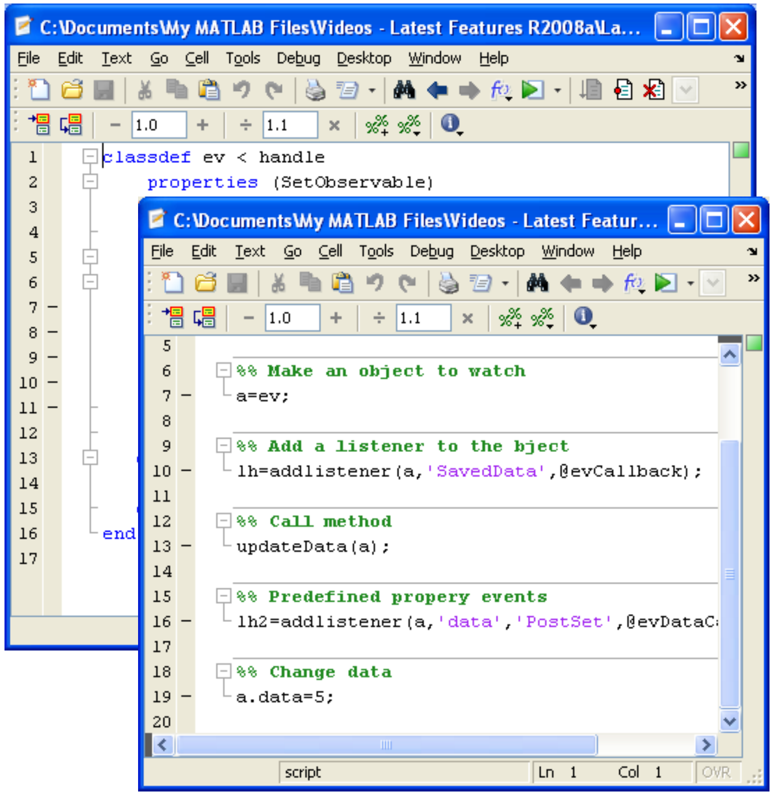
'=' *references* data

handle in workspace



Using Events and Listeners

- Events
  - Created in a handle object
  - `events` block in `classdef`
  - `notify(…)` triggers event
- Listeners
  - Triggers call back function
  - `addlistener(…)`
  - Useable anywhere


C:\Documents\My MATLAB Files\Videos - Latest Features R2008a\La...
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1

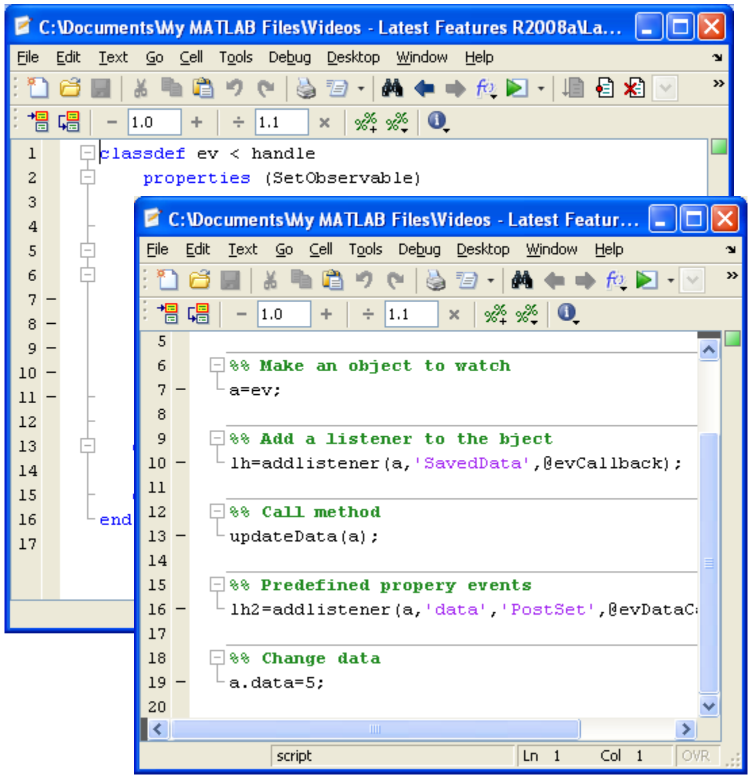
```
classdef ev < handle
    properties (SetObservable)




end
```

C:\Documents\My MATLAB Files\Videos - Latest Featur...
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
script
Ln 1
Col 1
OVR

```

%% Make an object to watch
a=ev;

%% Add a listener to the bject
lh=addlistener(a,'SavedData',@evCallback);

%% Call method
updateData(a);

%% Predefined propery events
lh2=addlistener(a,'data','PostSet',@evDataC

%% Change data
a.data=5;

```

Events and Listeners

- Uses technology related to
  - `preSet`
  - `postSet`
  - `preGet`
  - `postGet`
- Gives the ability to trigger action
- Anything can listen to an observable object


C:\Documents\My MATLAB Files\Videos - Latest Features R2008a\La...
File Edit Text Go Cell Tools Debug Desktop Window Help
C:\Documents\My MATLAB Files\Videos - Latest Featur...
script
Ln 1 Col 1
OVR

```
classdef ev < handle
    properties (SetObservable)
end
```

```
%% Make an object to watch
a=ev;

%% Add a listener to the bject
lh=addlistener(a,'SavedData',@evCallback);

%% Call method
updateData(a);

%% Predefined propery events
lh2=addlistener(a,'data','PostSet',@evDataC

%% Change data
a.data=5;
```

Object-Oriented Programming in MATLAB

- Class definition file describes object behavior
- Objects can substitute for structures
- Apply attributes for a clean interface
- Build on existing classes with inheritance

Extends the matrix-based language to objects

Additional Resources

Home | Select Country | Contact Us | Sto

The MathWorks
Accelerating the pace of engineering and science

Products & Services | Solutions | Academia

MATLAB Overview
- Description
- Function List
- Demos and Webinars
- System Requirements
- Latest Features

Support & Training
- Product Support
- Documentation
- Downloads & Trials
- Training
- Consulting

Other Resources
- Technical Literature
- User Stories
- Related Books
- News and Events

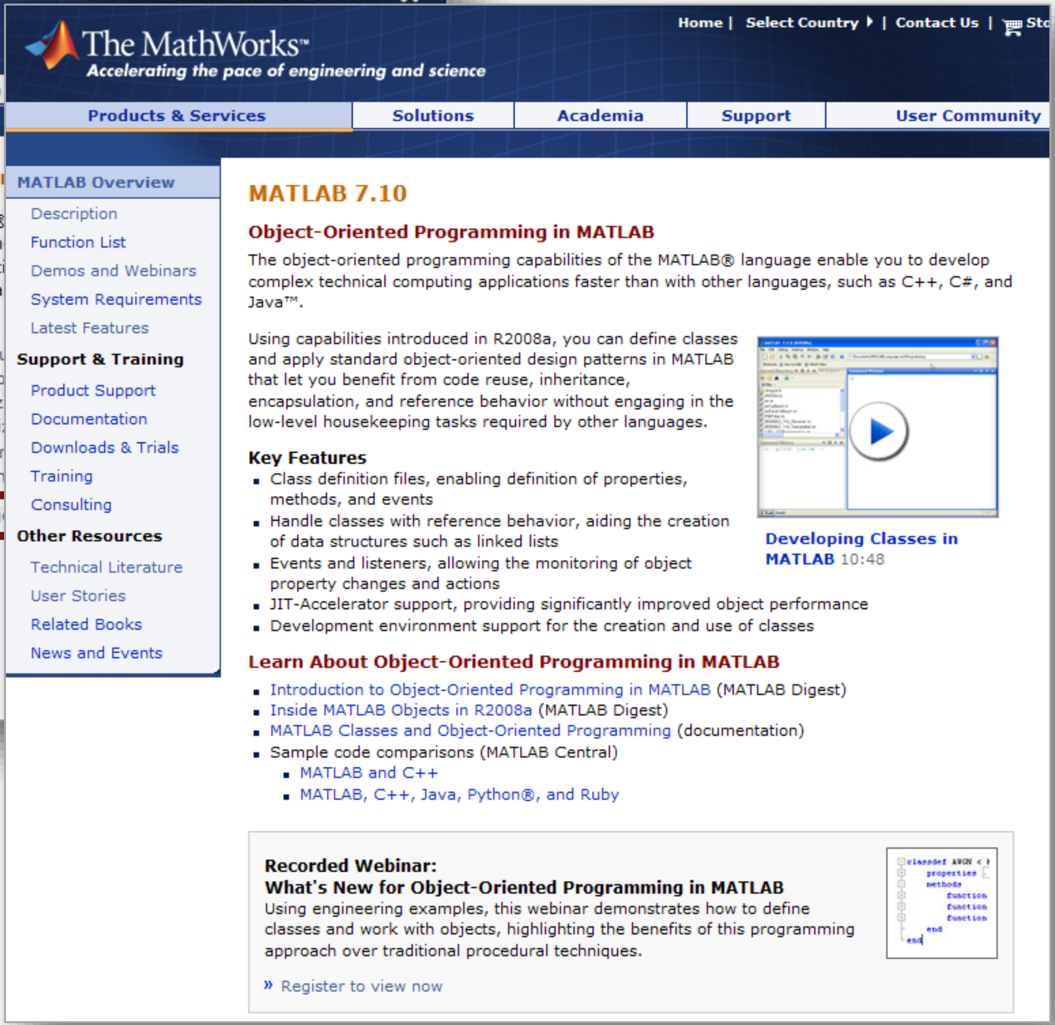
MATLAB - The Language Of Technical Co

MATLAB®
environm
computati
traditiona
Fortran.

- Introdu
- Develo
- Analyz
- Visuali
- Perforr
- Publish

NEW Obj

Learn more about MATLAB
- View a Video: Getting Started with MATLAB
- Watch a Webinar: Introduction to MATLAB
- Get MATLAB Trial Software
- Contact Sales | Get a Quote



Home | Select Country | Contact Us | Sto

The MathWorks
Accelerating the pace of engineering and science

Products & Services | Solutions | Academia | Support | User Community

MATLAB Overview
- Description
- Function List
- Demos and Webinars
- System Requirements
- Latest Features

Support & Training
- Product Support
- Documentation
- Downloads & Trials
- Training
- Consulting

Other Resources
- Technical Literature
- User Stories
- Related Books
- News and Events

MATLAB 7.10

Object-Oriented Programming in MATLAB

The object-oriented programming capabilities of the MATLAB® language enable you to develop complex technical computing applications faster than with other languages, such as C++, C#, and Java™.

Using capabilities introduced in R2008a, you can define classes and apply standard object-oriented design patterns in MATLAB that let you benefit from code reuse, inheritance, encapsulation, and reference behavior without engaging in the low-level housekeeping tasks required by other languages.

Developing Classes in MATLAB 10:48

Key Features
- Class definition files, enabling definition of properties, methods, and events
- Handle classes with reference behavior, aiding the creation of data structures such as linked lists
- Events and listeners, allowing the monitoring of object property changes and actions
- JIT-Accelerator support, providing significantly improved object performance
- Development environment support for the creation and use of classes

Learn About Object-Oriented Programming in MATLAB
- Introduction to Object-Oriented Programming in MATLAB (MATLAB Digest)
- Inside MATLAB Objects in R2008a (MATLAB Digest)
- MATLAB Classes and Object-Oriented Programming (documentation)
- Sample code comparisons (MATLAB Central)
  - MATLAB and C++
  - MATLAB, C++, Java, Python®, and Ruby

Recorded Webinar:
What's New for Object-Oriented Programming in MATLAB
Using engineering examples, this webinar demonstrates how to define classes and work with objects, highlighting the benefits of this programming approach over traditional procedural techniques.

» Register to view now

Questions and Answers