

MATLAB EXPO

MATLAB 프로그래밍

아키텍처 디자인

임형득 이사, 매스웍스코리아

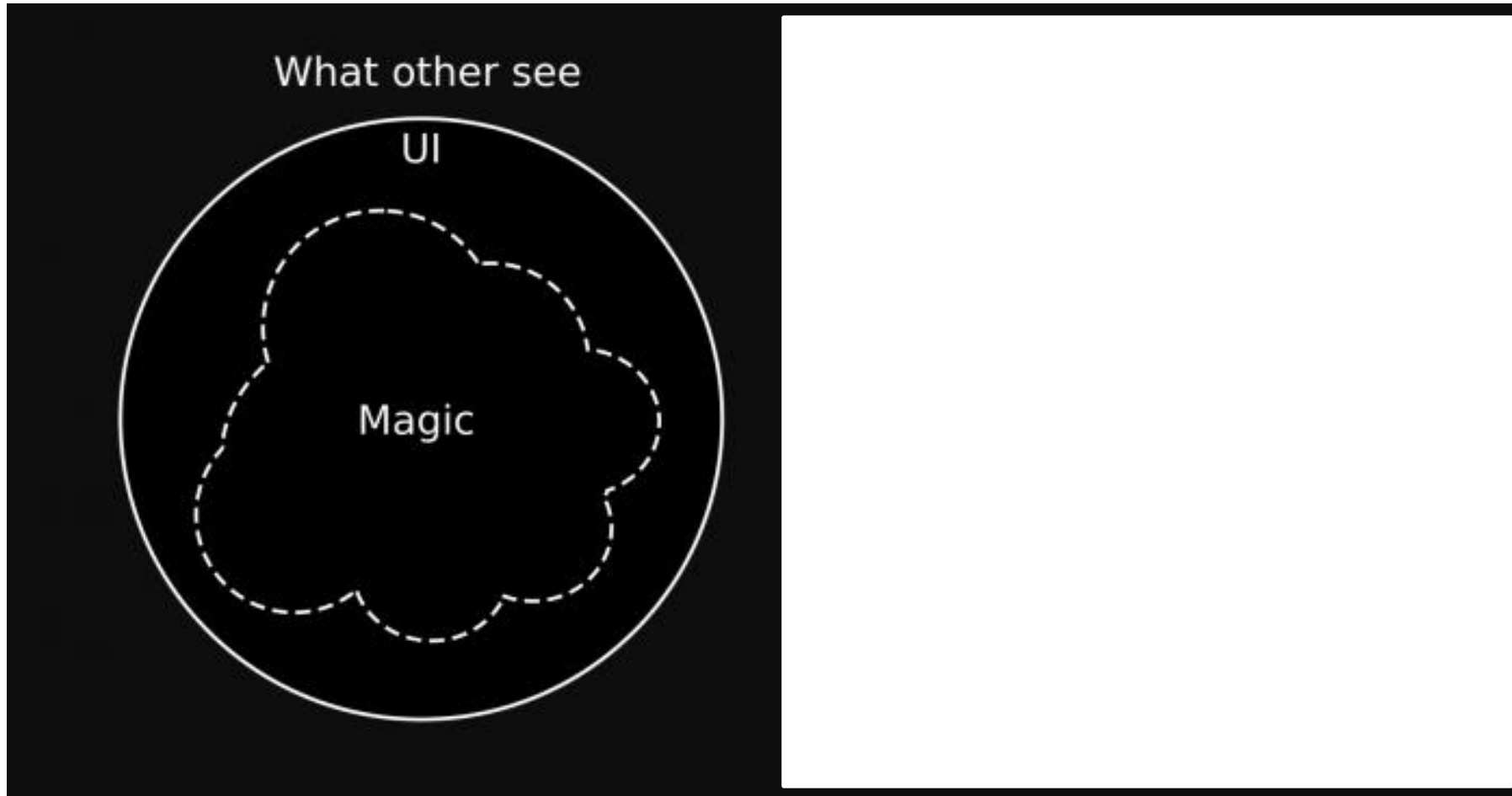


Agenda

MATLAB Programming Architecture Design

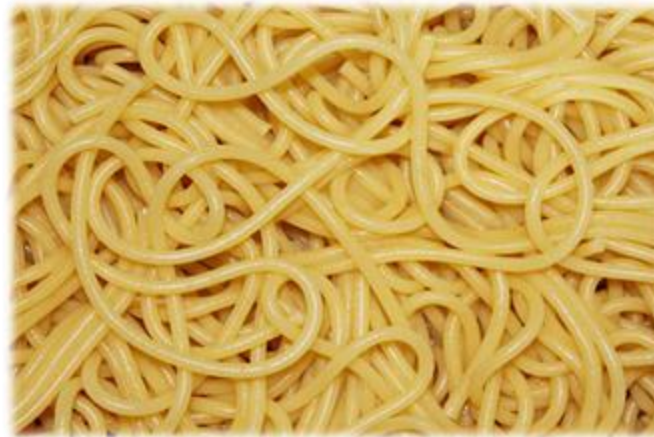
- **Software Architecture Design**
- Project Folder Design
- Class(Object Oriented Programming)
- App Architecture
- MATLAB Unit Test

Software Architecture Design

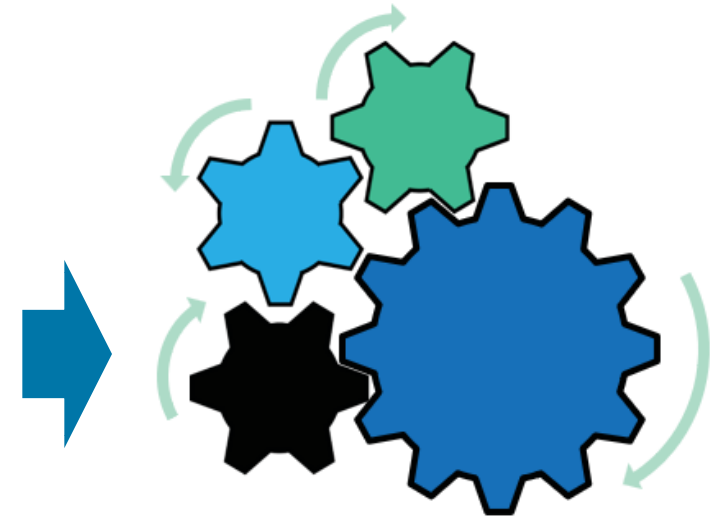


Software Architecture Design

- The Importance of Architecture Design
 - System Understanding
 - Scalability and Flexibility
 - Risk Mitigation
 - Modularity and Reusability
 - Collaboration and Communication
 - Quality Assurance
 - Cost and Time Efficiency



DISARRAY



ORGANIZATION

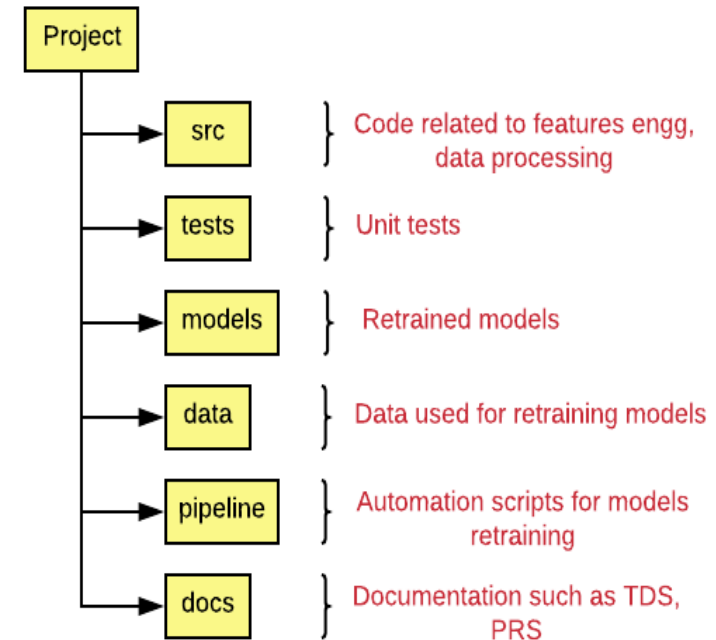
Agenda

MATLAB Programming Architecture Design

- Architecture Design
- **Project Folder Design**
- Class(Object Oriented Programming)
- App Architecture
- MATLAB Unit Test

Project Folder Design

- Folder Structure Design
 - Project Name
 - Source Code
 - Models
 - Data
 - Test
 - Document



Data Science Project Folder Structure

<https://vitalflux.com/data-science-project-folder-structure/>

Project Folder Design

- MATLAB Project
 - Automate Tasks
 - Path setup and startup/shutdown
 - Shortcut
 - Collecting Metadata
 - Labels, Grouping(Classification)
 - Source Control(Management)
 - Gitlab or SVN
 - Check in/ Check out
 - Track and Compare revisions
 - Analyze dependencies
 - Sharing Code
 - Package and share projects

The screenshot shows the MATLAB Project Explorer interface for a project named 'Project (226)'. The interface is divided into three main sections: TOOLS, ENVIRONMENT, and SOURCE CONTROL. The TOOLS section includes Search, Custom Tasks, Run Checks, References, and Details. The ENVIRONMENT section includes Project Path, Startup Shutdown, and Git Details. The SOURCE CONTROL section includes Refresh, Commit, Push, Pull, Fetch, Remote, and Branches. Below these sections is a table listing the project's contents, including folders and files, with columns for Name, Status, Git, and Classification.

Name	Status	Git	Classification
+Test	✓	■	Test
ACI	✓	·	
Dashboard	✓	·	
Documents	✓	·	
Elasticsearch	✓	·	
MachineLearning	✓	■	
MATLAB_Kafka_Producer_Java	✓	·	
mps_stream	✓	■	
SimExecutable	✓	·	
Simulation	✓	·	
DocExample_MultiClassFaultDetectionUsi...	✓	●	Design
genPumpData.m	✓	●	Design
javasetup.m	✓	+	Design
Main_ExampleWorkflow.mlx	✓	●	Design
MLModels.mat	✓	●	Design
rawdata.mat	✓	●	Design
README.md	✓	●	

Project Folder Design

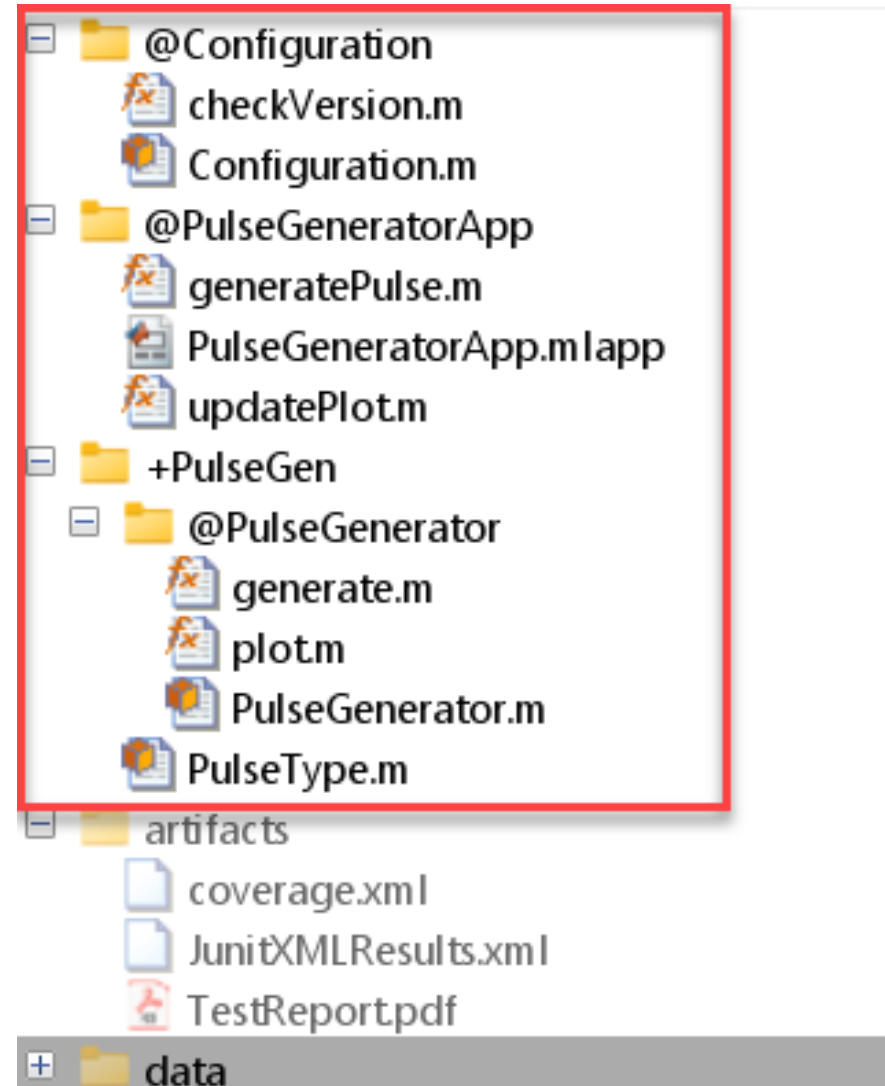
- Packages Namespaces(+folders)
 - Code Organization
 - Encapsulation and Modularity
 - Code Discoverability
 - Avoiding Naming Conflicts
 - Names must be unique
 - Contains class folders, function, and other packages
 - Top-level package folder must be on the MATLAB path

The screenshot displays the MATLAB IDE interface. On the left, the 'Current Folder' browser shows a hierarchical project structure. The '+GUI' folder is expanded, listing various MATLAB files such as 'ChooseDialog.m', 'CustomInputDialog.m', 'DisplayError.m', 'DisplayFeature.m', 'DisplayRegression.m', 'ExceptionDialog.m', 'FigureConfusionMat.m', 'FigureParallelCoords.m', 'FigureROC.m', 'GetFile.m', 'GetFilterParameter.m', 'GetFolder.m', 'GetInput.m', 'GetMATFile.m', 'GetSignalDataFolder.m', 'GetSignalParameter.m', 'QuestionDialog.m', 'SaveMATFile.m', and 'SaveMATLABFile.m'. The '+Excel' folder is also visible, containing sub-folders like '@ActxServ', '@ExcelDoc', '@XIColors', '@XICopyPictureFormat', '@XIFileFormat', '@XIFontStyle', '@XIPasteType', '@XIPictureAppearance', '@XISheetVisibility', and '@XIWindowState'. The 'Command Window' on the right shows the following MATLAB code:

```
>> Pack1.Pack1_1.Get
Pack1.Pack1_1.Get()
>> Pack1.Pack1_1.Set()
Pack1.Pack1_1.Set()
>> Pack2.Get()
Pack2.Get()
>> Pack2.Set()
pack2.set()
>> Get()
Get() function in MATLAB Path
>> Set()
Set() function in MATLAB Path
fx >>
```


Project Folder Design

- Class (@folders)
 - Code organization
 - Encapsulation Modularity
 - Code sharing and collaboration
 - Avoiding Naming Conflicts
 - Encourages best practices
 - Contains class folders, function, and other packages.
 - Top-level package folder must be on the MATLAB path.

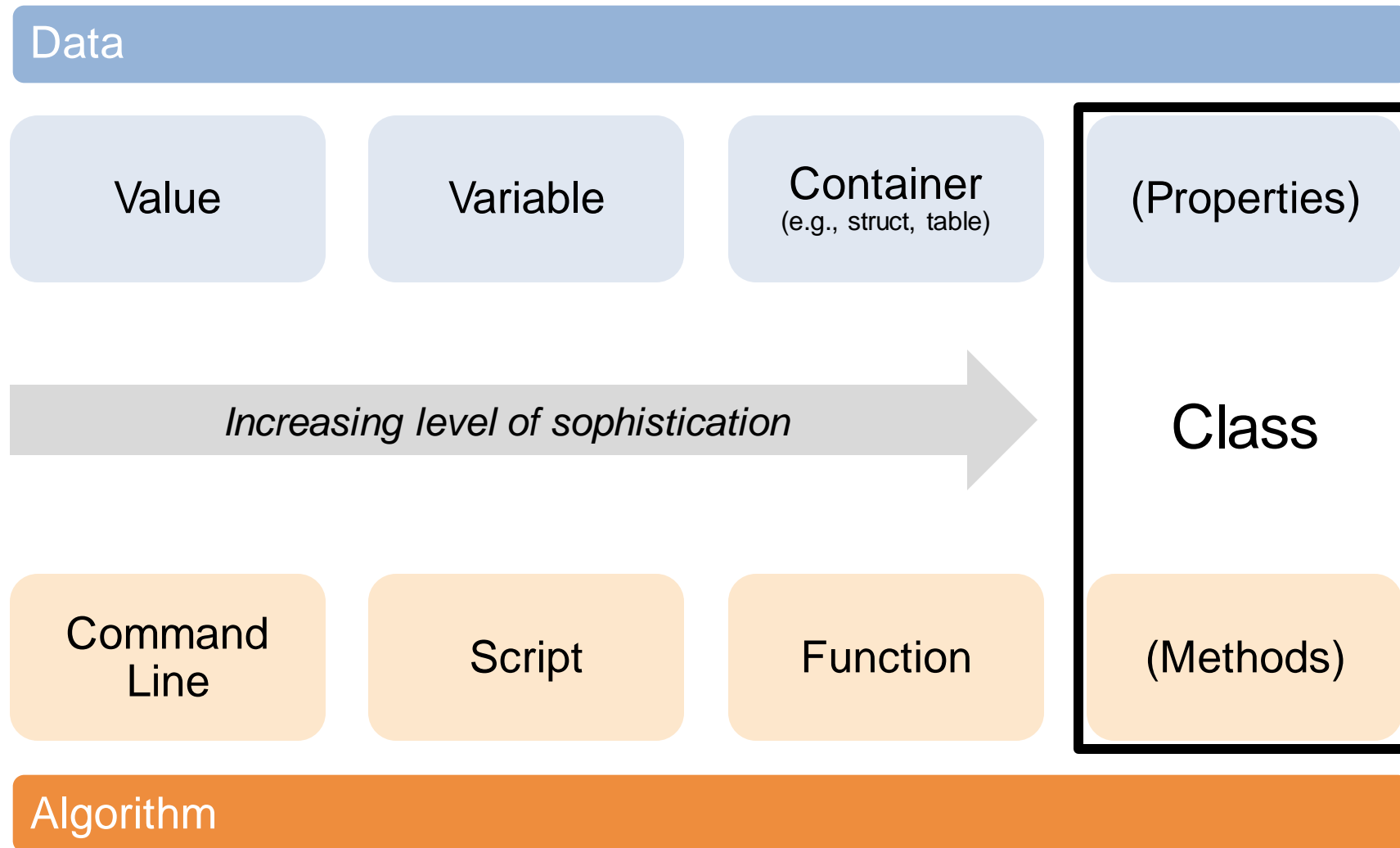


Agenda

MATLAB Programming Architecture Design

- Architecture Design
- Project Folder Design
- **Class(Object Oriented Programming)**
- App Architecture
- MATLAB Unit Test

Class(Object Oriented Programming)



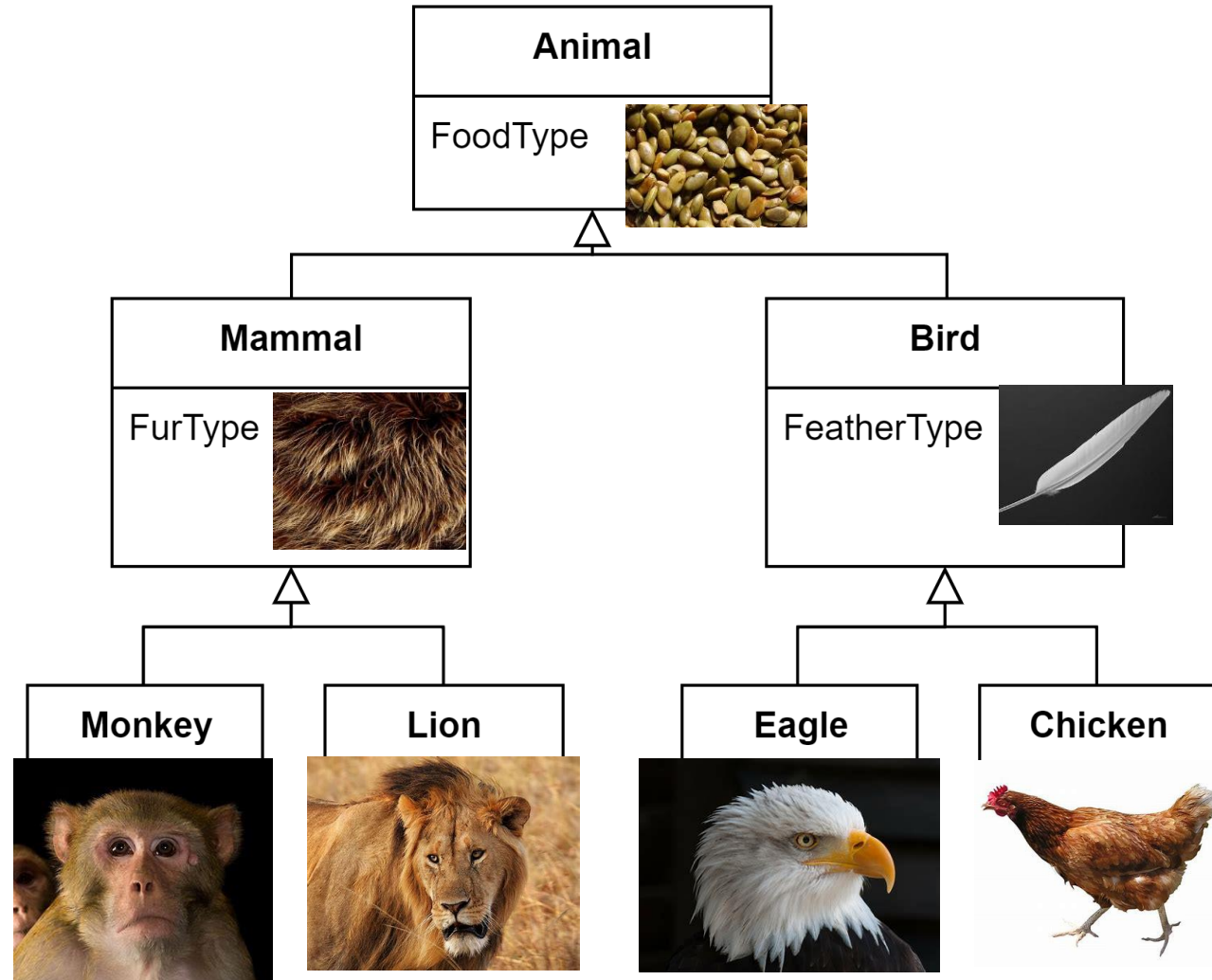
Class(Object Oriented Programming)

- Class
 - A **blueprint** for creating objects; a concept
 - Properties (data, state)
 - Methods (algorithms, behavior)
- Object
 - A specific instance of a class

```
Class: Dog
```

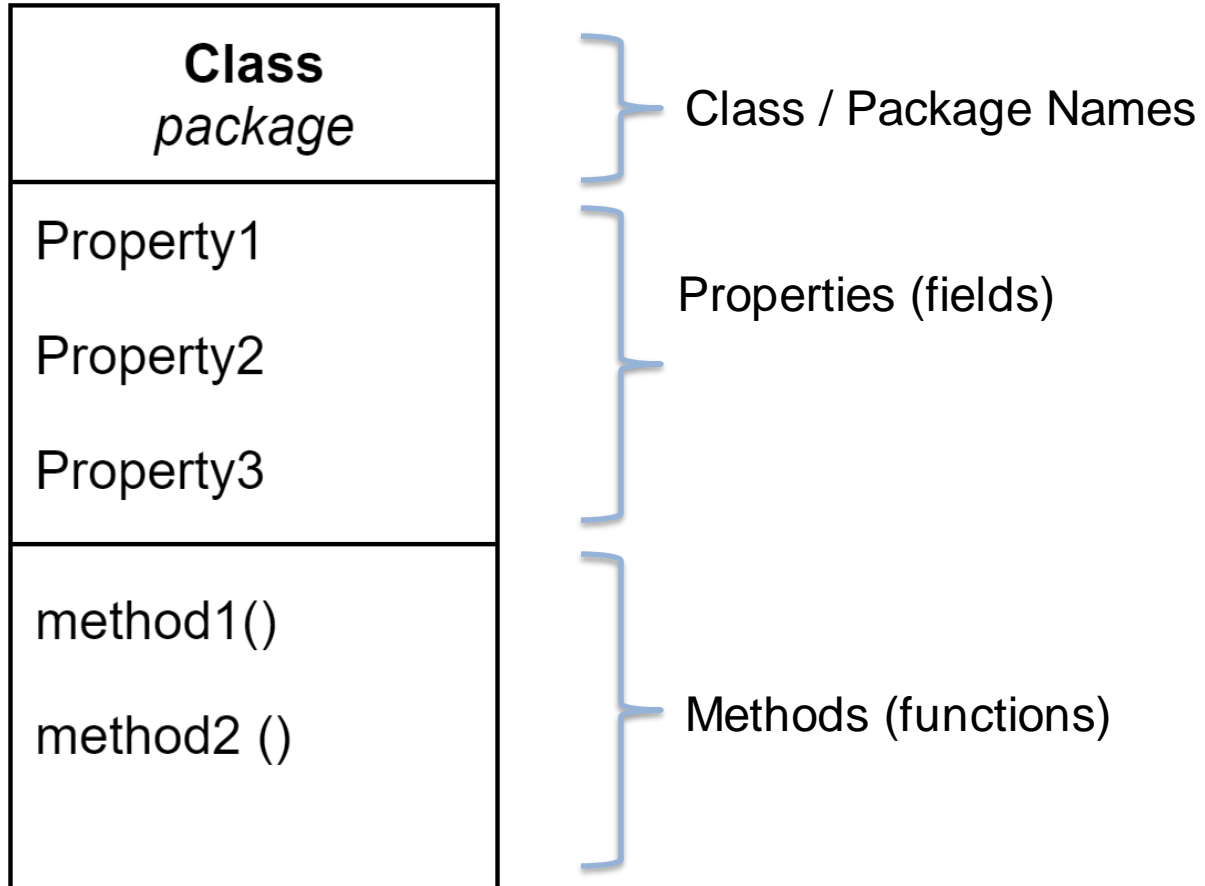
Class(Object Oriented Programming)

- Inheritance



→
Inheritance ("is a")

Class(Object Oriented Programming)



Class Diagrams: Class Block

```

HandleClass.m x +
20 %% Class Prototype
21 classdef HandleClass < handle & matlab.mixin.Copyable & dynamicprops & matlab.mixin.SetGet
22 %% Constants
23 properties( GetAccess = public , Constant = true)
24 end
25
26 %%Read-only properties
27 properties( GetAccess = public,SetAccess = protected )
28 end
29 %%Write-only properties
30 properties( GetAccess = protected,SetAccess = public )
31 end
32
33 %%Read/Write properties
34 properties( Access = public )
35     Color    (1,:) char    {mustBeText}    = 'Red';
36     State    (1,1) double  {mustBeNumeric}  = 100;
37 end
38
39 events
40     ToggledState;
41 end
42
43 methods( Access = public )
44     %%Constructor
45     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
46     function obj = HandleClass(c)
47         if nargin > 0
48             obj.Color = c;
49         end
50     end
51     %%Destructor
52     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53     delete(obj);
54 end
55
56 %%Public Methods
57 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
58 methods( Access = public)
59     ChangeColor(obj,c);
60     OnStateChange(obj,newState);
61 end
62
63 %%Private Methods
64 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
65 methods( Access = private )
66
67 end
68
69 %%Protected Methods
70 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
71 methods (Access = protected)
72
73 end
74
  
```

Class(Object Oriented Programming)

- Handle Class

- Instance variables refer to objects
- A copy of an instance variable **refers** to the same object as the original variable
- Good for representing physical entities (people, places, things)

```
classdef MyHandleClass < handle  
    ...  
end
```

- Value Class

- The data of an instance is independent of the data in the **copy** of that instance
- Good for representing mathematical abstractions such as double arrays or symbolic arrays

Class(Object Oriented Programming)

- OOP Design Patterns
 - A known good solution to a standard problem
 - Allows reuse
 - Easy for reference
- Adapter
 - Integrate objects that have different interfaces
- Singleton
 - Global object
- Factory
 - create complex objects more easily

Class(Object Oriented Programming)

```

Project - 01_ValueClass
ValueClass.m
1 % BasicClass - MATLAB Basic Class Define Example
2 % -----
3 % Abstract: MATLAB Basic Class Define Example
4 %
5 %
6 % Syntax:
7 %     [hBas]=BasicClass()
8 %
9 %
10 % Examples:
11 %     [hBas]=BasicClass;
12 %
13 %
14 % Notes: none
15 % Copyright
16 % -----
17 %     Copyright 2022 Consulting Services, The MathWorks, Inc.
18 % -----
19
20 %% Class Prototype
21 classdef ValueClass
22     %%Constants
23     properties( GetAccess = public , Constant = true)
24     end
25
26     %%Read-only properties
27     properties( GetAccess = public,SetAccess = protected )
28     end
29     %%Write-only properties
30     properties( GetAccess = protected,SetAccess = public )
31     end
32
33     %%Read/Write properties

```

Command Window

New to MATLAB? See resources for [Getting Started](#).

fx >>

```

HandleClass.m
1 % HandleClass - HandleClassDefine Example
2 % -----
3 % Abstract: HandleClass Define Example
4 %
5 %
6 % Syntax:
7 %     [handle]=HandleClass()
8 %
9 %
10 % Examples:
11 %     [handle]=HandleClass;
12 %
13 %
14 % Notes: none
15 % Copyright
16 % -----
17 %     Copyright 2022 Consulting Services, The MathWorks, Inc.
18 % -----
19
20 %% Class Prototype
21 classdef HandleClass < handle & matlab.mixin.Copyable & dynamicprops & matlab.mixin.SetGet
22     %%Constants
23     properties( GetAccess = public , Constant = true)
24     end
25
26     %%Read-only properties
27     properties( GetAccess = public,SetAccess = protected )
28     end
29     %%Write-only properties
30     properties( GetAccess = protected,SetAccess = public )
31     end
32
33     %%Read/Write properties

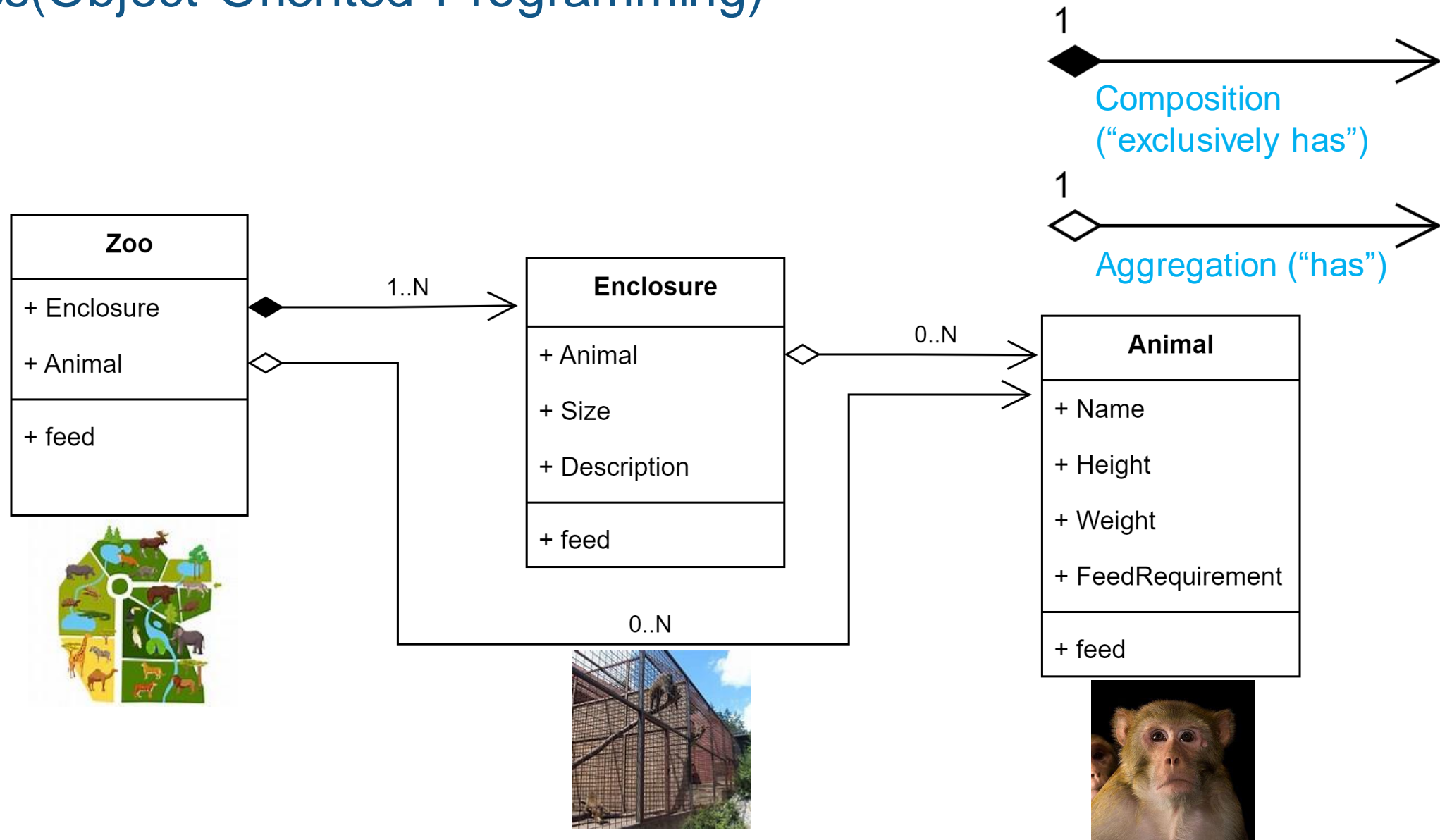
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

fx >>

Class(Object Oriented Programming)

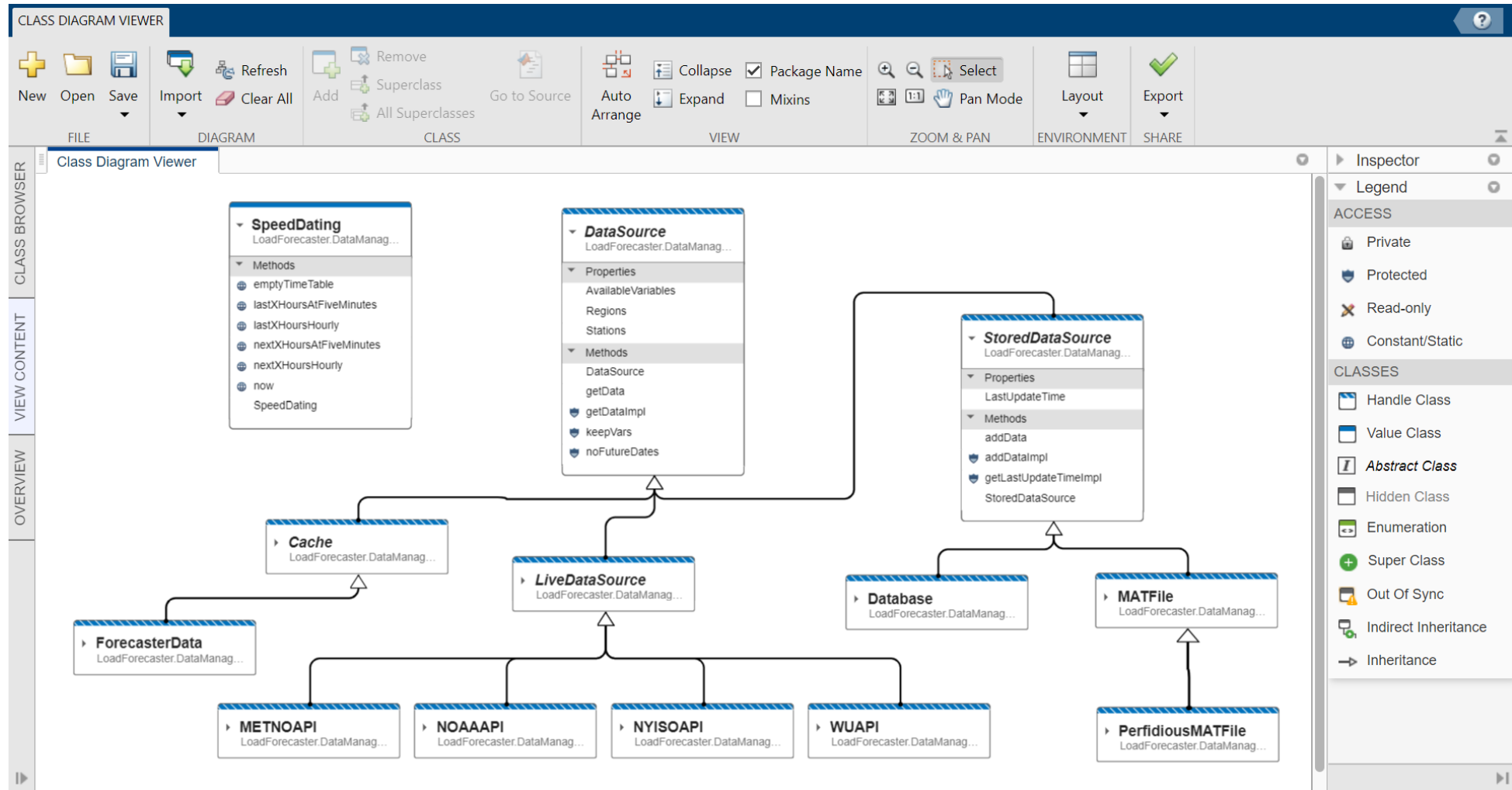


Class(Object Oriented Programming)

- OOP Design Patterns
 - A known good solution to a standard problem
 - Allows reuse
 - Easy for reference
- Adapter
 - Integrate objects that have different interfaces
- Singleton
 - Global object
- Factory
 - create complex objects more easily

Class(Object Oriented Programming)

- MATLAB's Class Diagram Viewer

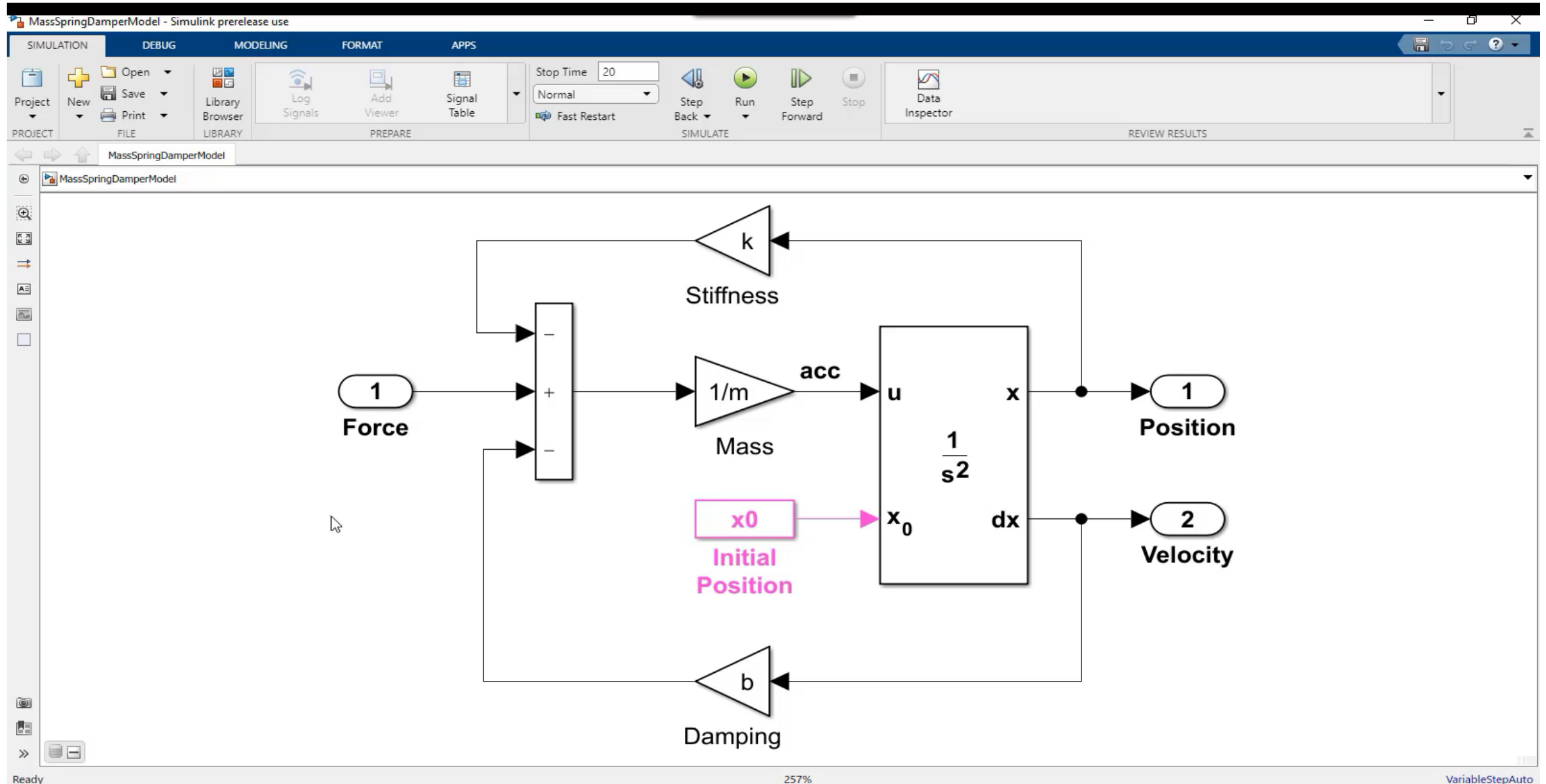


Agenda

MATLAB Programming Architecture Design

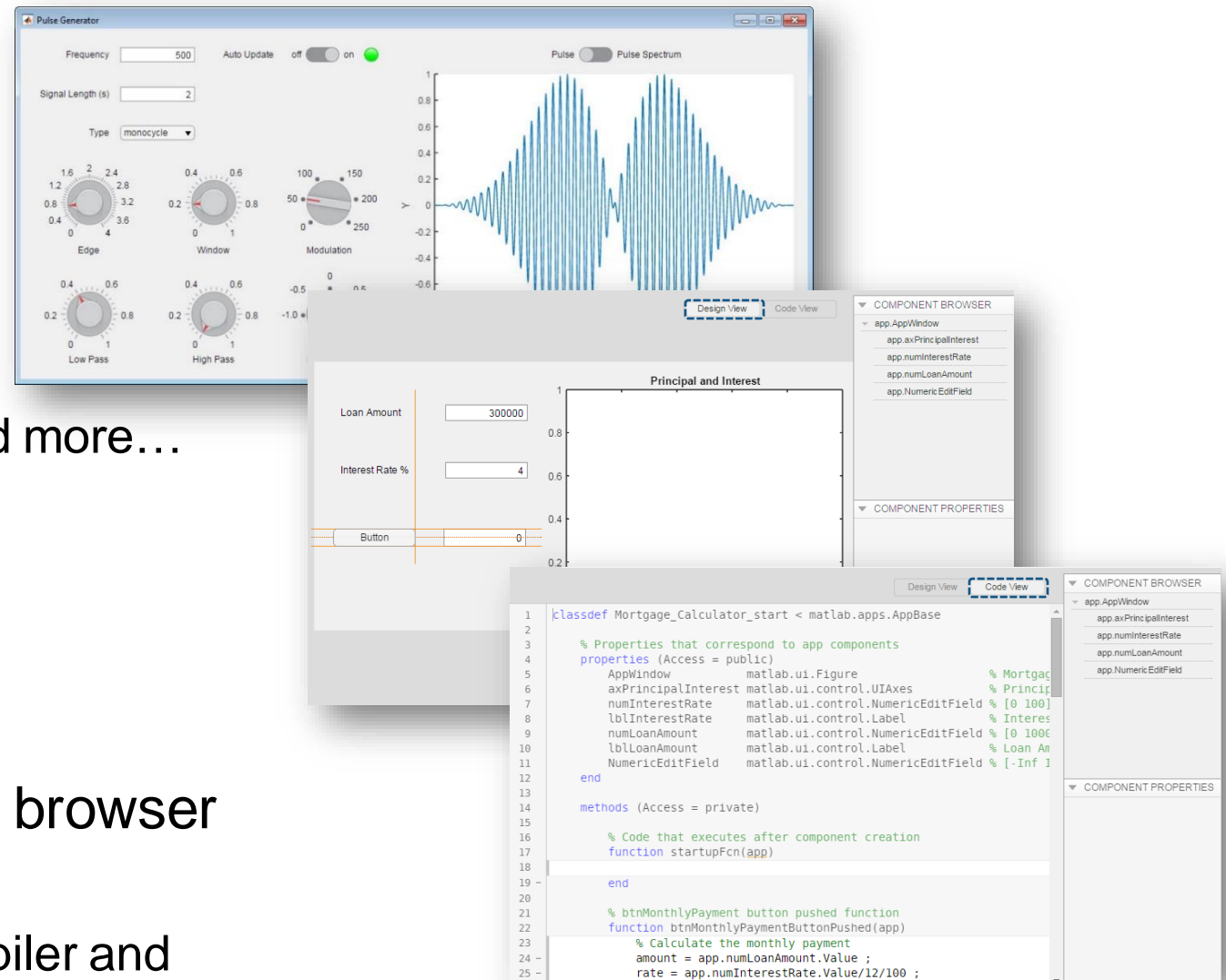
- Architecture Design
- Project Folder Design
- Class(Object Oriented Programming)
- **App Architecture**
- MATLAB Unit Test

App Architecture



App Architecture

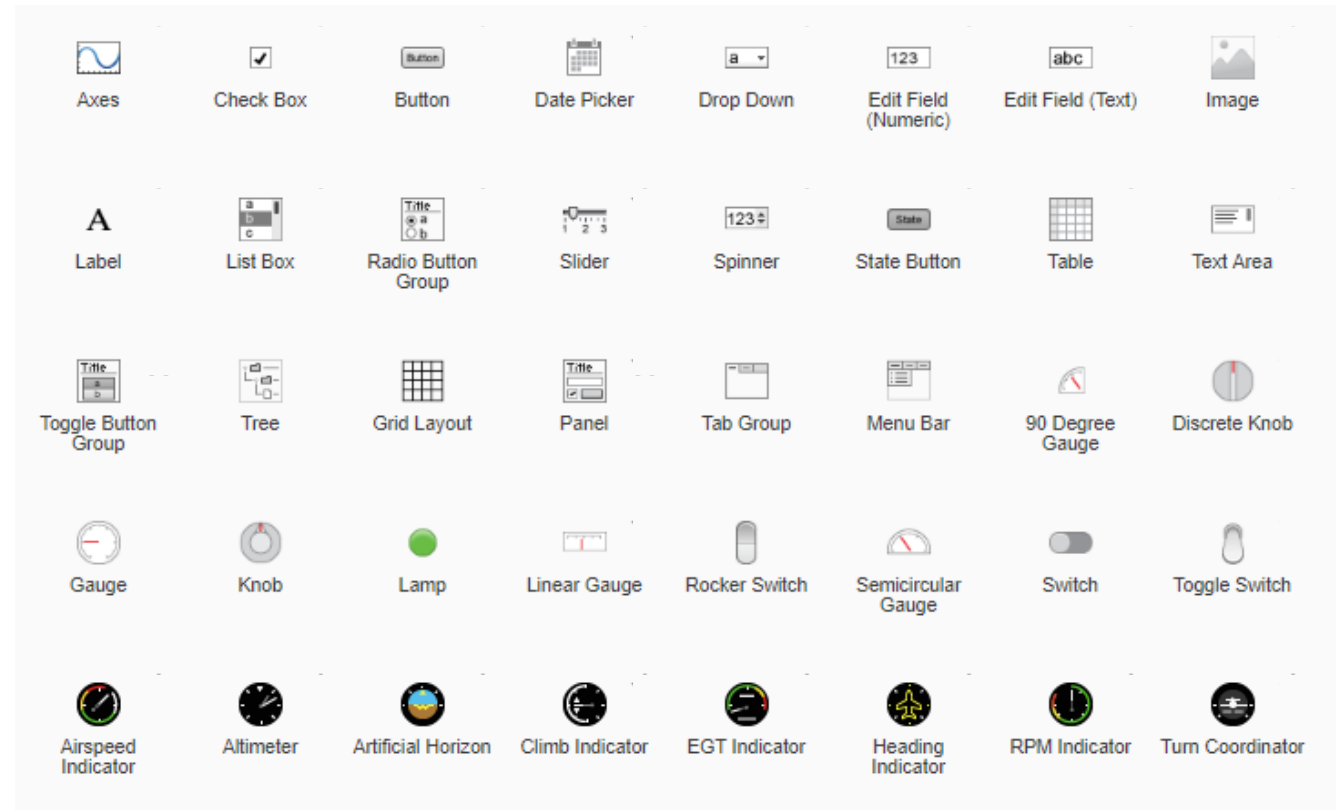
- Enhanced design environment
 - Component alignment guides
 - Simpler property inspectors
 - Intuitive menu bar interface
- Expanded UI component set
 - Gauges, dials, tabs, date picker, and more...
- Improved code and coding tools
 - Object-based code format
 - Property and method management
 - Code refactoring
- Run App Designer apps in a web browser
 - Run apps in MATLAB Online
 - Package apps using MATLAB Compiler and host them using MATLAB Web App Server



The App Designer

App Architecture

Components

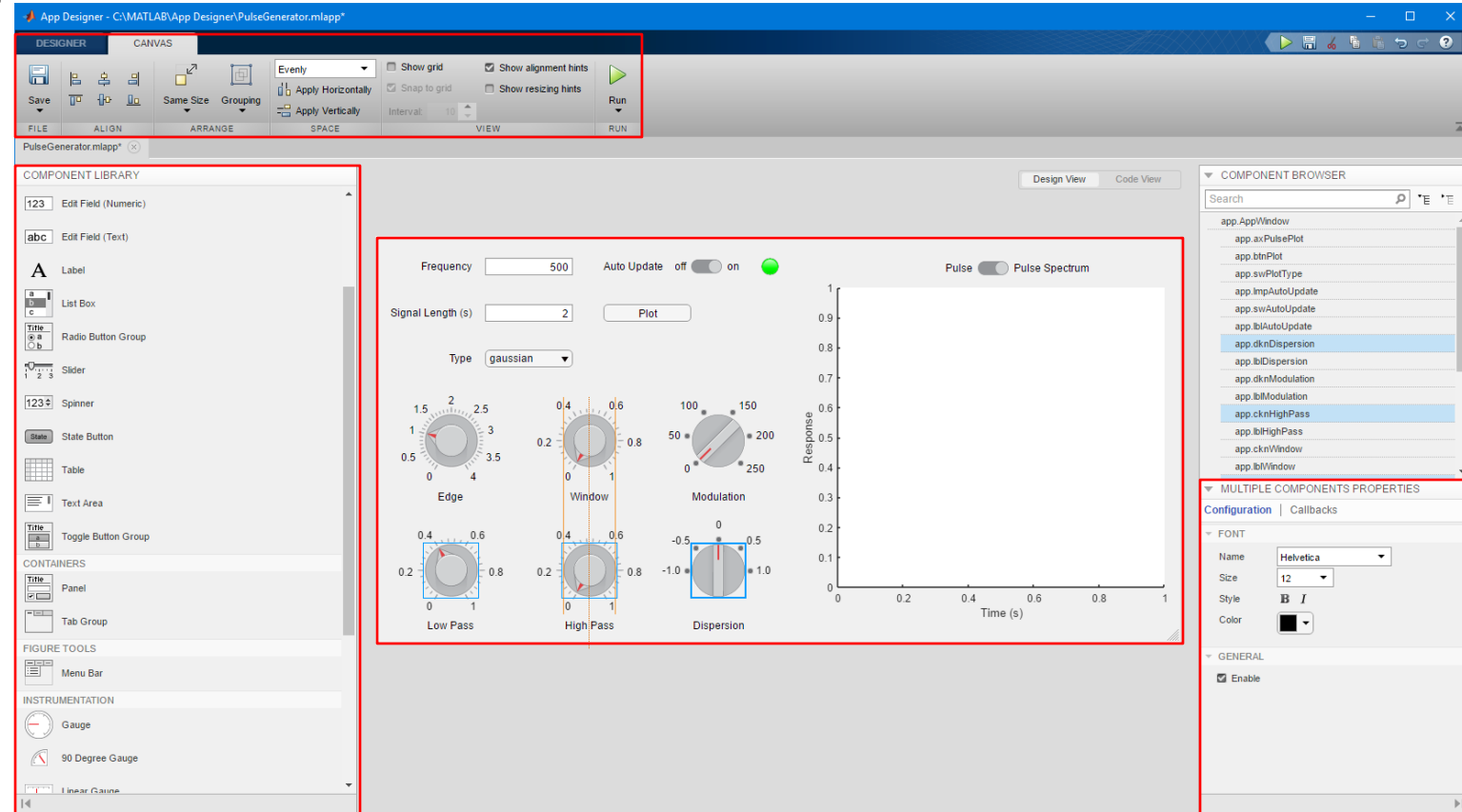


UI Components

App Architecture

Design View

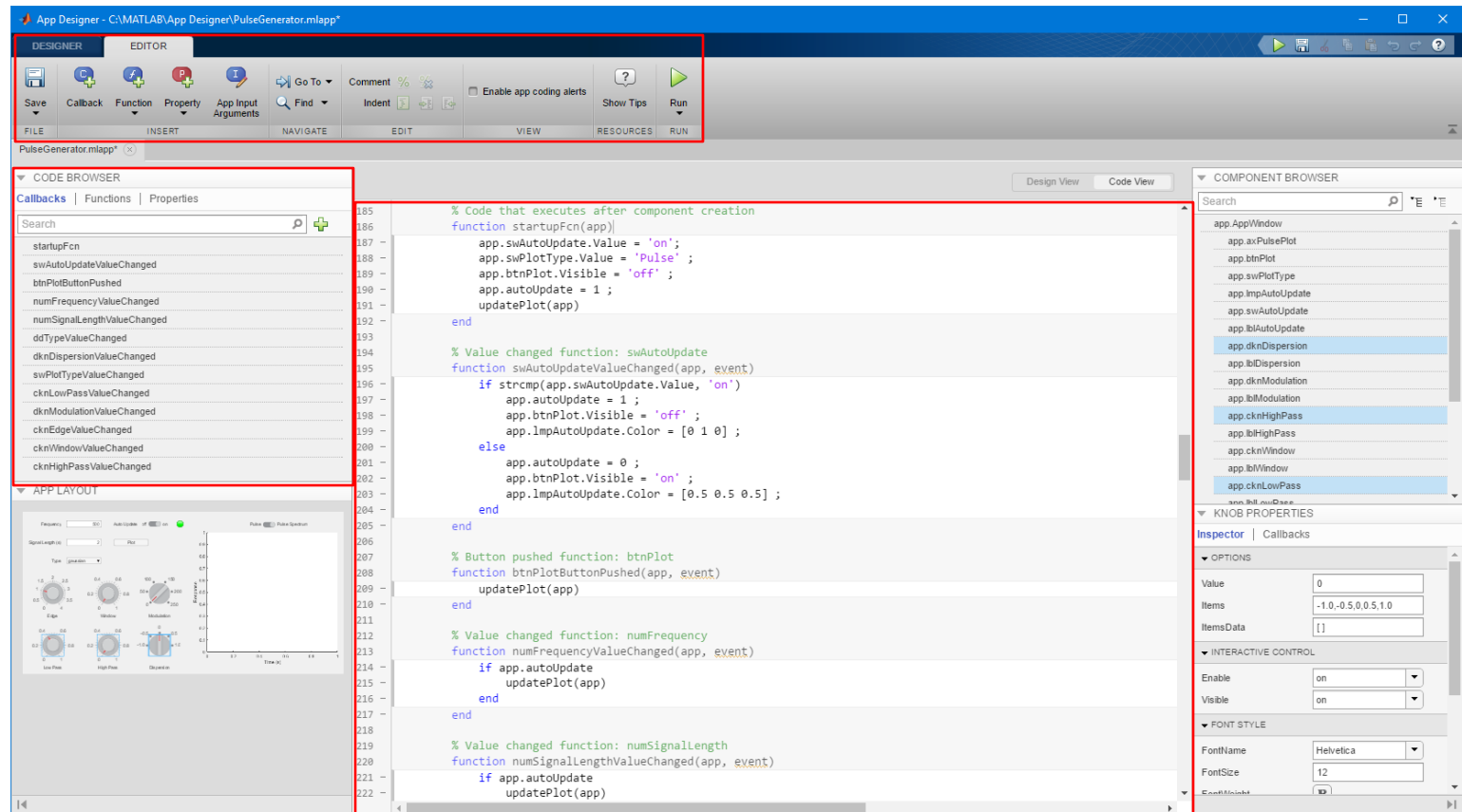
- Design and layout the app's interface
- Component Library
 - Select components and add them to the canvas
- Design Canvas
 - Layout components
- Toolstrip
 - Align, space, and group components
- Properties panel
 - Set common component properties



App Architecture

Code View

- Write code to control the app's behavior
- Editor
 - Write code for callbacks and other functions
- Code Browser
 - Navigate to callbacks and app properties
- Toolstrip
 - Add new code elements properties, callbacks, and functions



App Architecture

▪ App Architectures

– Maintenance / Traceability / Readability

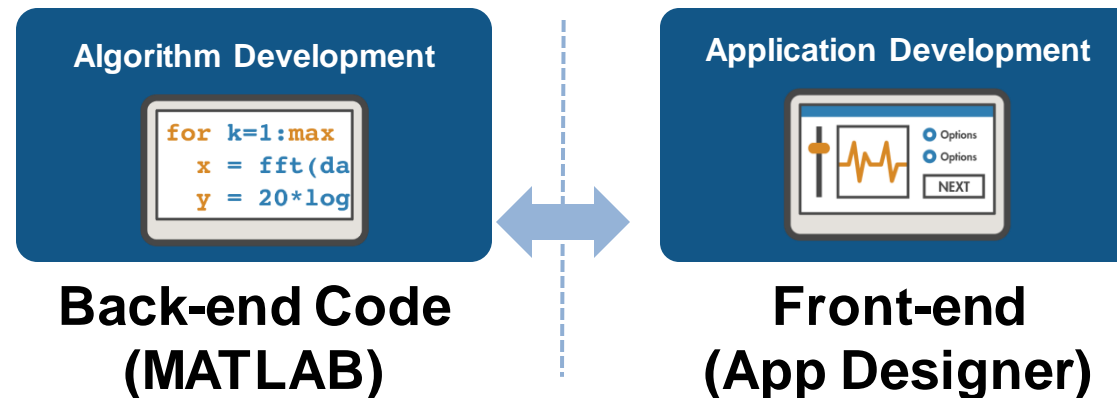
- Back-end Architectures : Algorithms, Methods(Functions), Properties(Data)
- Front-end Architectures : GUI(App Designer)

– Stability / Robust

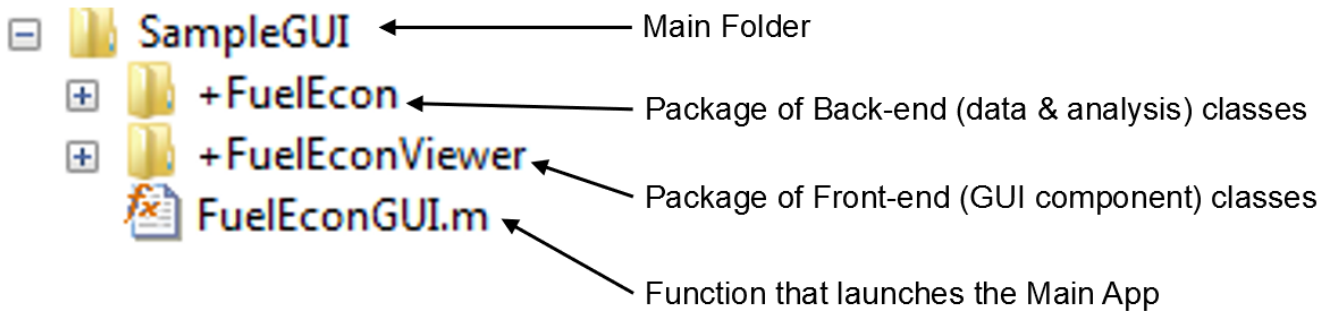
- Unit Test : Script ,Function ,Class , App Unit Test

– Reusability

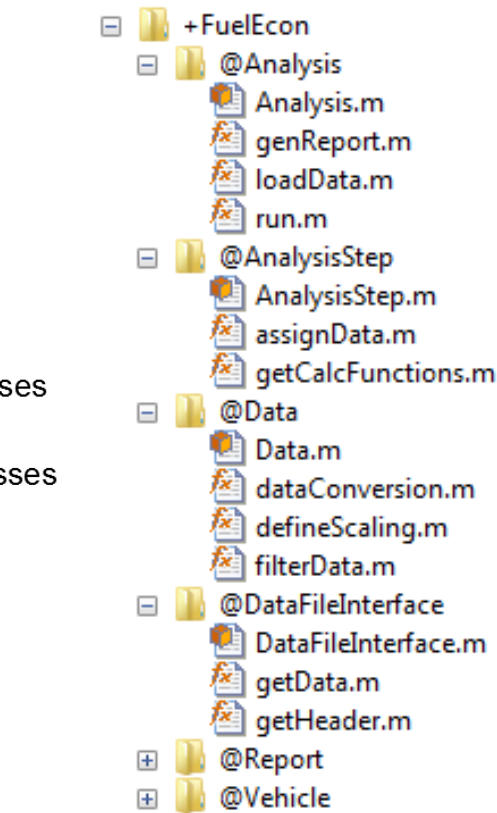
- Class, OOP(Object-Oriented-Programming)
- OOP Pattern Design



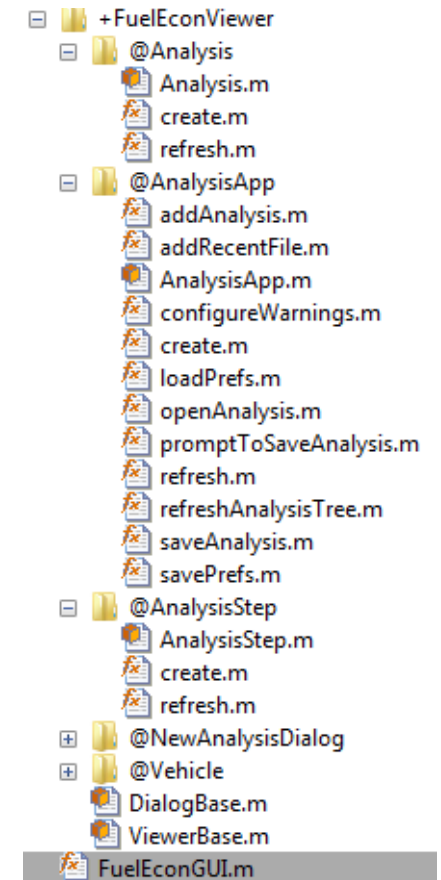
App Architecture



App Architecture



Back-end Package & Class






Front-end Package & Class

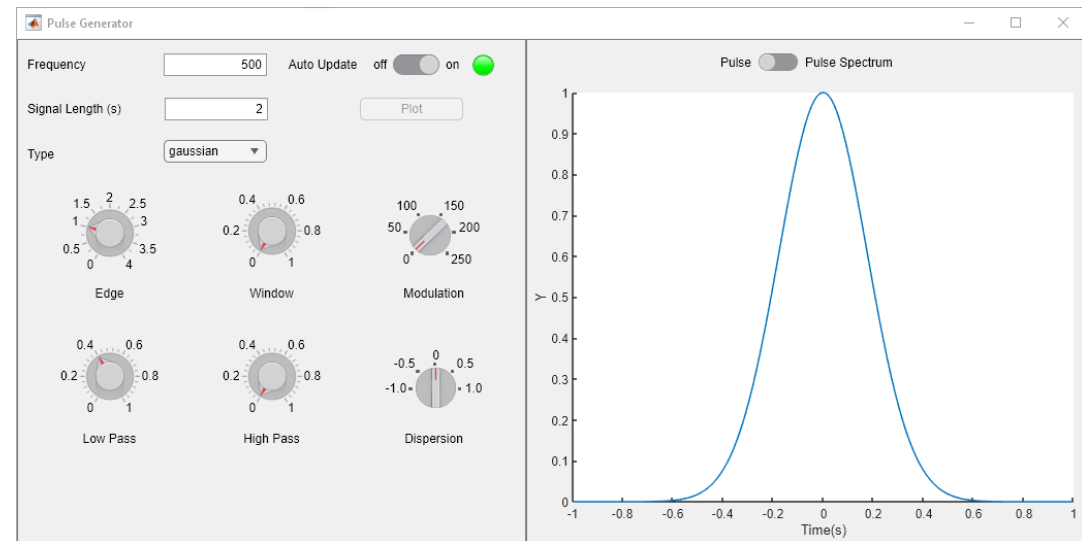
App Architecture

- Pulse Generator App

https://www.mathworks.com/help/matlab/creating_guis/app-or-gui-with-instrument-controls.html

Name ▲
 pulsegen_screenshot.png
 PulseGenerator.mlapp
 PulseGeneratorAppExample.m

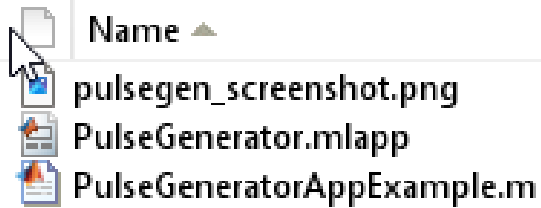
PulseGenerator.mlapp



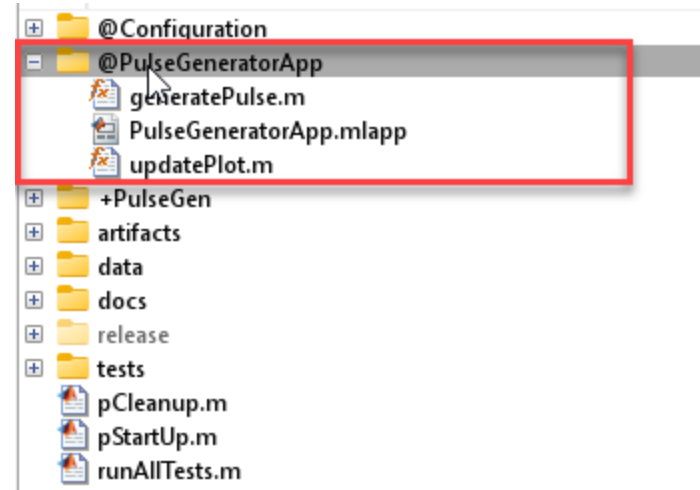
Pulse Generator App

App Architecture

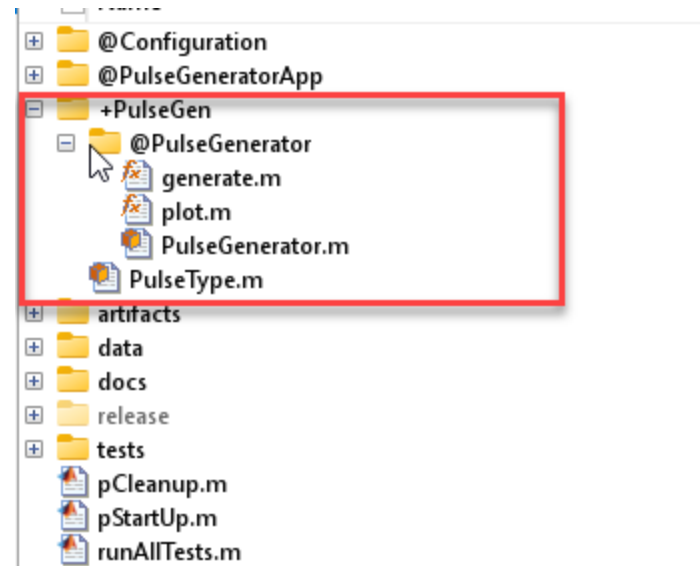
- Back-end / Front-end Architectures



PulseGenerator.mlapp



Front-end App Class



Back-end Package

App Architecture

The screenshot displays the MATLAB R2022b interface with the App Architecture tool open. The current folder is `C:\01_Denny_Working\02_Consulting\02_GIT\01_InternalGitLab\16_2023_sko_refactoring\02_Refactoring_Process`. The workspace is empty, and the Command Window shows the prompt `f>>`.

The Current Folder pane shows the following files and folders:

Name	Git
@Configuration	.
@PulseGeneratorApp	.
+PulseGen	.
artifacts	.
data	.
docs	.
release	.
resources	.
tests	.
.gitattributes	●
.gitignore	●
.gitlab-ci.yml	●
pCleanup.m	●
pStartUp.m	●
publishDocument.m	●
PulseGen.prj	●
runAllTests.m	●

The Command History pane shows the following commands:

```
doc Pu...
%-- 20...
runAll...
%-- 20...
f_FunC...
load('...
f_FunC...
load('...
untitled
untitled
untitled
untit...
untit...
untit...
```

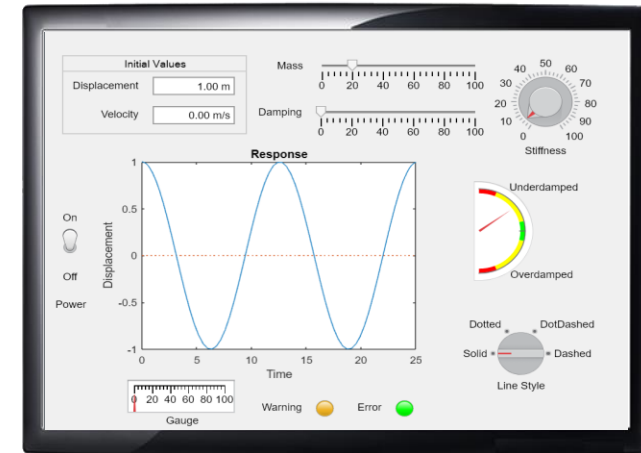
App Architecture

App Deployment

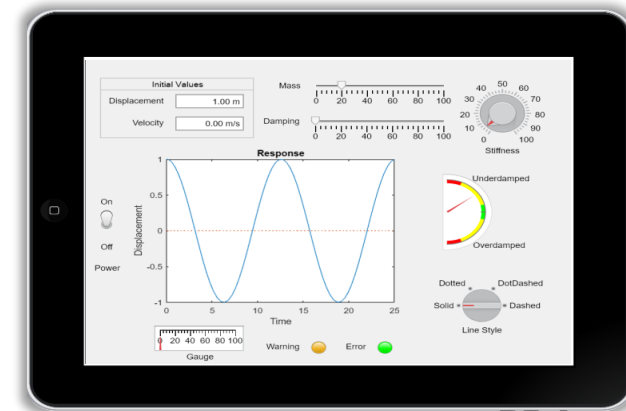


Stand alone

Web deploy



app.exe



<https://www.app.com>

Agenda

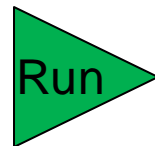
MATLAB Programming Architecture Design

- Architecture Design
- Project Folder Design
- Class(Object Oriented Programming)
- App Architecture
- **MATLAB Unit Test**

WHY?



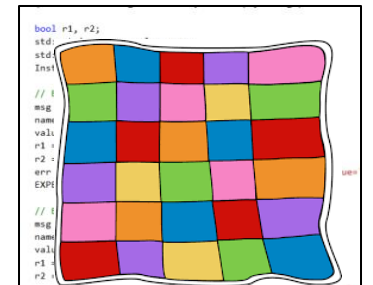
Testable Code?



Running Tests

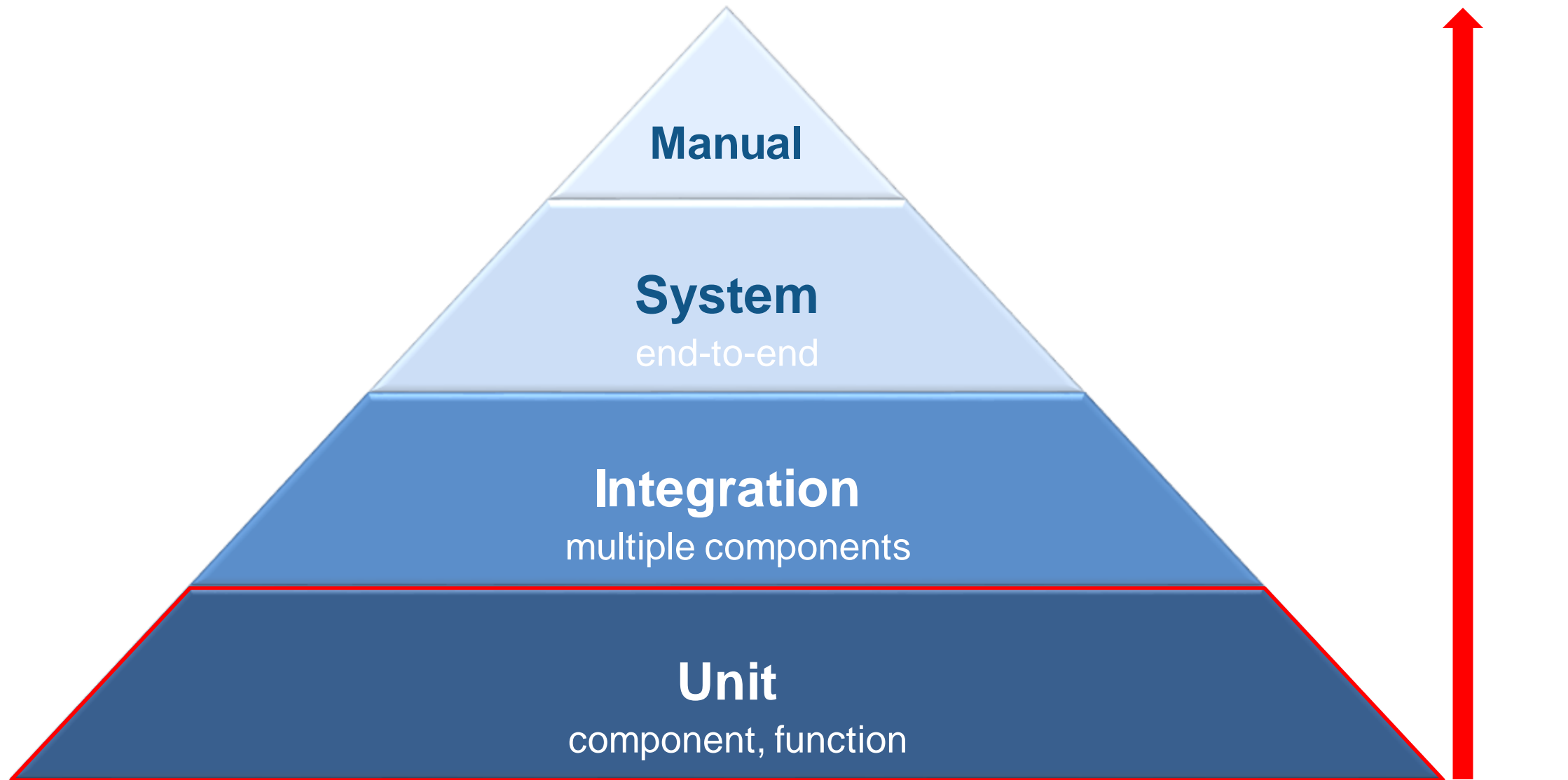


Debugging



Code Coverage

MATLAB Unit Test



Unit Testing Pyramid

MATLAB Unit Test



2 unit tests, 0 system tests...

MATLAB Unit Test

- MATLAB Unit Testing Framework
 - Script-based Unit Tests
 - Function-Based Unit Tests
 - Class-Based Unit Tests
 - App-Based Unit Tests

- Use the **TestCase** class template to create tests more quickly and accurately

- Works with continuous integration servers



- Code coverage metrics (statement and function coverage) and report format

MATLAB® Code Coverage Report

The MATLAB code coverage report provides a detailed analysis of the source code covered by the tests.

Overall Coverage Summary

Summary of the code coverage metrics for all source files.

Coverage Metric	Executable	Code Coverage
Statement Coverage	196	 44.89%
Function Coverage	36	 58.33%

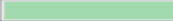
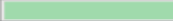
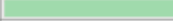

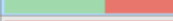
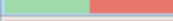






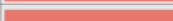
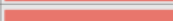
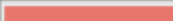
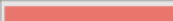

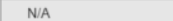

Total Files: **29**

Breakdown by Source

Code coverage metrics per source file.

Summary View Detailed View

Root Folder - L:\Troubleshoot\22a\CodeCoverage\CollectorForRepro\

File Name	Statement Coverage	Function Coverage
11 foo.m	 100%	 100%
12 liveSourceScript.mlx	 100%	N/A
13 perfTestCoverage.m	 0%	N/A
14 quadraticSolver.m	 58.82%	 50%
15 qux.m	 0%	 0%
16 reproCD.m	 0%	N/A
17 reproCobertura.m	 0%	N/A
18 reproCollector.m	 0%	 0%
19 reproCollectorMCDL.m	 0%	 0%
20 reproCollectorPFile.m	 0%	 0%
21 reproCollector_changesFileOrder.m	 0%	 0%
22 reproDecision.m	 0%	N/A

Source Details

Detailed analysis of code coverage for a source file.

Hit Count	Line Number	L:\Troubleshoot\22a\CodeCoverage\CollectorForRepro\quadraticSolver.m
1	1	<code>function roots = quadraticSolver(a, b, c, printBool)</code>
	2	<code>% quadraticSolver returns solutions to the</code>
	3	<code>% quadratic equation a*x^2 + b*x + c = 0.</code>
	4	<code></code>
	5	<code>arguments</code>
1,1,0	6	<code>a (mustBeNumeric, mustBeNonNaN) = 1</code>
1,0	7	<code>b (mustBeNumeric) = 1</code>

MATLAB Unit Test

```
classdef tMyData < matlab.unittest.TestCase
```

← Test class (inherit from TestCase class)

```
methods (Test)
```

← Group of test points

```
function tCreate(testCase)
```

← Test point – a unit test

```
    writeData = [0.9 -1.1 -1.1 -0.8];
```

```
    key = MyData.write(writeData);
```

```
    readData = MyData.read(key);
```

```
    verifyEqual(testCase, readData, writeData);
```

← Qualification

```
end
```

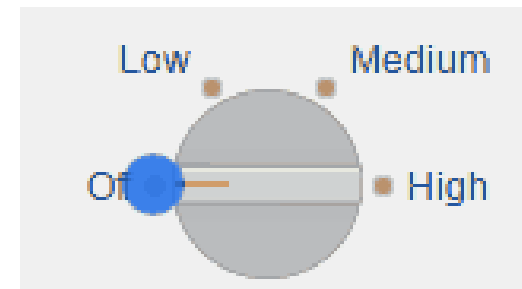
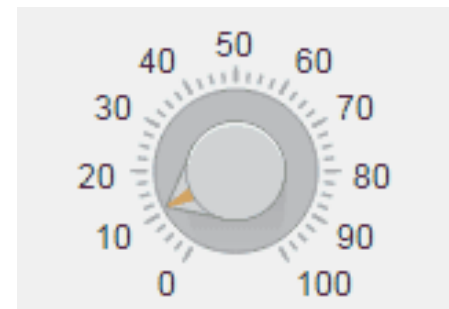
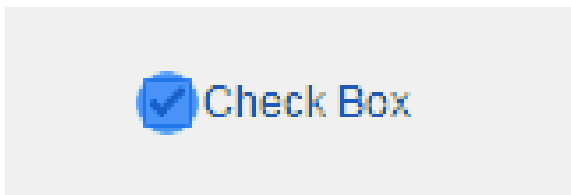
```
end
```

```
end
```

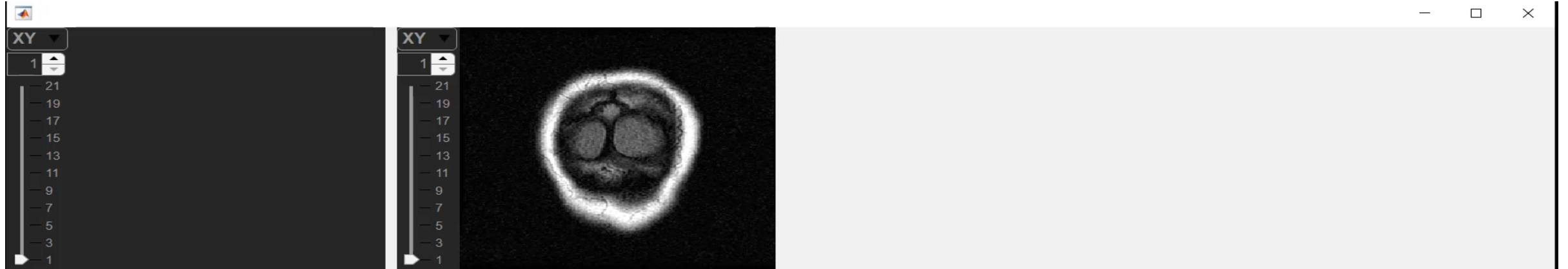
MATLAB Unit Test

App Unit Test Methods

press	Perform press gesture on UI component
choose	Perform choose gesture on UI component
drag	Perform drag gesture on UI component
type	Type in UI component
hover	Perform hover gesture on UI component
chooseContextMenu	Perform choose gesture on context menu item
dismissAlertDialog	Close frontmost alert dialog box in figure window
matlab.uitest.unlock	Unlock figure locked by app testing framework
matlab.uitest.TestCase.forInteractiveUse	Create a TestCase object for interactive use

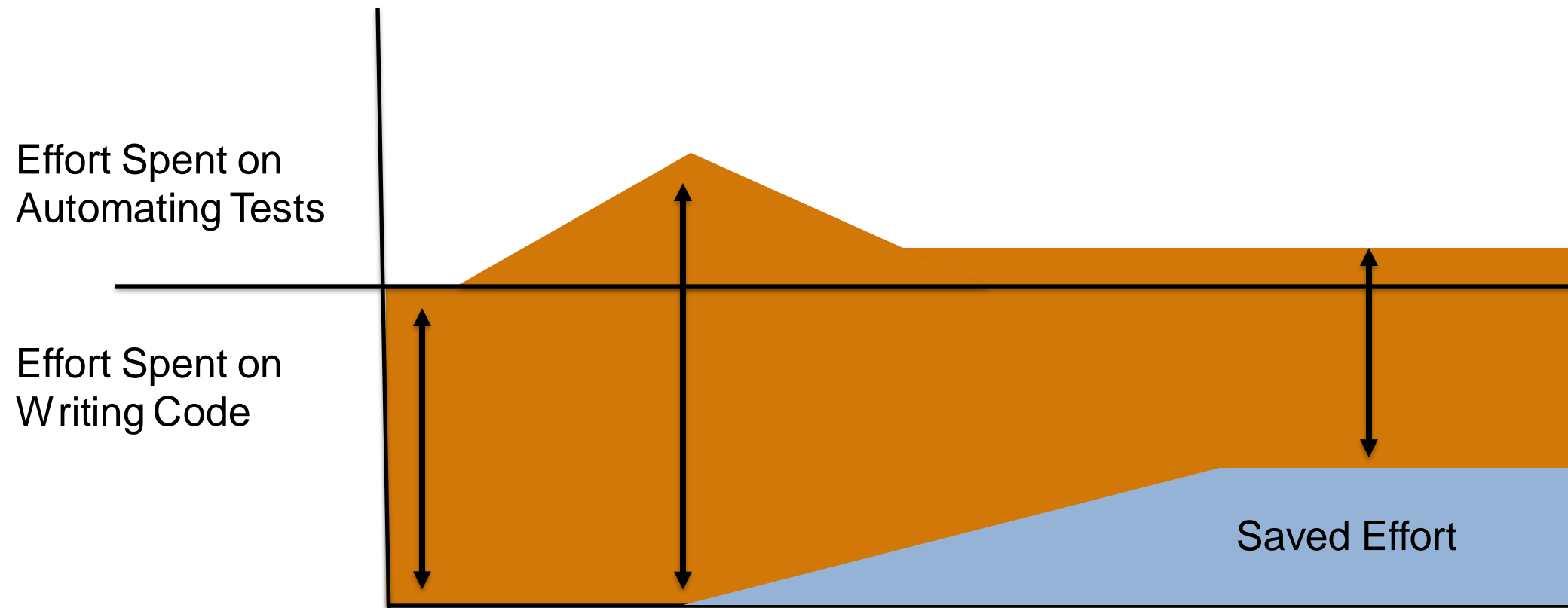


MATLAB Unit Test



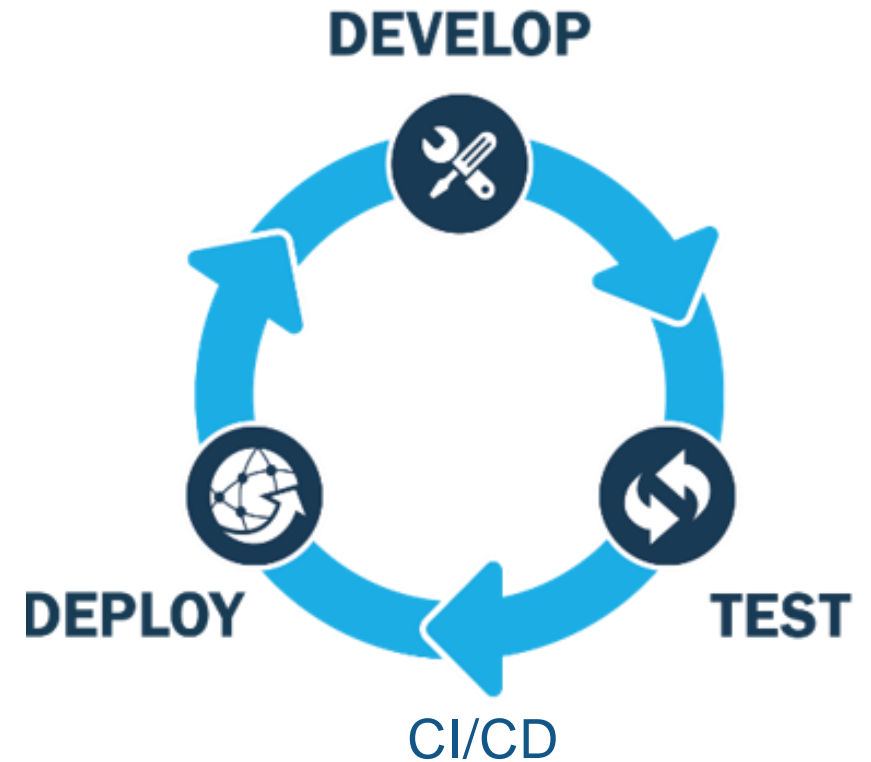
MATLAB Unit Test

- Why Automate Tests?



MATLAB Unit Test

- Test Automation
 - Improve Quality / Reduce Risk (Bug)
 - Easy to run, to Write and Maintain
 - CI/CD : Continuous Integration
Continuous Deploy/Delivery
 - CI Tool Integration(Jenkins, Gitlab , Github)



MATLAB Unit Test

The screenshot displays the MATLAB R2019b environment. The top ribbon includes tabs for HOME, PLOTS, APPS, PROJECT, and PROJECT SHORTCUTS. The PROJECT tab is active, showing options like 'Run Checks' and 'Project Path'. The current folder is 'C:\Users\promero\git_work\Demos\DUT_demo\work'. The file explorer on the left shows a 'cache' folder and a 'codegen' file. The main workspace area shows a list of files and folders, including '.SimulinkProject', '.gitmodules', and 'ci-test-manager'. The 'Status' column shows a green checkmark for the selected file. The 'Git' section indicates the current branch is 'master' and the branch status is 'Normal'. The 'Command Window' at the bottom shows the prompt 'f >>'.

MATLAB EXPO

Thank you



© 2023 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.