Object-Oriented Programming with MATLAB

Prerequisites

MATLAB Programming Techniques or equivalent experience using MATLAB.

Day 1 of 2	
Creating Custom Data Types	**Objective:** Organize your files into packages. Learn some of the basic techniques and benefits of object-oriented programming and experience the differences between procedural and object-oriented programming. Creating a namespace by storing multiple files in a package Encapsulating data and functionality into a single class definition file Documenting custom data types Creating and using variables of custom data types
Designing a MATLAB Class	**Objective:** Make objects reliable by separating interface and implementation. Enhance code maintainability by avoiding code duplication. Customize standard operations for your classes. Defining safe interactions via data access methods Designing the public class interface with property and method attributes Customizing standard operators for your class Avoiding code duplication through internal refactoring
Building Class Hierarchies	**Objective:** Relate multiple similar classes via a common superclass. Extend the generic superclass by specializing its behavior in the subclasses. Creating a superclass via abstraction Inheriting from a superclass Defining abstract properties and methods Implementing specialized behavior in subclasses

Day 2 of 2	
Facilitating Multiple References	**Objective:** Embed one class into another via aggregation. Distinguish the use cases for pass-by-value vs. pass-by-reference behavior. Define a class that exhibits reference behavior. Creating a viewer class containing a data class Writing context-sensitive (polymorphic) code Referencing one data object from multiple viewer objects Choosing between handle and value classes

Day 2 of 2	
Writing Unit Tests	**Objective:** Guarantee correct behavior by writing formal tests for the corresponding unit of code. Use the unit-testing framework provided within MATLAB. Enhance the quality and flexibility of your software. Overview of the MATLAB unit testing framework Writing a test method Creating a test environment using setup and teardown methods Parameterizing a test method Testing for error conditions Aggregating and running suites of tests Logging test and coverage results
Synchronizing Objects	**Objective:** Automatically react to property changes using predefined events, listeners, and callbacks. Trigger function calls based on custom events. Events, listeners, and callbacks Using predefined property events Querying class meta information Defining property listeners Implementing a callback function Defining custom events and their callbacks