LabVIEW Object Oriented Programming

Steven Hoenig
Business Unit Leader – NY/NJ
Certified LabVIEW Architect
Certified Professional Instructor
Agenda

• About Bloomy Controls
• Object Orientation Concepts
• Benefits of Object Oriented Programming
• Real-Life Examples
• Common Concerns
• Questions
Who is Bloomy Controls?

Bloomy Controls is a full service integrator providing turnkey systems, consulting, and training for Test and Measurement systems.

- Founded in 1991
- Windsor, CT; Marlborough, MA; Fort Lee, NJ
- Industry Leader in NI LabVIEW development
- NI Select Alliance Partner
- 16 Certified LabVIEW Architects
- CSIA (Control Systems Integrators Association) Certified Member
LabVIEW Style Experts

- Best practice for developing quality LabVIEW applications
- Over 200 style rules
  - Ease of use
  - Efficiency
  - Readability
  - Simplicity
  - Performance
  - Maintainability
  - Reliability
- Companion website at www.bloomy.com/lvstyle
Agenda

• About Bloomy Controls
• **Object Orientation Concepts**
• Benefits of Object Oriented Programming
• Real-Life Examples
• Common Concerns
• Questions
Object Orientation

- A software engineering methodology / philosophy
- Break a system down into discrete, independent entities (or “Objects”) with a single, distinct role or responsibility
- Accepted as one of the best techniques for modeling complex systems
- Promotes greater reliability, flexibility and maintainability in programming
• Consider these descriptions from a recent NI Virtual User Group presentation:

“We want to test each board at the end of an assembly line to make sure each is functional. We want to send an email to management for any failed tests”

and

“We want the assembly line to produce boards. Each board should test itself for functionality and report any problem. The log should send an email to management for any failed tests”
What is an Object

- Data and actions are grouped together into one self-contained unit called an “Object”

- An object’s definition is known as a “class”
OOP In LabVIEW = LVOOP

- An Object is similar to a cluster
  - Cluster + Library (lvlib) + magic
- Objects are defined in terms of their “class”
- Classes are defined in LVClass files and contain:
  - A data-type (the “cluster of private class data”, or “class cluster”)
  - A list of “member” VIs
  - A description of what the wire should look like
  - Inheritance Information (more on this later)
- Native to LabVIEW 8.20+ (PCs), and LabVIEW 2009+ (Real-Time and FPGA)
In LabVIEW...

DEMO: Creating a class

• Goal: A class to represent a shape in a drawing application

• What should a particular shape know about itself?
  • It’s location
  • It’s color / style

• What should any shape be able to do?
  • Allow its color and style to be programmatically changed
  • Draw itself on a LabVIEW 2D Picture Control
Agenda

- About Bloomy Controls
- Object Orientation Concepts
- **Benefits of Object Oriented Programming**
- Real-Life Examples
- Common Concerns
- Questions
Benefits: Scope and Encapsulation

- A VI’s scope is a description as to where that VI can be used.
- Change implementation without affecting other parts of your program – “Encapsulation”
- Helps limit an object to a specific, single role as we
- Classes have four scope levels:
  - Public
  - Private
  - Protected (talk about this later)
  - Community (outside the scope of this presentation)
- Allows you to have internal functionality that does not have to be checked against all use-cases (e.g. can presume that an array is sorted and has 10 elements)
Benefits: Scope

DEMO: Using Scope

• How to mark parts of our class as Private

• Task:
  • Restrict the number of colors that can be used to one of a discrete list
  • Code to convert color name into raw color value should be internal to the class so that we can make presumption / change the values later
Benefits: Inheritance

- Classes can have “parents” and “children” (also known as “ancestors” and “descendants”)
  - A circle is a kind of shape, or a circle is a descendant of shape
- A child class automatically knows the same information and can do the same actions as its parent, along with anything else more specific
- Allows common code to be developed once
  - Child VIs need only add new, or replace (“override”) functionality that is different from the parent
  - Less repetition = more reliability
Benefits: Inheritance

• An object can flow down any of it’s own wires or any of it’s parents wires.
  • LabVIEW keeps track of what object is actually on a wire at a given point
  • E.g. a circle can travel along a shape wire and can use shape VIs.
• LabVIEW will select the correct VI at RUNTIME (e.g. parent’s X.vi or child’s X.vi)
  • Called Runtime Polymorphism or Dynamic Dispatch
  • LabVIEW will choose the correct functionality for you
Benefits: Inheritance

DEMOS: Inheritance and Dynamic Dispatch

- Task:
  - Create a Circle class that inherits from Shape

- Demonstrate LabVIEW selecting correct VI at runtime
Benefits: Good Style

- Encapsulation and modularity are key pillars of Bloomy Controls’ Style Guidelines, even without LVOOP
  - Drivers and other modules should encapsulate key functionality and function as independent entities
- Why not let LVOOP do the hard work for you?
  - Very low overhead
  - Enforces modularity through scope protections
  - Allows you, the developer to control how your code is used.
  - Built in versioning can allow your program to automatically adjust for new versions of a class.
Benefits Summary

• Encourages good design
  • Enforces encapsulation and protects data
  • Forces code to be used as the developer intended

• Permits run-time polymorphism
  • Allows interface to be defined regardless of implementation

• Can lead to better, more stable code
  • Code can be based on thoroughly tested starting point
  • Scalable without affecting existing software
Agenda

- About Bloomy Controls
- Object Orientation Concepts
- Benefits of Object Oriented Programming
- **Real-Life Examples**
- Common Concerns
- Questions
Use Cases: Drivers

• Most logical use case – many devices carry out the same functionality in different ways.
• A class allows the functionality for an application to be defined, regardless of implementation
• Hardware can be easily replaced without any changes in main application code
• A motor framework that would allow 3 different kinds of motor to be used with a system
• A “Virtual Motor” was also created to allow the system to be tested without the hardware present
Use Cases: Data Logger

• In a typical Data Logger a large number of instruments are periodically polled for data. This data is then written to file.

• Timing is typically non-critical and all instruments can post their data to the log at the “same” time.

• All instruments function the same way from the software’s point of view. (Ask for data, get a double.)

• In an object-oriented design all instruments can inherit from a base “instrument” class having a “read” method.

• The program can now simply loop through an array of instruments and call “read”.
Use Cases: Plug-Ins

• Classes can be dynamically loaded from a path

• Created software to manage data from multiple devices that communicated the same kind of data over different interfaces (including RS232, USB, and Wireless Ethernet).

• Software architected so that each type of device has its driver dynamically loaded as a plug-in.

• Allows new devices to be added to software without rebuilding EXE.
Agenda

• About Bloomy Controls
• Object Orientation Concepts
• Benefits of Object Oriented Programming
• Real-Life Examples
• **Common Concerns**
• Questions
Common Concerns

• A LOT of VIs
  • Each interface, interaction and data manipulation becomes its own VI, leading to many small VIs

• DESIGN IS CRITICAL
  • More design effort is required up front and, because classes are independent, changing a parent class VI may require changing many child VIs *(upfront design is not necessarily a bad thing...)*
Development Time?

- Like any engineered system, LVOOP requires up-front design effort.
- Payoffs are a function of reuse and code simplicity.
- Less prone to bugs and can be independently tested.
- Inheritance can dramatically reduce time needed for similar components.
- Code can be more easily reused within the same as well as other applications.
- Do not have to hard code all aspects of your software up front.

Definition 1.1: Development time includes the hours required to develop, document, test, modify, and maintain an application throughout its entire life cycle.

(Adapted from: LabVIEW Style Book, P. Blume)
Conclusion

LabVIEW Object Oriented Programming offers a powerful set of tools to help you:

- Simplify the process of designing applications
- Create code that is more modular through encapsulation and scope control
- Leverage language-level tools to help ensure that your code is used as intended
- Define functionality at runtime, instead of having to hard code how your application will run
- Improve maintainability by allowing implementations to change without affecting the overall application
- Design software that can grow
“A VI outside a class is a gun without a safety. Data outside a class is a target”

A message from LabVOOP R&D (courtesy of LAVA member “Aristos Queue”)
Contact Bloomy Controls

Steven Hoenig
NJ Unit Leader
Certified LabVIEW Architect | Certified Professional Instructor
Office: 201.944.9890 | Mobile: 201.240.8749
Steven.hoenig@bloomy.com

More information and downloads:
www.bloomy.com

Increase Productivity. Improve Quality. Reduce Cost.
Additional Resources

- LAVA Object-Oriented Forum

- LabVIEW Object-Oriented Programming: The Decisions Behind the Design
  - [http://zone.ni.com/devzone/cda/tut/p/id/3574](http://zone.ni.com/devzone/cda/tut/p/id/3574)

- NI Large Application Development Group
  - [http://decibel.ni.com/content/groups/large-labview-application-development](http://decibel.ni.com/content/groups/large-labview-application-development)

- Bloomy Controls website
  - [http://www.bloomy.com](http://www.bloomy.com)