

LabVIEW Object-Oriented Programming

David Bonal
National Instruments
512-553-5683
david.bonal@ni.com



ni.com



Agenda

- Newest Math Features in LabVIEW
- Object-oriented programming
- Benefits of OO
- OO development in LabVIEW



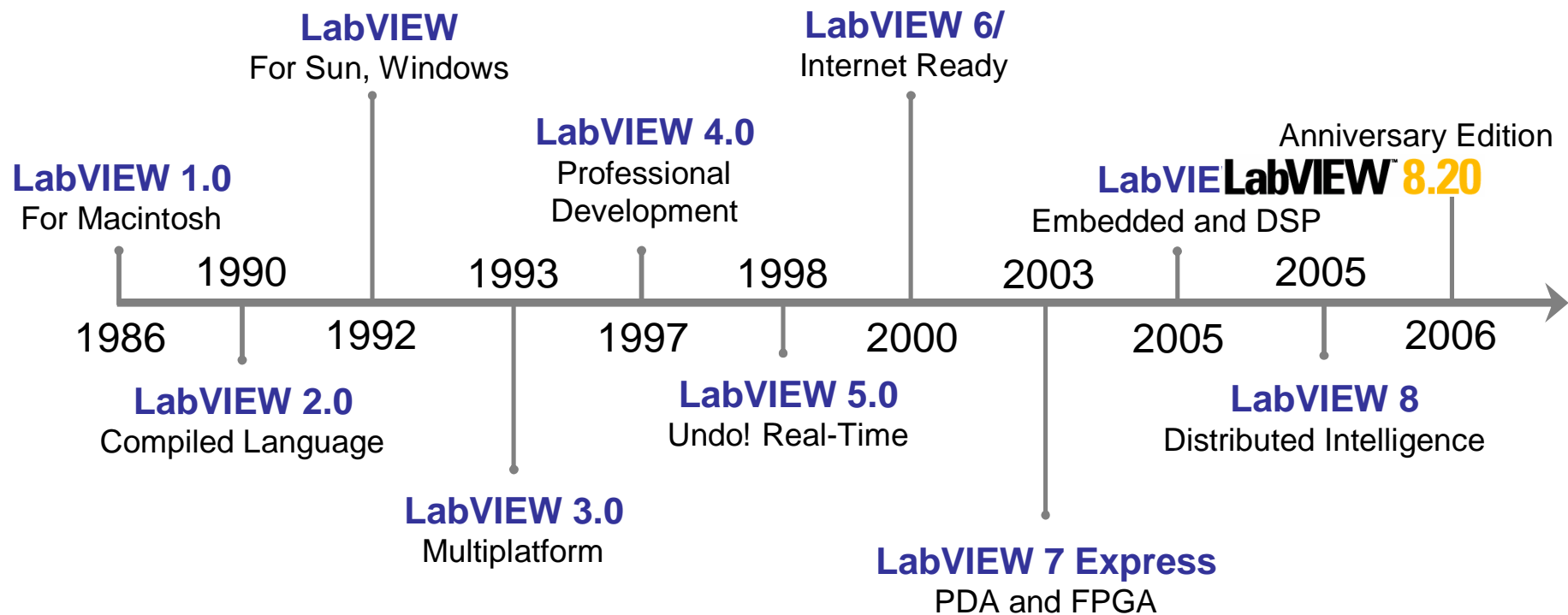


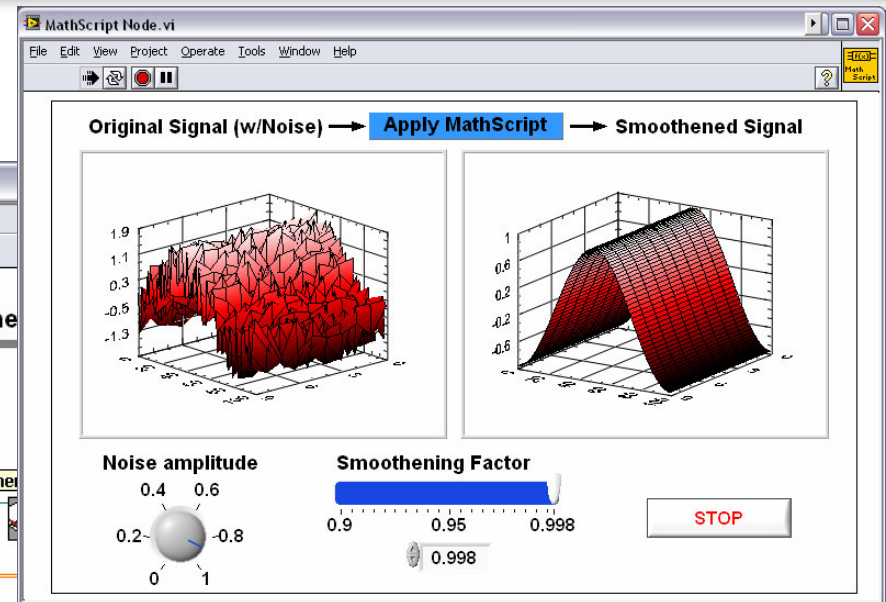
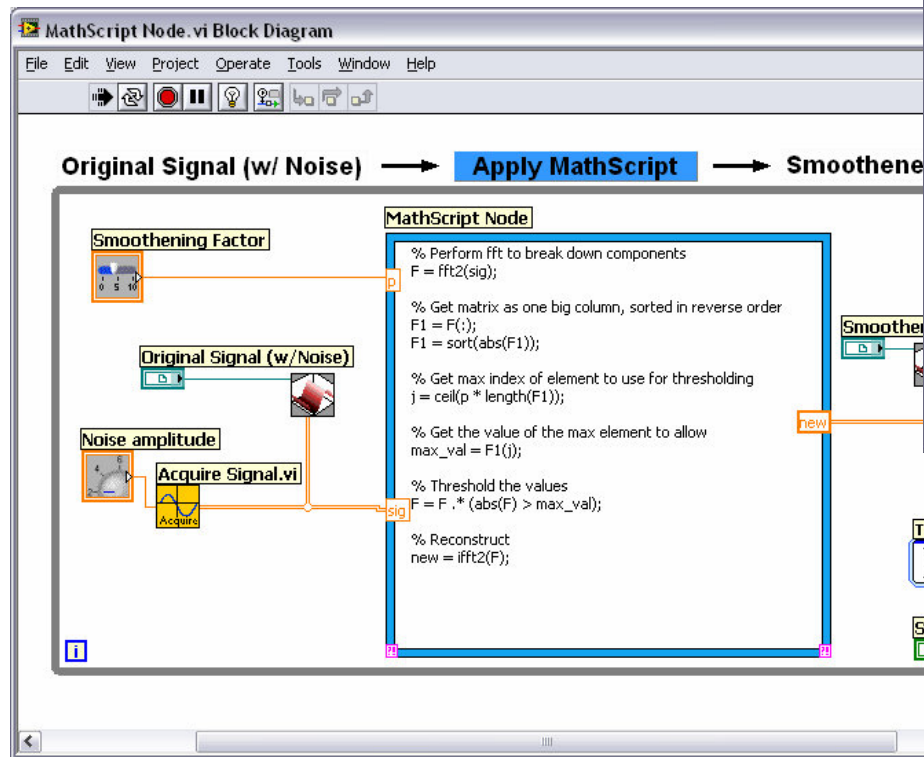
Easy. Powerful. Open.

- Make common measurements **EASY** with any board, instrument, or bus
- Help users build **POWERFUL** solutions with standard PC technology
- Integrate external tools and technologies with an **OPEN** platform



20 Years of Innovation



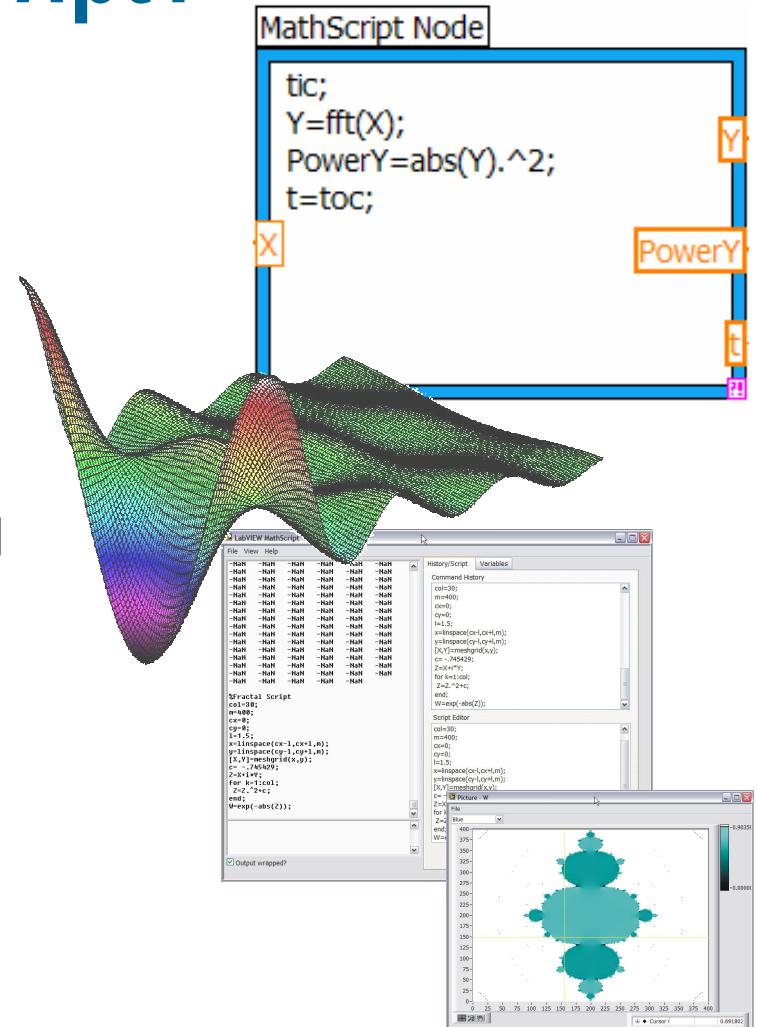


- Combine textual math & graphical programming
- Reuse many of your m-file scripts created with The MathWorks, Inc. MATLAB® software



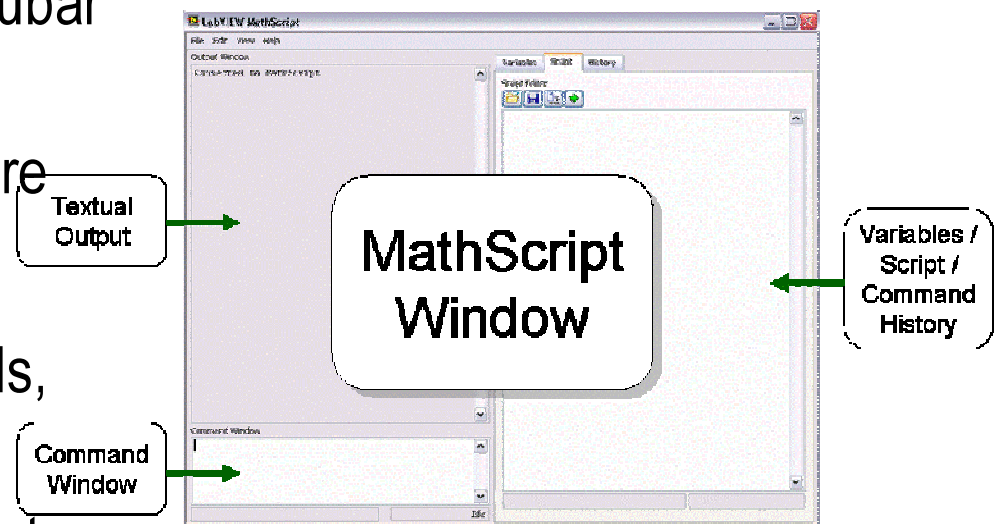
What is LabVIEW MathScript?

- **Powerful textual programming for Signal Processing, Analysis and Math**
 - Over 650 built-in functions
 - Reuse many of your m-file scripts created with The MathWorks Inc.'s MATLAB® and others
 - Based on original math from MATRIX_x
- **A native LabVIEW solution**
 - Interactive and programmatic interfaces
 - Does not require 3rd party software



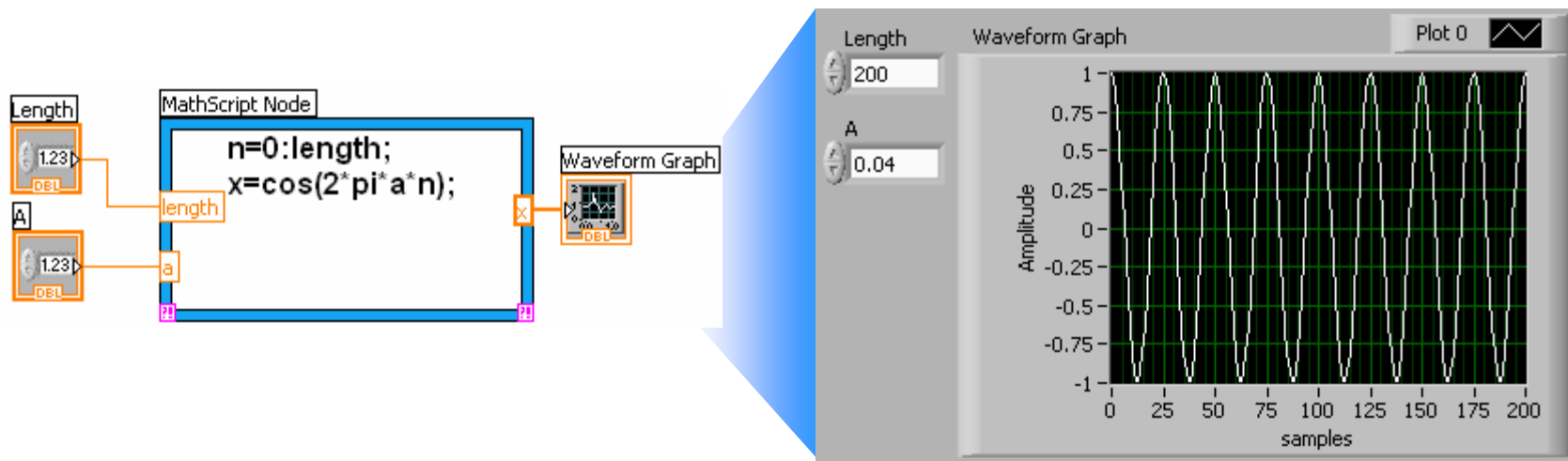
Interactive Math with the MathScript Window

- Fully integrated into LabVIEW
 - Access from LabVIEW menubar (Tools→MathScript Window)
 - No need for 3rd party software
- Interactive interface
 - Enter m-file script commands, see immediate response
 - Open / run saved m-file scripts
 - View text output, command history, variables, and plots



Programmatic Math with the MathScript Node

- Combine graphical system design with textual math
- Deploy with LabVIEW graphical programming
- Implement equations and algorithms with text



Object-Oriented Programming

- An approach to application development
- Appropriate for large scale applications with teams of developers



Benefits of OO Development

- Promotes code-reuse
- Reduces code maintenance
- Simplifies extending applications



Modular Development in LabVIEW

- VIs & sub-VIs
- Project Libraries (LabVIEW 8)
- Classes (LabVIEW 8.20)



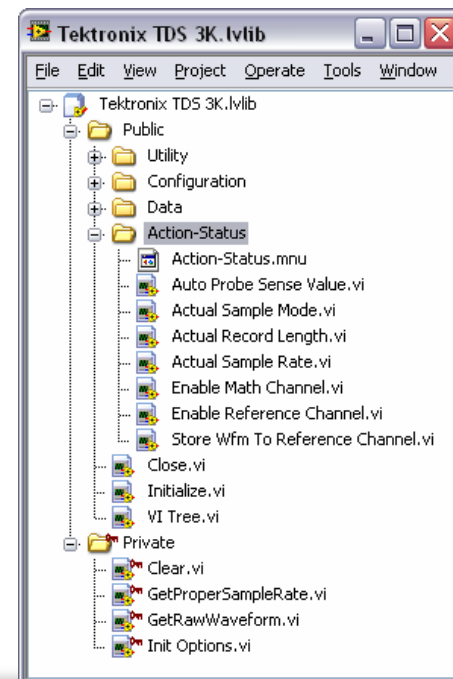
SubVIs

- SubVIs improve code readability and maintainability
- Signs that you need subVIs:
 - The same code appears twice on the block diagram
 - Block diagram is larger than one screen
 - Repeated use of sequence structures



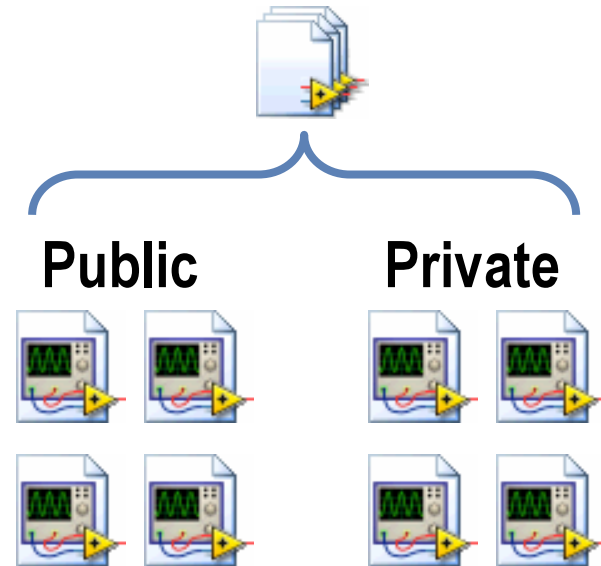
Project Library

- Set of related VIs and other LabVIEW files
- Information on library is stored in a text file (.lvlib)

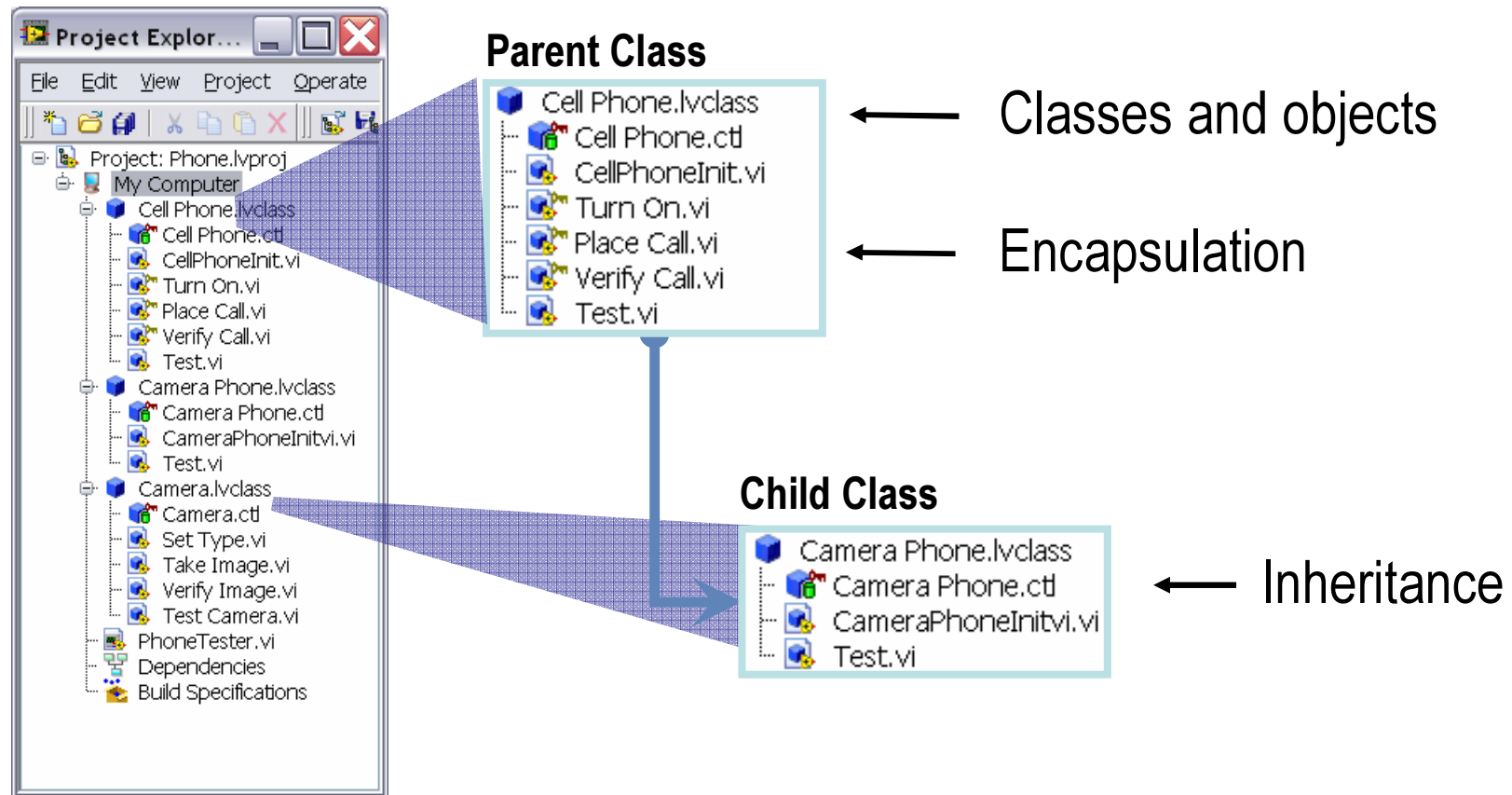


Advantages of Project Libraries

- Reduced naming conflicts
 - **Namespace** applied to VI names
- Restricted access to specific VIs
 - **Public VIs**
 - **Private VIs**
- Consistent icon appearance



Object-Oriented Programming



- For power programmers and large-scale application development



Classes & Objects

- Objects are actors in your application
 - Refer to individual pieces of data
- A Class defines the data and behavior of objects
 - Objects in your application are instances of a class



OO Design - Testing Cell Phones

- Test cell phones and camera phones by the cell phone placing calls and the camera phone placing calls and taking pictures
- Break the application into “nouns” and “***action verbs***”
 - Camera phones are types of cell phones that also contain a camera
 - Cell phones ***make calls***
 - Camera phones ***make calls*** and ***take pictures***
 - To ***test*** a cell phone a call must be ***placed*** and ***verified***
 - To ***test*** a camera phone the cell phone and camera functionality must be ***tested***
 - To ***test*** a camera an image is taken and compared to a reference
- Nouns generally map to classes or objects
- Verbs generally map to functions



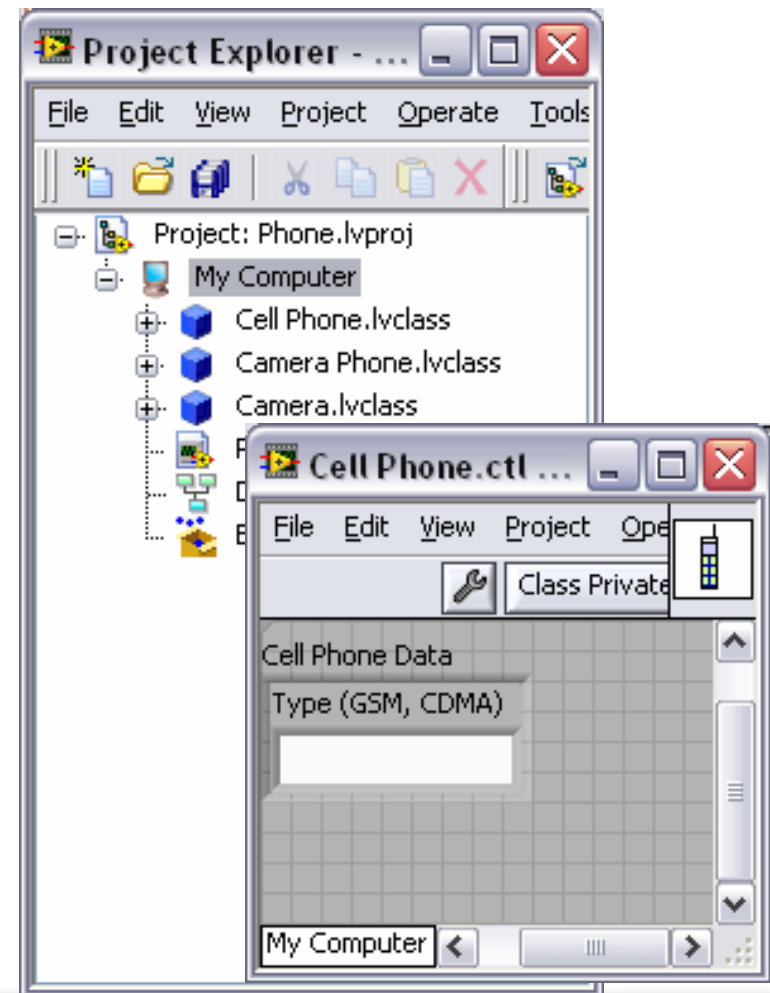
Creating Classes in LabVIEW

- Create a class in the project
- Specify the data with the .ctl for the class
 - Defining a class in effect defines a new data type
- Additional features
 - Specify the class icon
 - Specify a VI icon template
 - Specify the wire color



Testing Cell Phones - Classes

- Cell Phone
- Camera Phone
- Camera



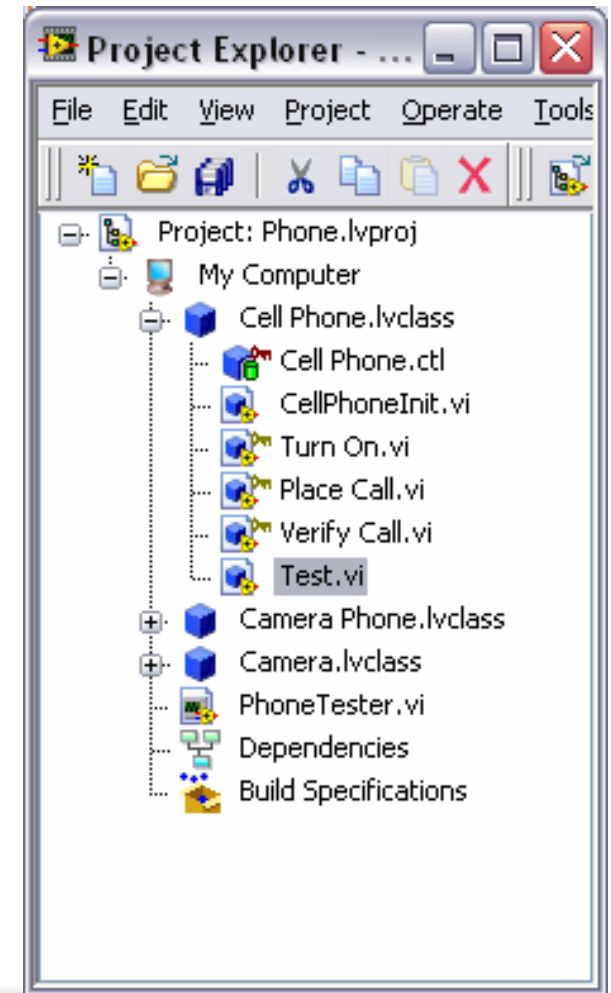
Creating Methods for a Class

- Methods
 - Actions or requests
 - Performed by objects
 - Generally verbs
- Create a VI
- Specify scope
 - Public
 - Private
 - Protected



Testing Cell Phones - Methods

- Cell Phone
 - Data
 - Phone Type (CDMA, GSM)
 - Methods
 - Initialize
 - Turn On
 - Place Call
 - Verify Call
 - Test



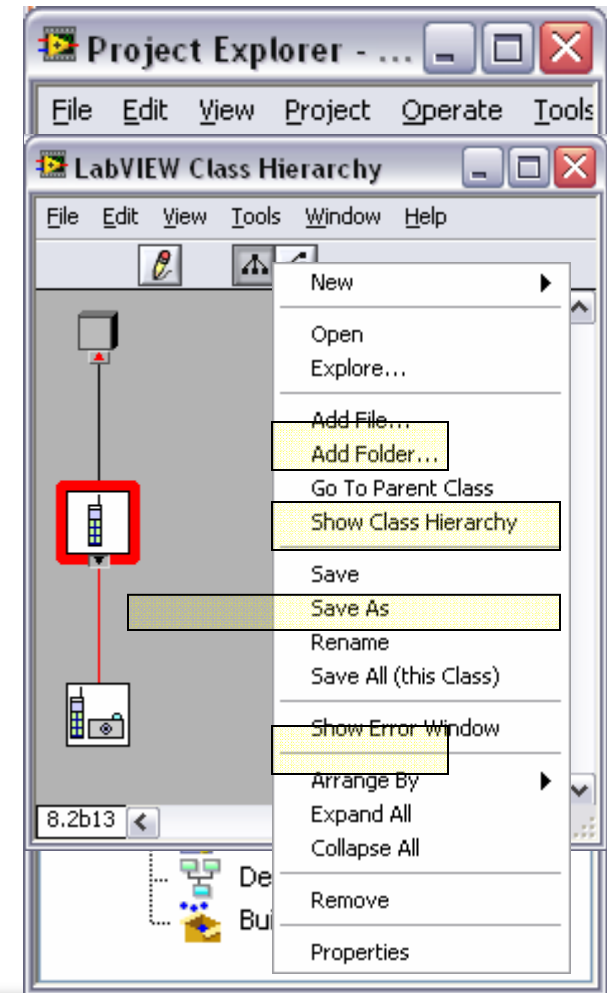
Inheritance

- Defines subclasses
- Creates “is a” relationship
 - Example: Camera Phone “is a” Cell Phone
 - Re-use common functionality
- Specialization
 - Extend or override common functionality for specific needs



Testing Cell Phones - Inheritance

- Camera Phone Class
 - *Inherits from Cell Phone*
 - Data
 - Camera
 - Methods
 - Test – Extends “Cell Phone Class”
Test method to test camera functionality



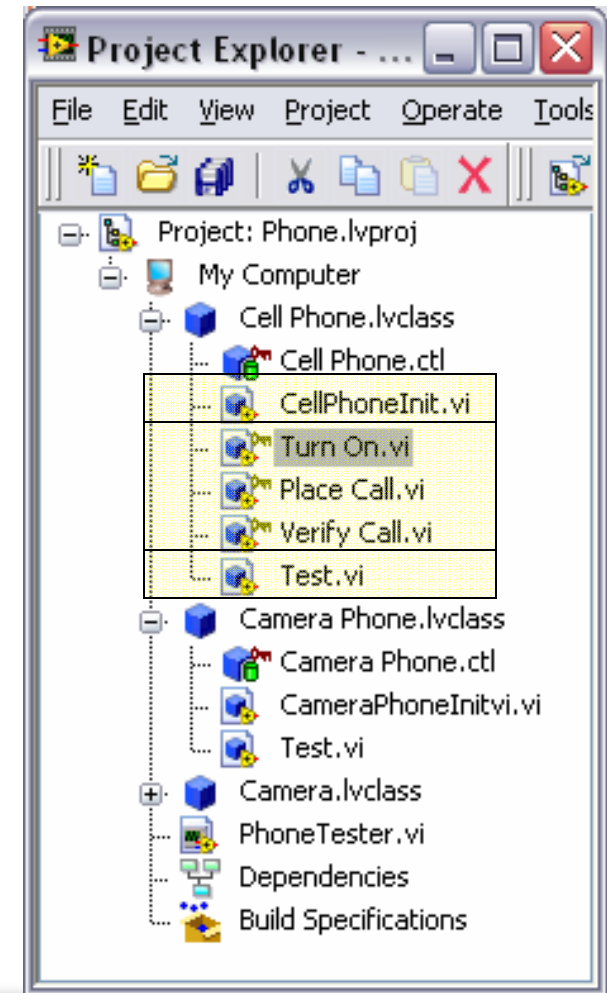
Encapsulation

- Treat each object as a black-box
 - Well defined interface of data and methods
 - Must use this interface in the application
- All data is private
- Methods can be public, private or protected



Testing Cell Phones - Encapsulation

- Top level application only needs to initialize phone and test it
 - *Init* and *Test* methods are public
- Camera phone needs to be able to turn on the phone
 - *Turn On* method is protected



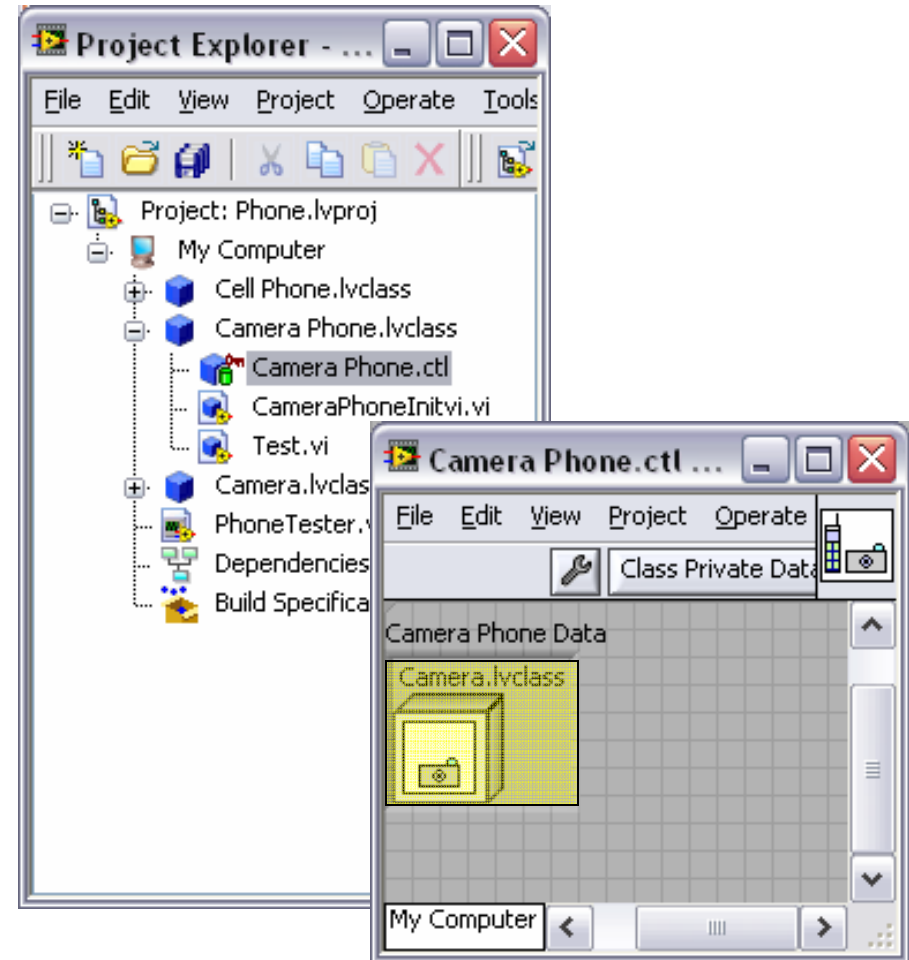
Class Composition

- Defining a classes creates a new data type
- A class can be made up of other classes



Testing Cell Phones – Class Composition

- Camera Phone Class
 - Inherits from Cell Phone
 - Data
 - Camera Class
 - Methods
 - Test (Cell Phone and Camera)
- Camera Class
 - Data
 - Camera Type
 - Methods
 - Take Image
 - Verify Image
 - Test Camera

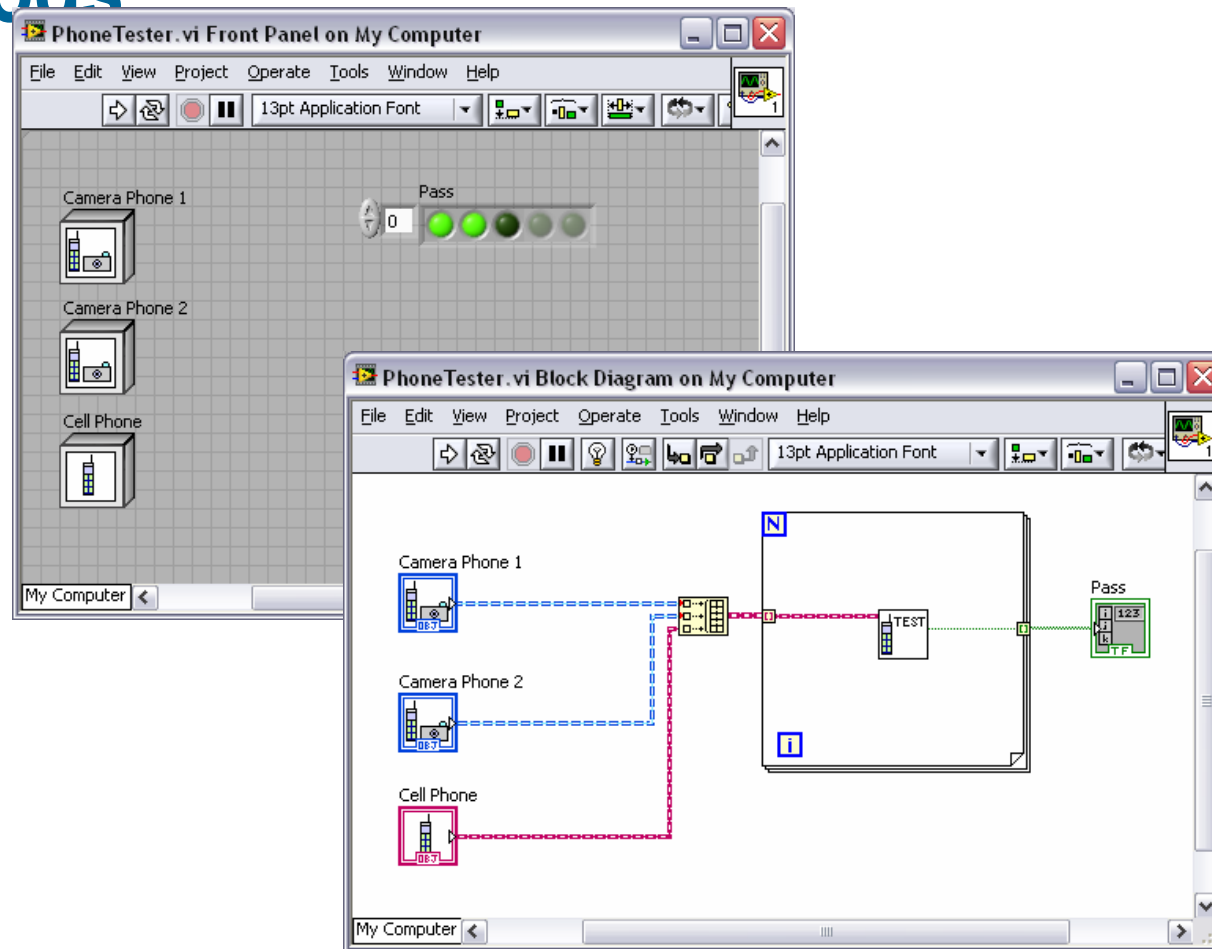


LabVIEW Application using Classes

- Call methods on objects
- Reduce code rework with inheritance and dynamic dispatching



Testing Cell Phones – Calling Test Methods



Resources

- ni.com/labview/upgrade
 - Multi-media demonstration
 - Application note

