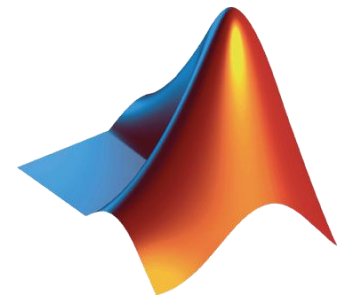


# Introduction to Object-Oriented Programming in MATLAB

**Jamie Winter**  
**Sr. Account Manager**

**Abhishek Gupta**  
**Application Engineer**



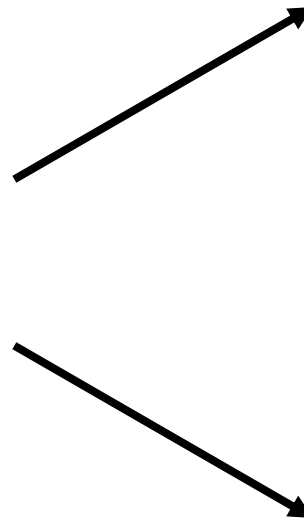
# Agenda

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

# What is a program?

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

## Code



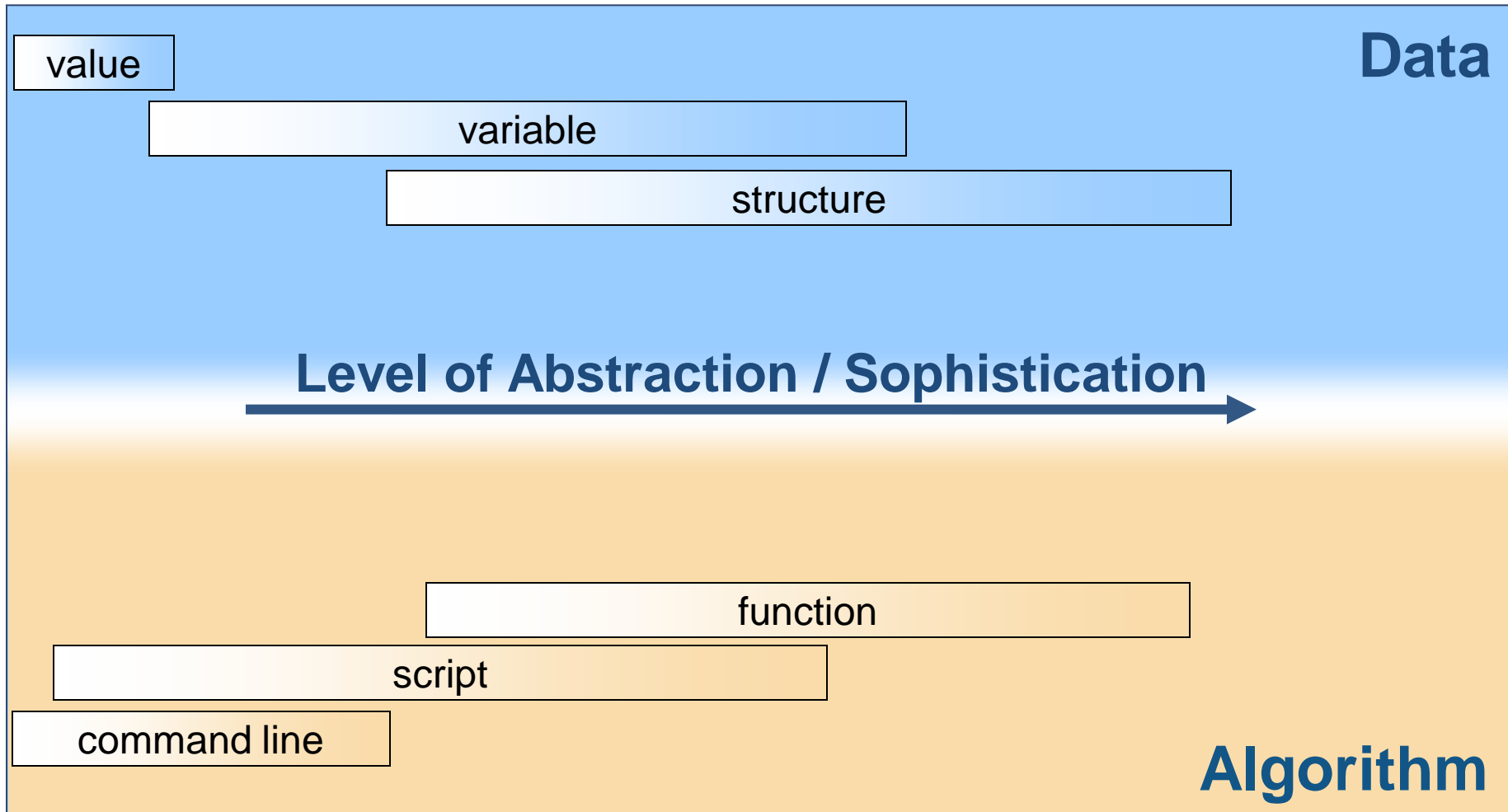
## Data

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

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

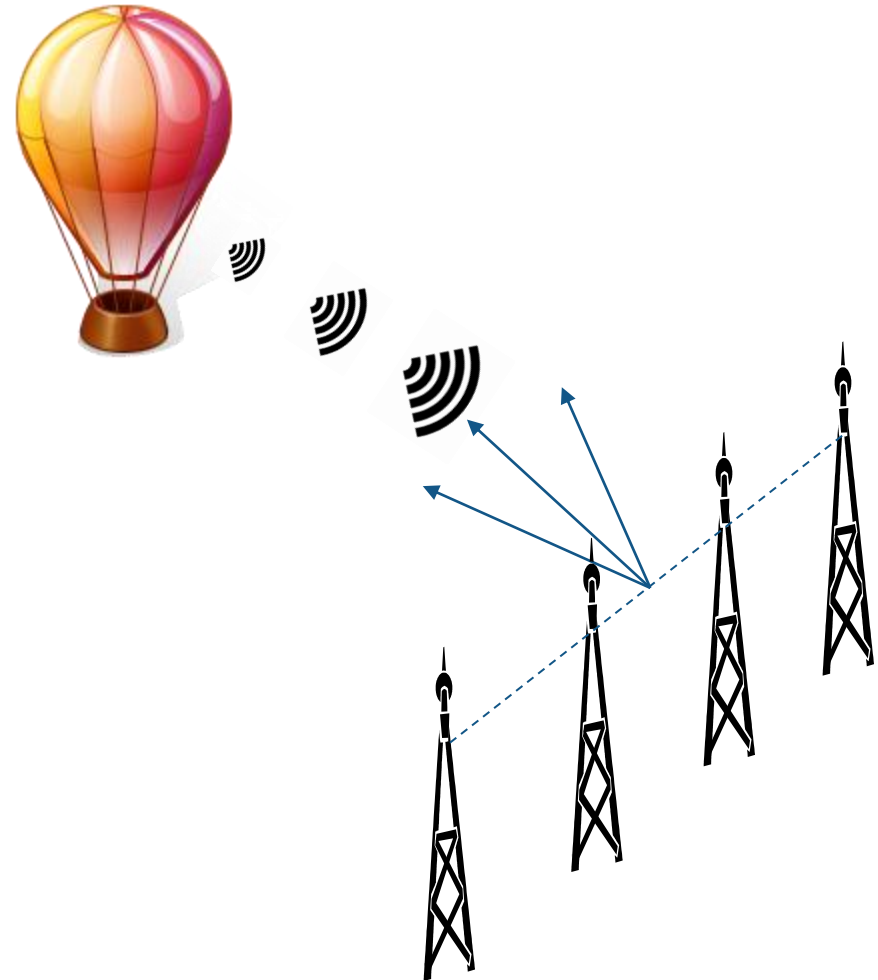
## Actions

# Progression of Programming Techniques



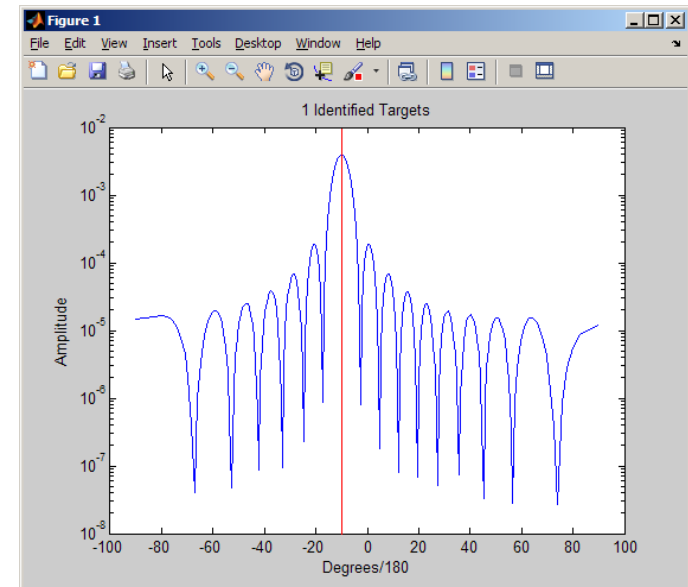
## Example: Sensor Array

- Transmitting a signal from a weather balloon
- Locating the signal with a sensor array
- Computing the angle of arrival (AoA) for the signal

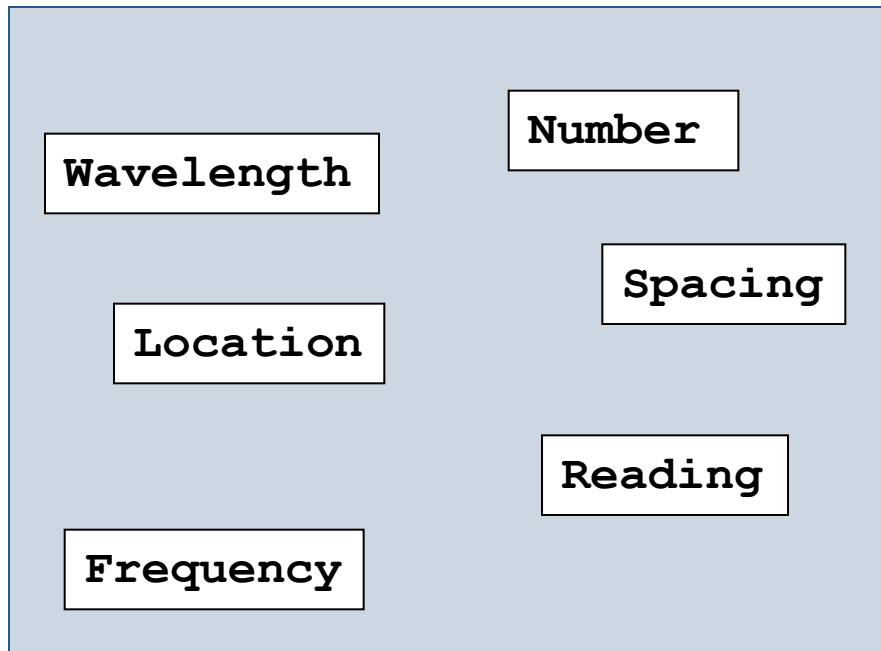


# Procedural Programming

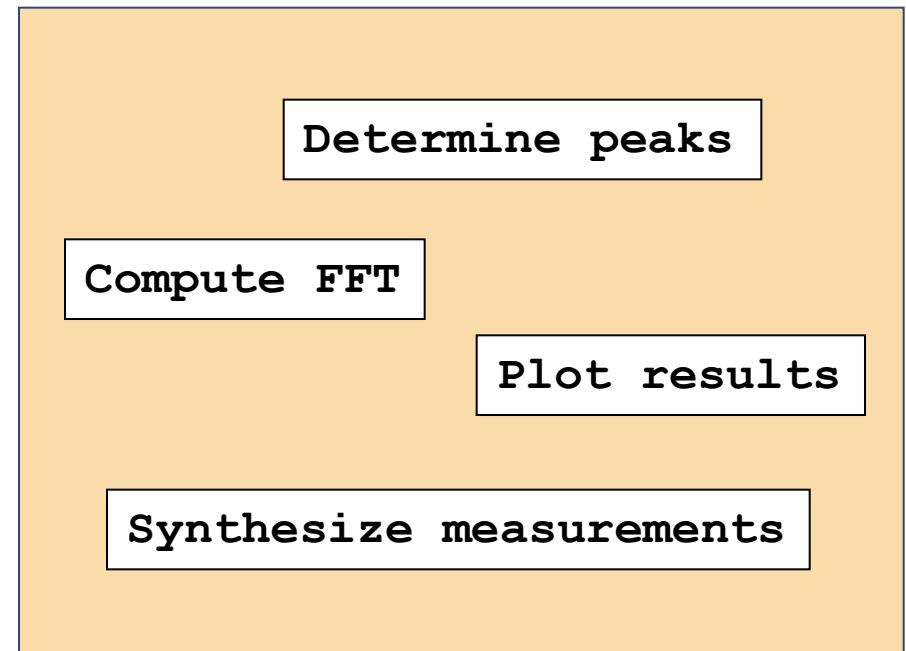
- Easy to learn
- Minimal planning
  
- There is no formal relationship between data and functions.
- Every detail is exposed.



# Data and Actions to Implement

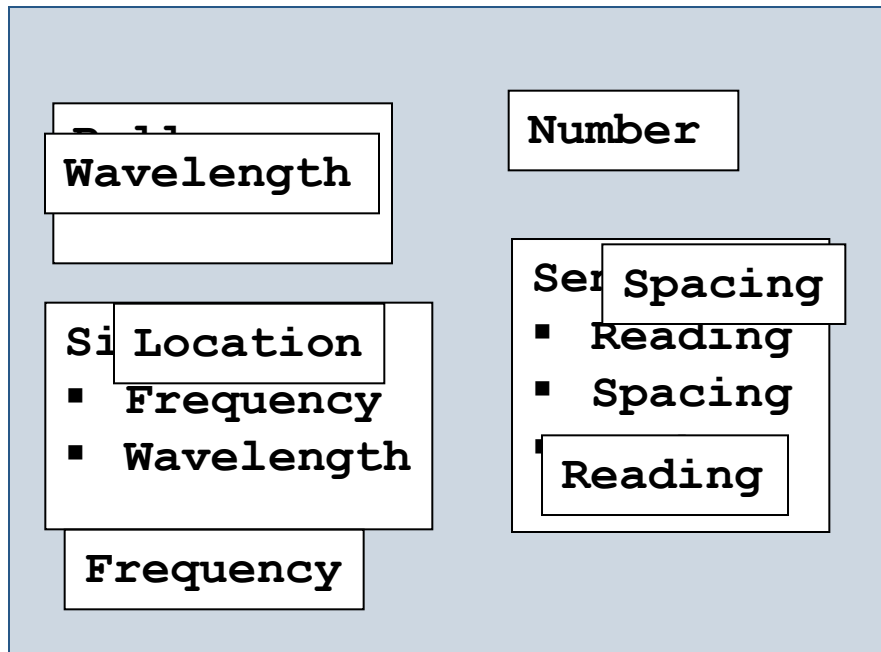


**Data**

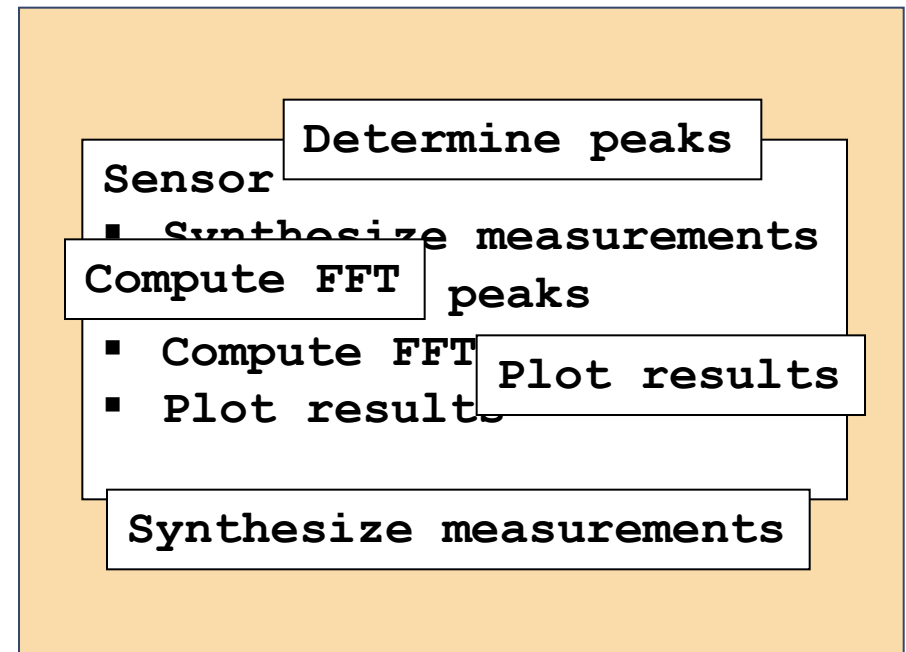


**Actions**

# Related Data and Actions



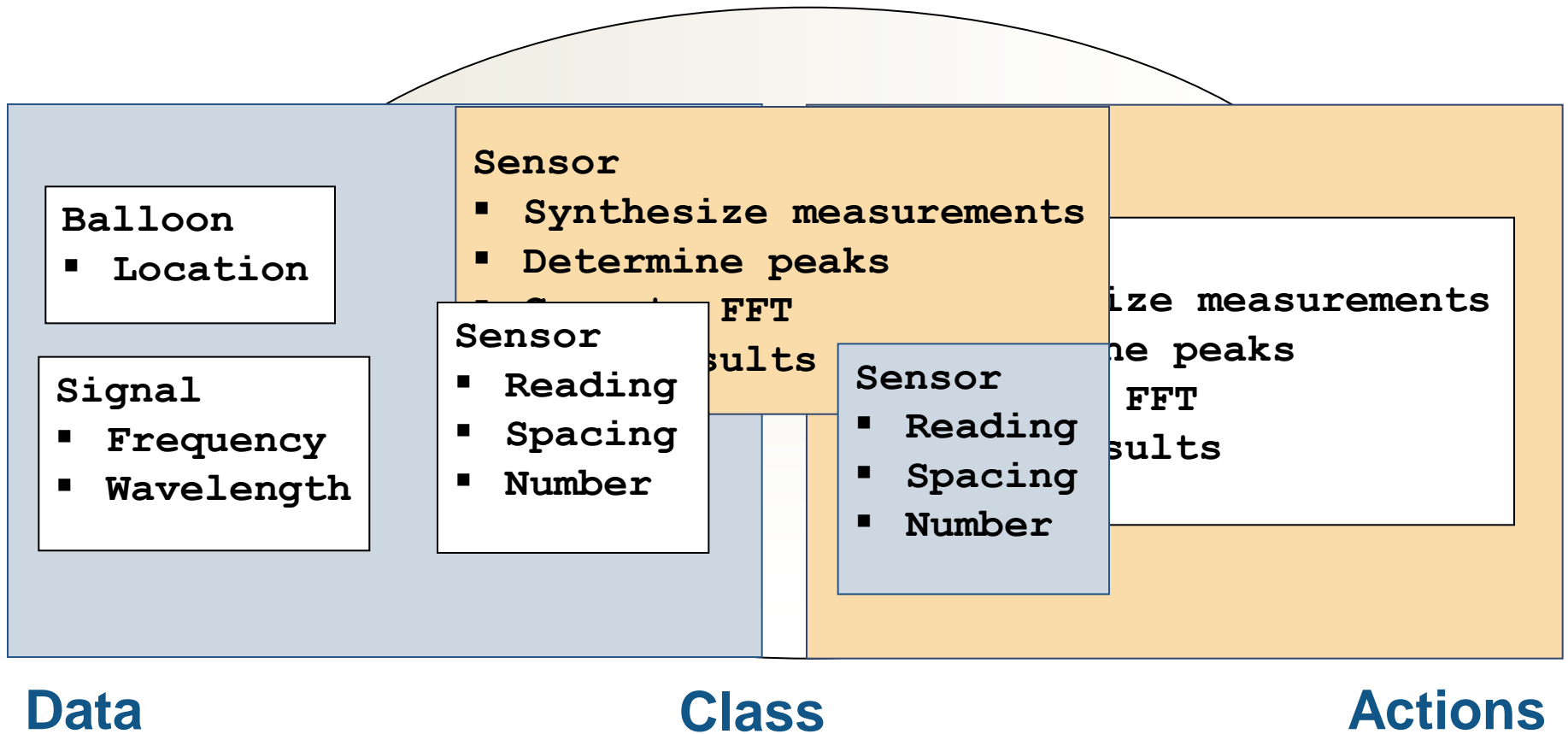
Data



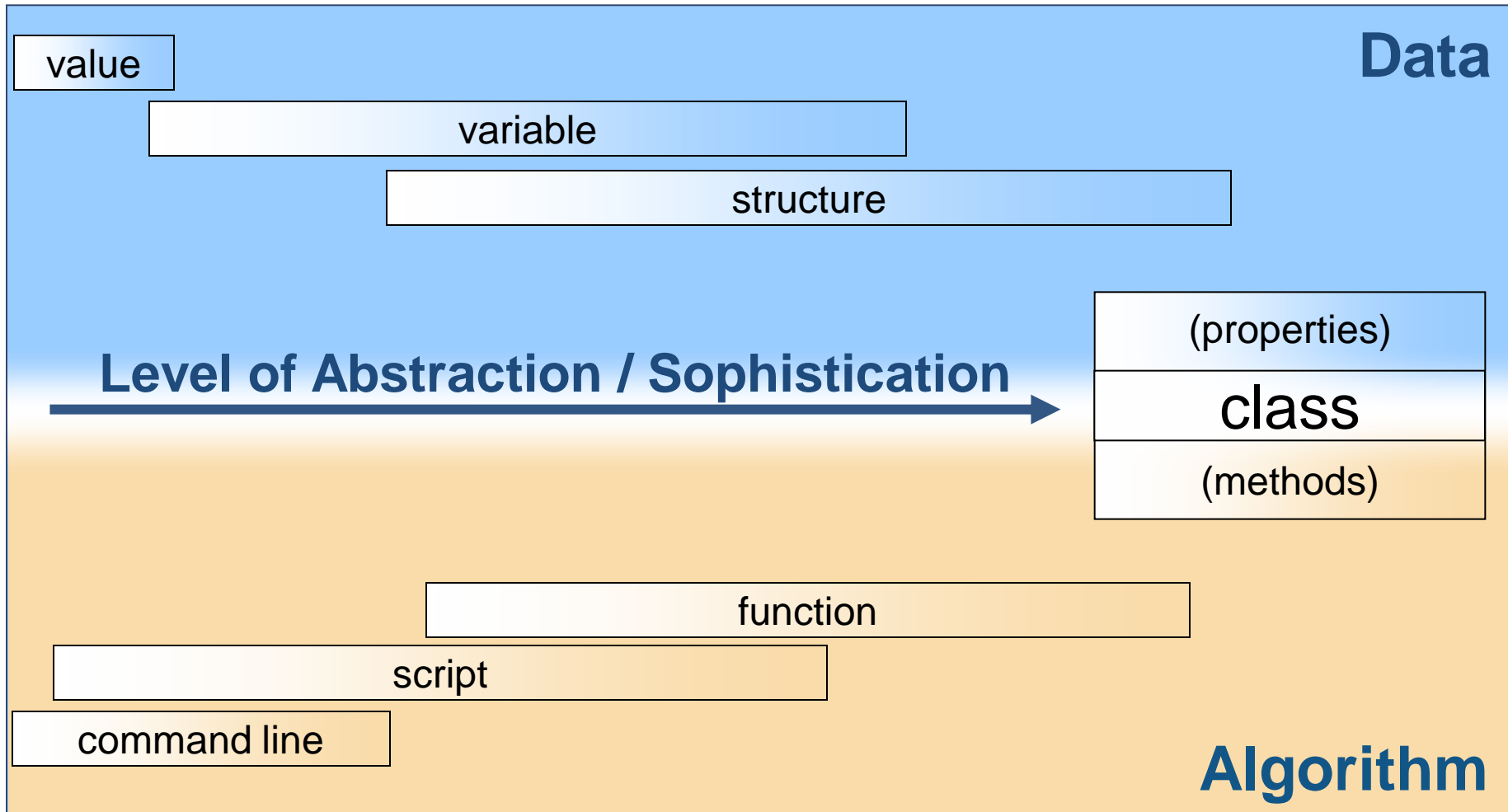
Actions



# Grouping Related Data and Actions

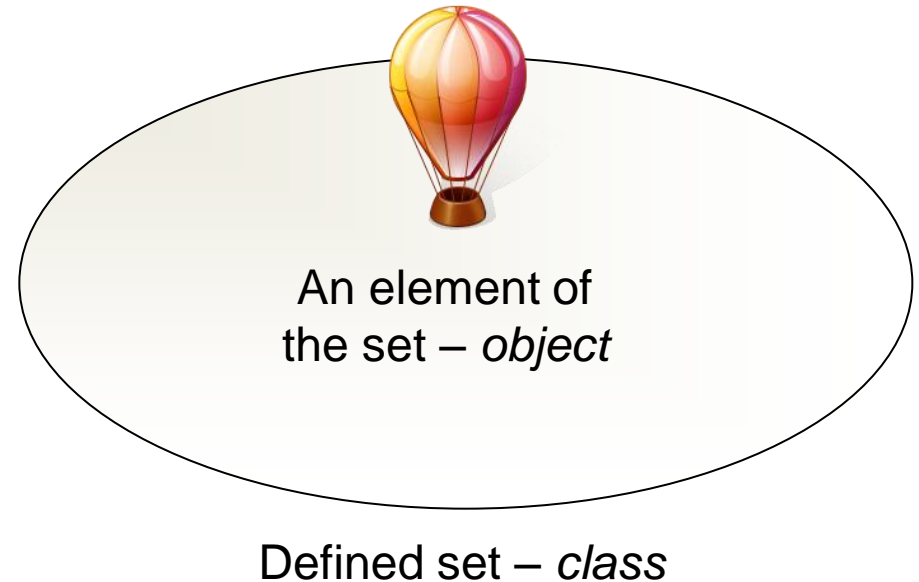


# Progression of Programming Techniques




# Object-Oriented Terminology

- **Class**
  - Outline of an idea
  - *Properties* (data)
  - *Methods* (algorithms)
  
- **Object**
  - Specific example of a *class*
  - *Instance*

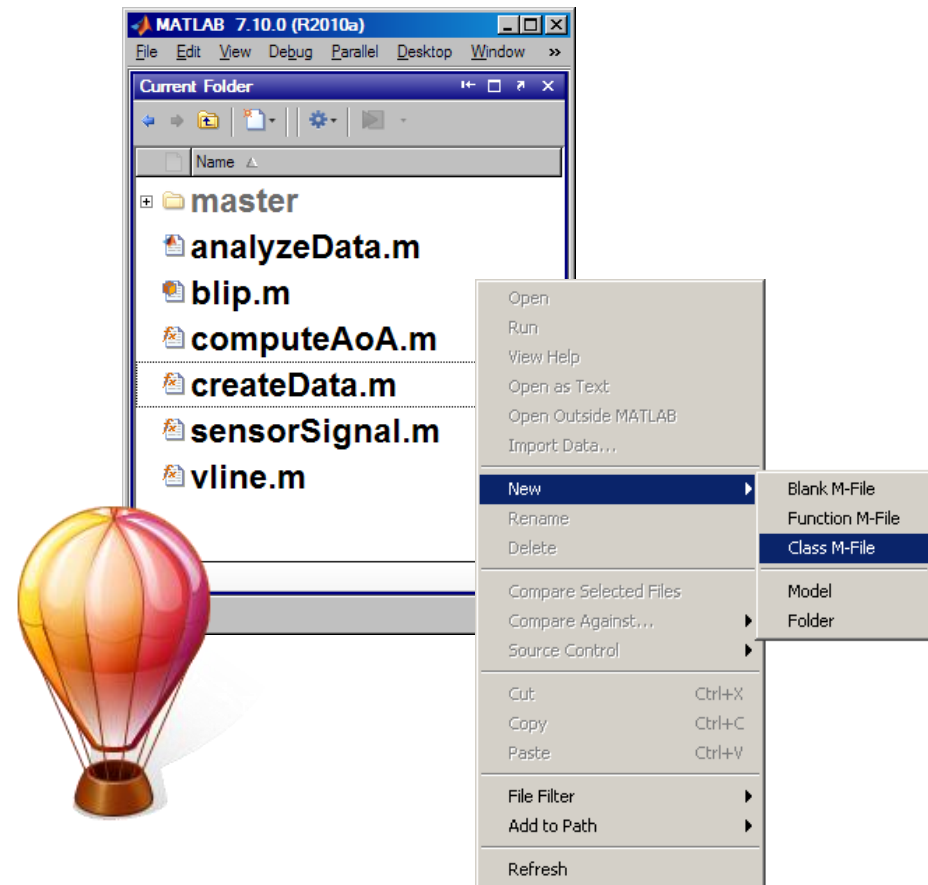


# Agenda

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

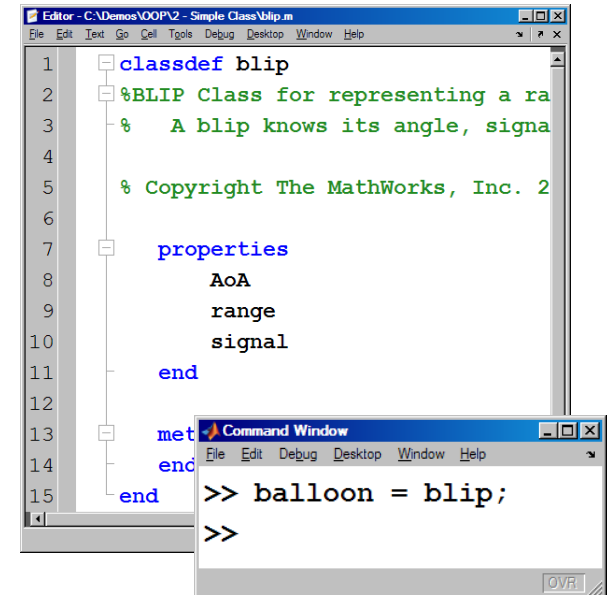
# Demonstration: Building a Simple Class

- 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.



The screenshot shows a MATLAB Editor window titled "Editor - C:\Demos\OOP\2 - Simple Class\blip.m" and a Command Window window titled "Command Window".

The Editor window contains the following code:

```

1  classdef blip
2  %BLIP Class for representing a ra
3  % A blip knows its angle, signa
4
5  % Copyright The MathWorks, Inc. 2
6
7  properties
8      AoA
9      range
10     signal
11 end
12
13 met
14 end
15 end

```

The Command Window shows the following command and output:

```

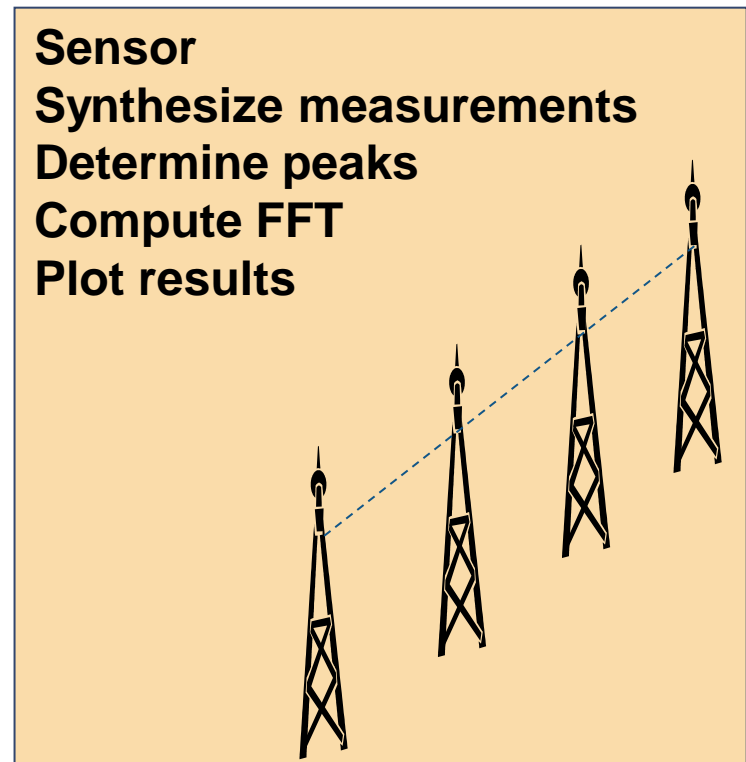
>> balloon = blip;
>>

```



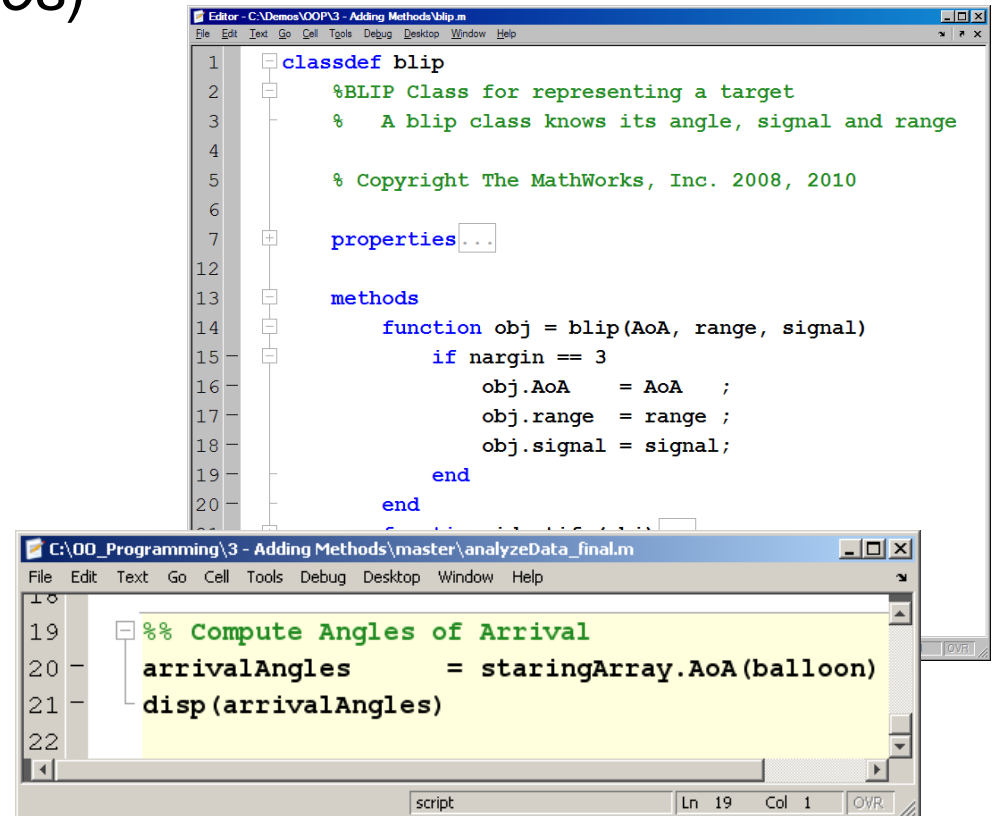
## Demonstration: Adding Methods to a Class

- 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



# 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



```

1 classdef blip
2     %BLIP Class for representing a target
3     % A blip class knows its angle, signal and range
4
5     % Copyright The MathWorks, Inc. 2008, 2010
6
7     properties ...
12
13     methods
14         function obj = blip(AoA, range, signal)
15             if nargin == 3
16                 obj.AoA = AoA ;
17                 obj.range = range ;
18                 obj.signal = signal;
19             end
20         end
    end
end

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



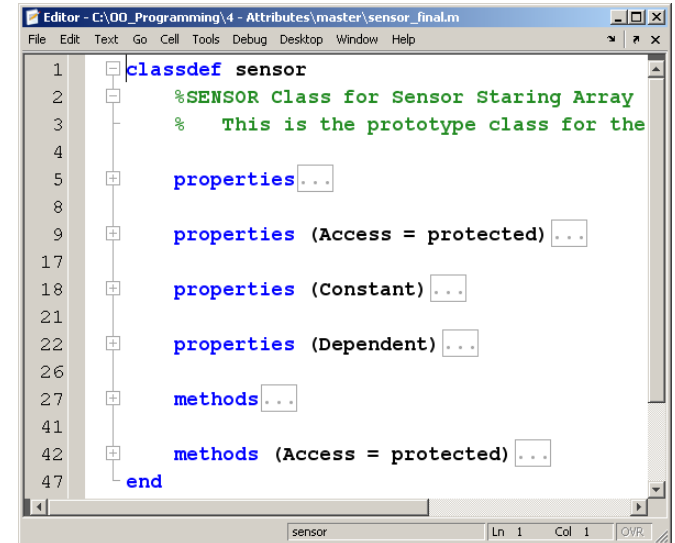
# Agenda

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

-  ▪ Classes in MATLAB

# Taking Methods and Properties Further

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



```

Editor - C:\OO_Programming\4 - Attributes\master\sensor_final.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 classdef sensor
2 %SENSOR Class for Sensor Staring Array
3 % This is the prototype class for the
4
5 properties ...
8
9 properties (Access = protected) ...
17
18 properties (Constant) ...
21
22 properties (Dependent) ...
26
27 methods ...
41
42 methods (Access = protected) ...
47 end
  
```

# Demonstration: Applying Attributes

- Control access

Access = public

Access = protected

- Restrict modification

Constant

Dependent

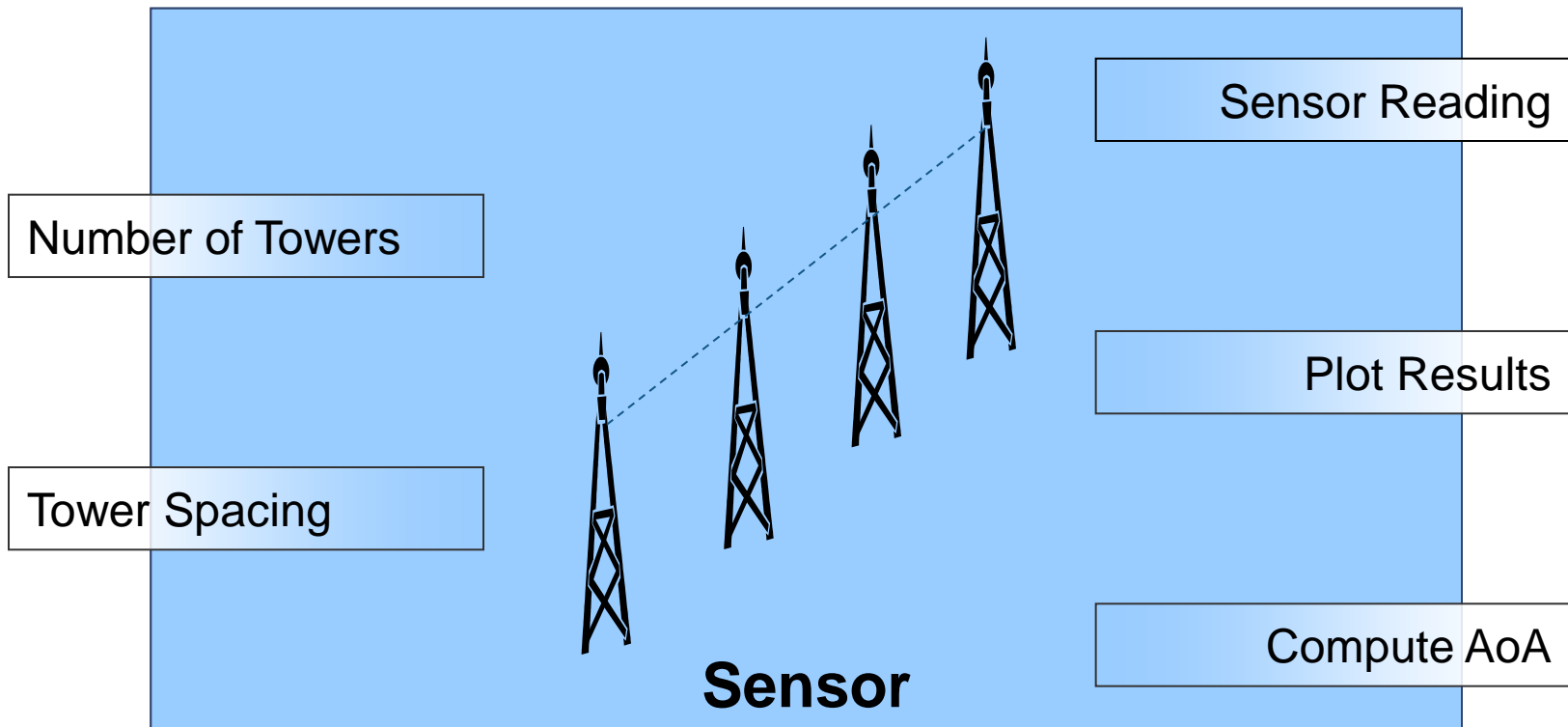
The screenshot shows a MATLAB editor window titled 'C:\00\_Programming\4 - Attributes\master\sensor\_final.m\*'. The code defines a class named 'sensor' with the following attributes:

```

1  classdef sensor
2      %SENSOR Class for Sensor Staring
3      % This is the prototype class
4
5      properties (Access = public) ...
8      properties (Access = protected) ...
16     properties (Constant) ...
19     properties (Dependent) ...
23
24     methods ...
38     methods (Access = protected) ...
  
```

The editor window includes a menu bar (File, Edit, Text, Go, Cell, Tools, Debug, Desktop, Window, Help), a toolbar with various icons, and a status bar at the bottom showing 'sensor', 'Ln 1', 'Col 1', and 'OVR'.

# Encapsulation

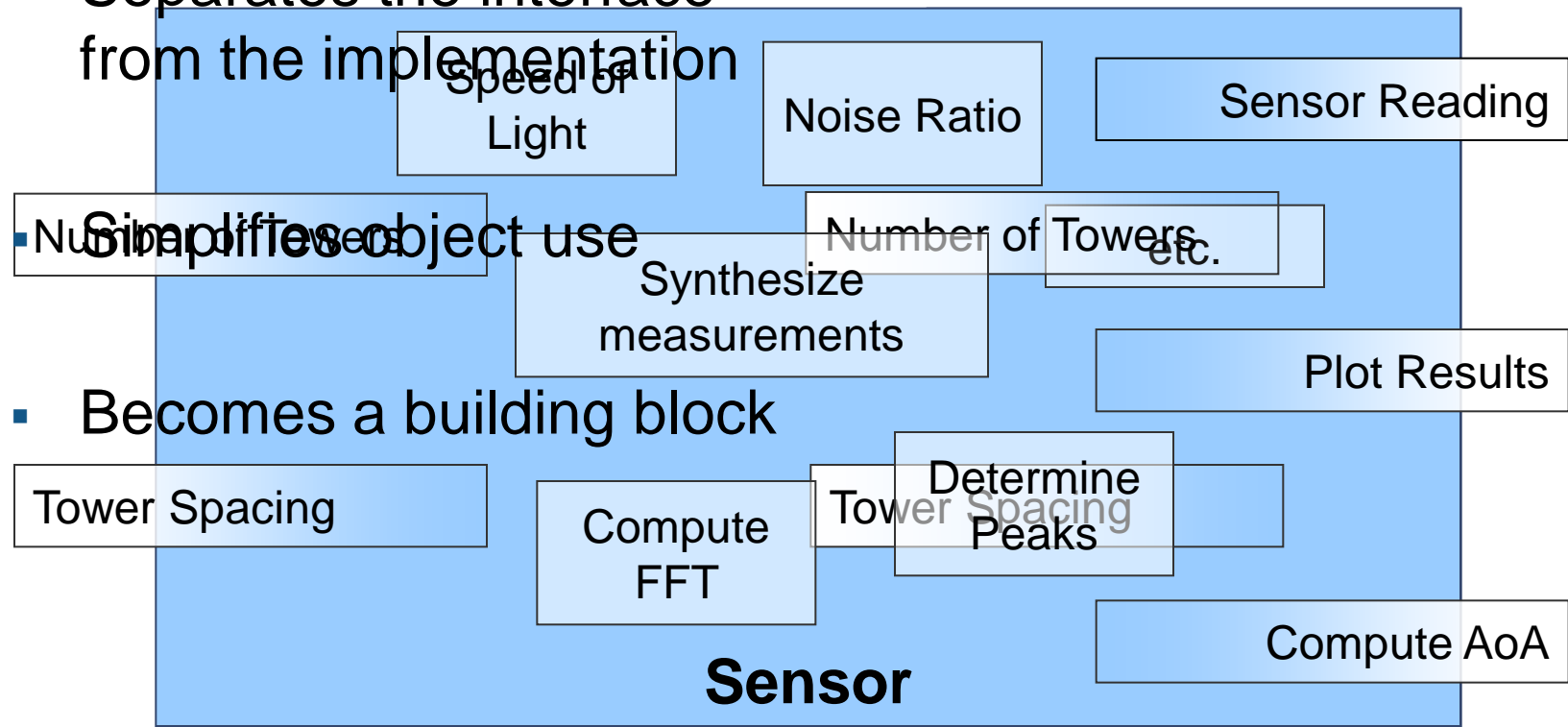


# Encapsulation

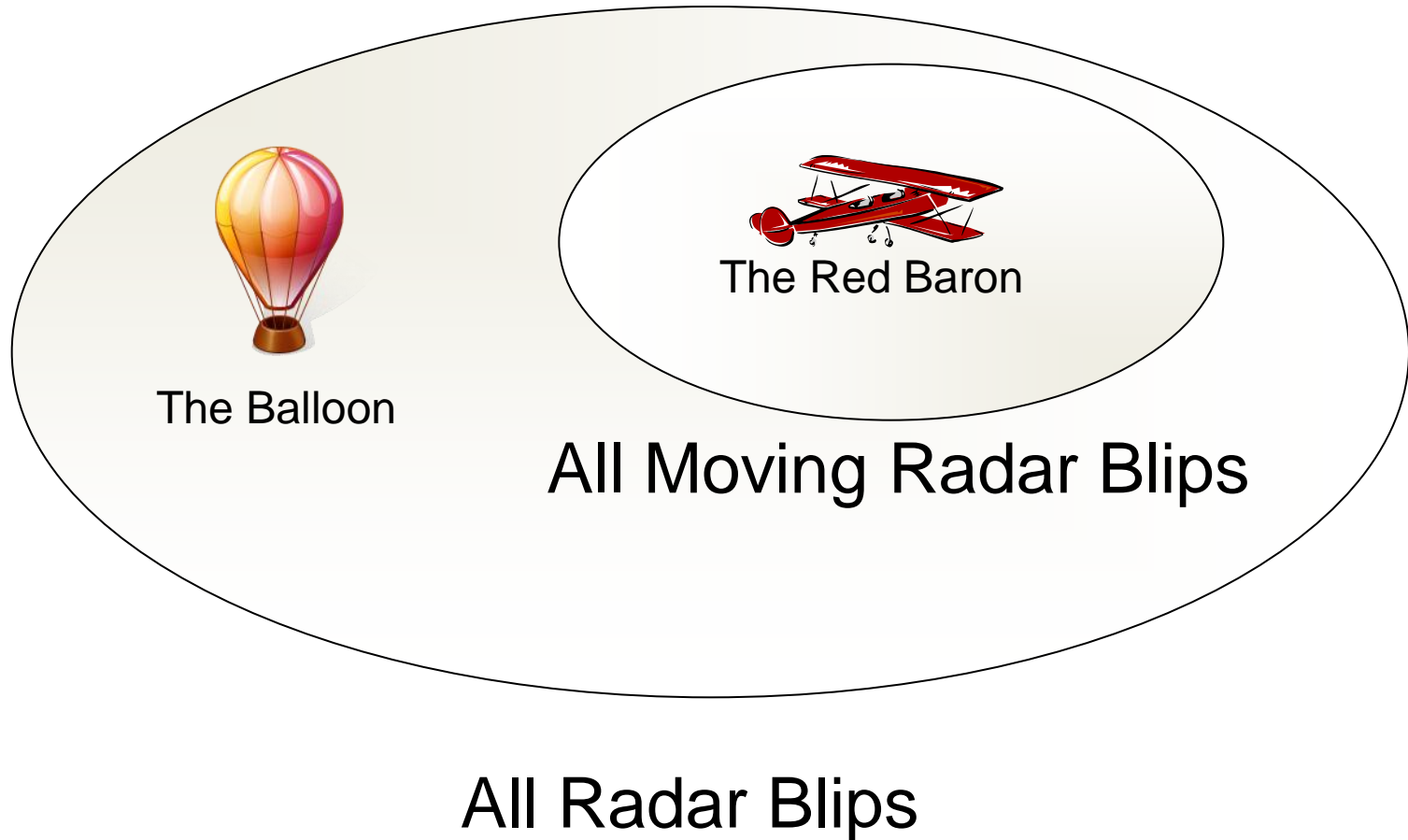
- Separates the interface from the implementation

- Simplifies object use

- Becomes a building block



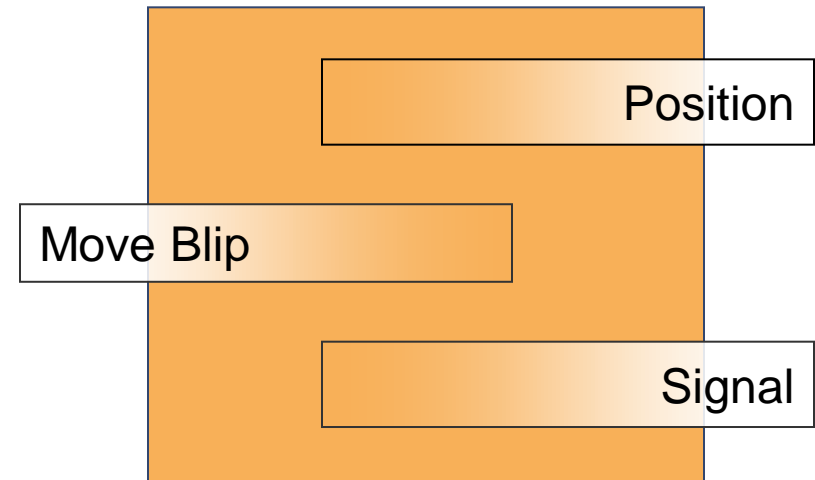
# Using a Class as a Building Block





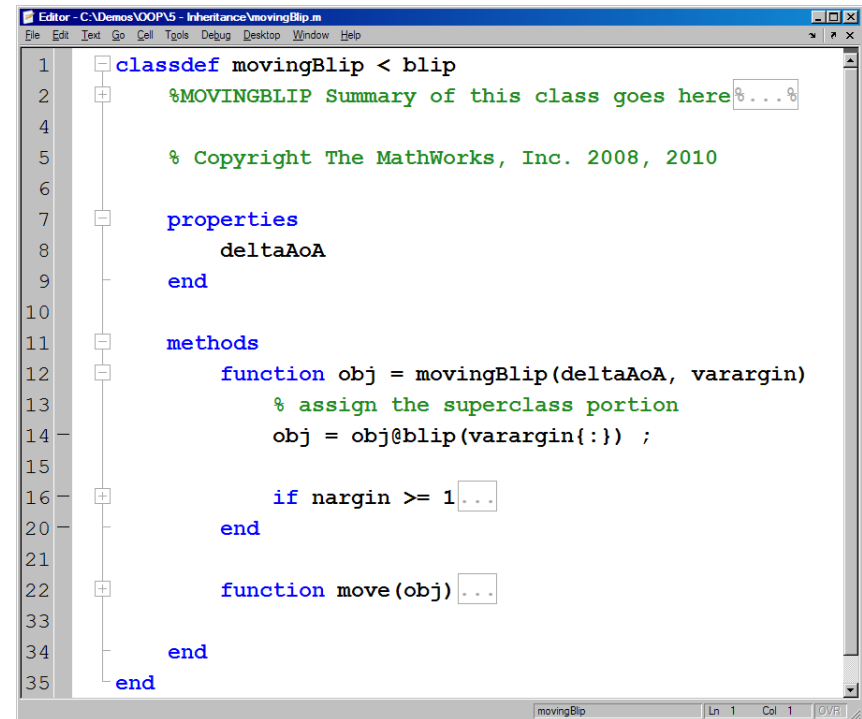
## Demonstration: Creating a Moving Target

- 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
- Allows inheriting from the base MATLAB classes



```

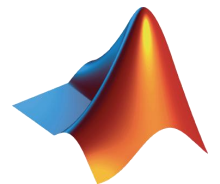
1  classdef movingBlip < blip
2      %MOVINGBLIP Summary of this class goes here ...
3
4
5      % Copyright The MathWorks, Inc. 2008, 2010
6
7      properties
8          deltaAoA
9      end
10
11     methods
12         function obj = movingBlip(deltaAoA, varargin)
13             % assign the superclass portion
14             obj = obj@blip(varargin{:}) ;
15
16             if nargin >= 1 ...
17
18             end
19
20         function move(obj) ...
21
22
23
24
25
26
27
28
29
30
31
32
33
34     end
35 end
  
```



# 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](#) | [Shopping Cart](#)

---

[Home](#) | [Select Country](#) | [Contact Us](#) | [Shopping Cart](#)

---

**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 Computing

MATLAB® is a high-level programming language and computing environment for scientific and technical computing. It combines the capabilities of traditional procedural programming with matrix-oriented computing and interactive graphics.

- Introduction
- Development
- Analysis
- Visualization
- Performance
- Publishing

**NEW** Object-Oriented Programming

**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](#)

**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.

```
classdef ANGLE
    properties
        method
    end
end
```

» [Register to view now](#)

# Questions and Answers