Introduction to Object-Oriented Programming in MATLAB®

Jos Martin
Principle Software Engineer
jos.martin@mathworks.co.uk
Goals

- Object-oriented programming
- Basic syntax in MATLAB®
- The *MATLAB* class system
**What is a program?**

<table>
<thead>
<tr>
<th>Code</th>
<th>Data</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x = 12</code>&lt;br&gt;<code>while (x &lt; 100)</code>&lt;br&gt;<code>    x = x+1</code>&lt;br&gt;<code>    if ( x == 23)</code>&lt;br&gt;<code>        disp('Hello')</code>&lt;br&gt;<code>    end</code>&lt;br&gt;<code>end</code></td>
<td><code>x = 12</code>&lt;br&gt;<code>while (x &lt; 100)</code>&lt;br&gt;<code>    x = x+1</code>&lt;br&gt;<code>    if ( x == 23)</code>&lt;br&gt;<code>        disp('Hello')</code>&lt;br&gt;<code>    end</code>&lt;br&gt;<code>end</code></td>
<td>Assignment&lt;br&gt;Looping Test&lt;br&gt;Increment&lt;br&gt;Test to Act&lt;br&gt;Take Action&lt;br&gt;End&lt;br&gt;End</td>
</tr>
</tbody>
</table>
Progression of Programming Techniques

**Data**
- literal
- variable
- structure

**Algorithm**
- command line
- script
- function

**Level of Abstraction / Sophistication**
Example: Sensor Array

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

- Easy to learn
- Minimal planning
- No formal relationship between data and functions
- Every detail is exposed
Data and Actions to Implement

**Data**
- Wavelength
- Location
- Frequency
- Number
- Spacing
- Reading

**Actions**
- Compute FFT
- Plot results
- Determine peaks
- Synthesize measurements
Related Data and Actions

Data
- Location
  - Wavelength
- Spacing
  - Reading
- Number

Actions
- Sensor
- Synthesize measurements
  - Compute FFT
  - Plot results
  - Determine peaks
Grouping Related Items

- **Target**
  - Location

- **Signal**
  - Frequency
  - Wavelength

- **Sensor**
  - Synthesize measurements
  - Determine peaks

- **Class**
  - Sensor
    - Reading
    - Spacing
    - Number

- **Actions**
  - Synthesize measurements
  - Determine peaks
  - Compute FFT
  - Plot results

MathWorks
Aerospace and Defence Conference '08
Progression of Programming Techniques

Data

- literal
- variable
- structure

Algorithm

- class
- function
- script
- command line

MathWorks
Aerospace and Defence Conference '08
Object-Oriented Terminology

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

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

- An element of the set – *object*

- Defined set – *class*
Goals

- Object-oriented programming
- Basic syntax in MATLAB®
- The MATLAB class system
Demonstration: Building a Simple Class

- Define a target class
- Create the weather balloon object
- Use the object in place of the structure
Objects

- Easy to create
- Manage their own data
- Interchangeable with a structure
  - No other code changes required
  - *Properties* behave similar to field names
  - Can’t add fields arbitrarily
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
Goals

- Object-oriented programming
- Basic syntax in MATLAB®
- The *MATLAB* class system
The MATLAB Class System

- Designed to ‘feel’ like MATLAB
  - Incorporates matrix indexing
    \[ x = 2 \times \text{obj.data}(1:end); \]
  - Inherent overloading
    \[ \text{varargout} = \text{obj.function}(	ext{varargin}) \]

- Works like an object-oriented language
  - Encapsulation, inheritance, polymorphism, etc.
Taking Methods and Properties Further

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

**External Methods**
- Plot results
- Compute AoA

**Internal Methods**
- Synthesize measurements
- Determine peaks
- Compute FFT

**External Data**
- Reading
- Spacing
- Number

**Internal Data**
- Speed of light
- Noise ratio
- etc.
Demonstration: Applying Attributes

- Control access
  Access = public
  Access = protected

- Restrict modification
  Constant
  Dependent
Encapsulation

- Sensor Reading
- Number of Towers
- Tower Spacing

Plot results
Compute AoA

Sensor
Encapsulation

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

Sensor

- Plot results
- Synthesize measurements
- Compute FFT
- Determine Peaks
- Compute AoA
- Noise Ratio
- Sensor Reading
- Number of Towers
- Tower Spacing
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 Targets

All Moving Targets

The Balloon

The Red Baron
Demonstration: Creating a Moving Target

- Define a new class `moving target`
- *Inherit* from the existing class `target`
- Add a *method*
- Use the moving target
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
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
>> B = A;
Value Class

MATLAB default

'= copies data

data in workspace

Handle Class

Use: < handle

'= references data

handle in workspace
Optional Demonstration: Using Events

- Events
  - Created in a handle object
  - `events block in classdef`
  - `notify(...) triggers event`

- Listeners
  - Triggers call back function
  - `addlistener(...)`
  - Useable anywhere
Events and Listeners

- Uses technology related to
  - preSet
  - postSet
  - preGet
  - postGet

- Gives the ability to trigger action

- Anything can listen to an observable object
The MATLAB Class System

- 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

The MathWorks®

MATLAB® - The Language of Technical Computing

MATLAB® is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.

- Introduction and Key Features
- Developing Algorithms and Applications
- Analyzing and Accessing Data
- Visualizing Data
- Performing Numeric Computation
- Publishing Results and Designing Applications

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

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

MATLAB Add-On Products

MATLAB® - The Language of Technical Computing

MATLAB® is the foundation for Simulink® and all of its add-on products designed for:
- Math and Optimization

MATLAB® Object-Oriented Programming

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 new capabilities 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.

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®

MATLAB® 7.6

Aerospace and Defence Conference '08
Questions and Answers