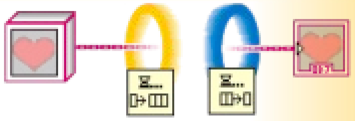


LabVIEW Object-Oriented Programming

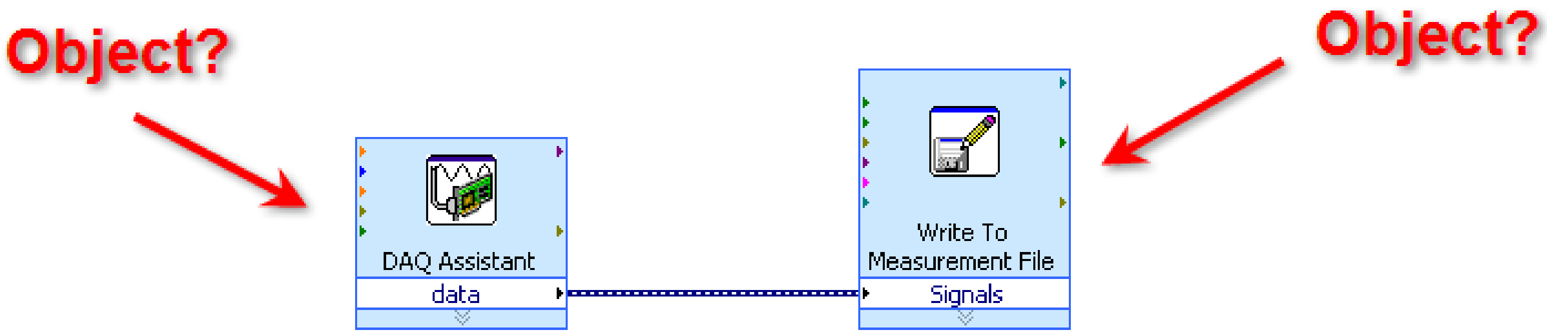
Concepts, Use Cases and Best Practices

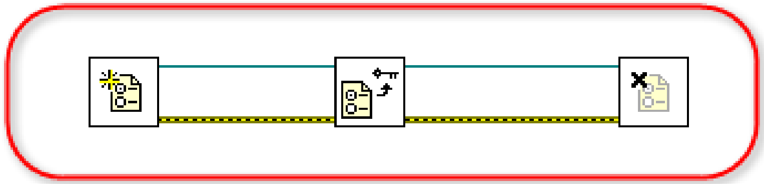
Jeffrey Habets
NI Certified LabVIEW Architect
www.vi-tech.nl

Agenda

- Object-Oriented Concepts
  - What is it?
  - Why use it?
- LVOOP
  - Use of native LV classes, by-value
  - Manipulation of object data
  - Inheritance
- GOOP
  - By-reference possibilities
  - Tools

Hasn't LabVIEW Always Been “Object-Oriented”?





What Is Object-Oriented Design?

- It's a way of structuring your software
  - OOD requires the programmer to think of a program in terms of objects, instead of procedures / VI's
- An object:
  - Encapsulated data and the methods for accessing that data
  - “Cluster + VI's”
  - Group of VI's with a common responsibility

What Is Object-Oriented Programming?

- OOP uses objects and their interactions to design applications
- OOP is bases on programming techniques such as encapsulation, inheritance and polymorphism

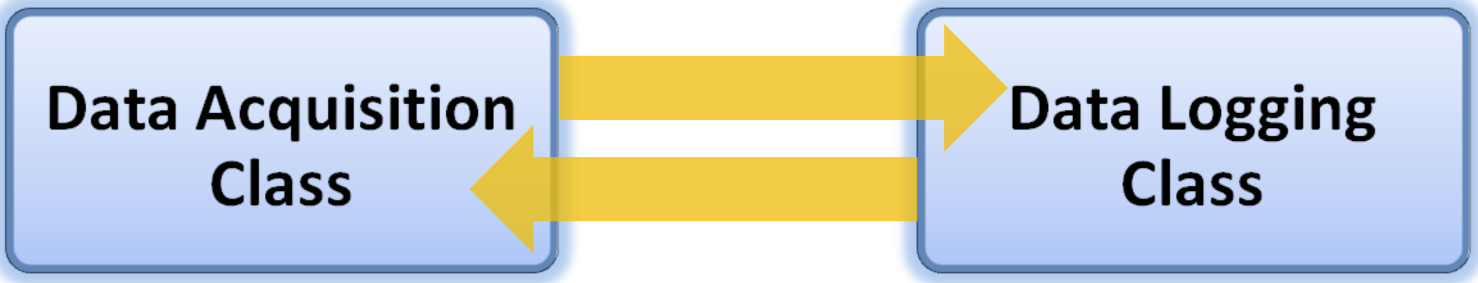
When and why to use Object-Orientation?

- Use it when you need
  - Encapsulation
  - Inheritance
  - Dynamic dispatching (polymorphism)
- Benefits of OOP
  - Easier to maintain your code
  - Easier to extend your code
  - Easier to test your code
  - Increase of code reuse
  - Benefits increase when the system grows

Example: Large Test Application

One object can communicate to another without knowledge of its internal organization

- Internal structure can change over time
- Interfaces (public methods) must remain the same



Common OOP Languages

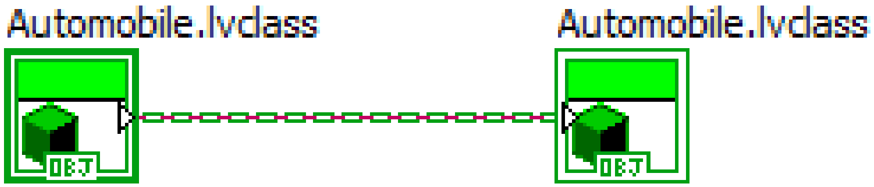
- C++
- C#
- Java
- Objective-C
- Perl
- Python
- LabVIEW 8.20 and later

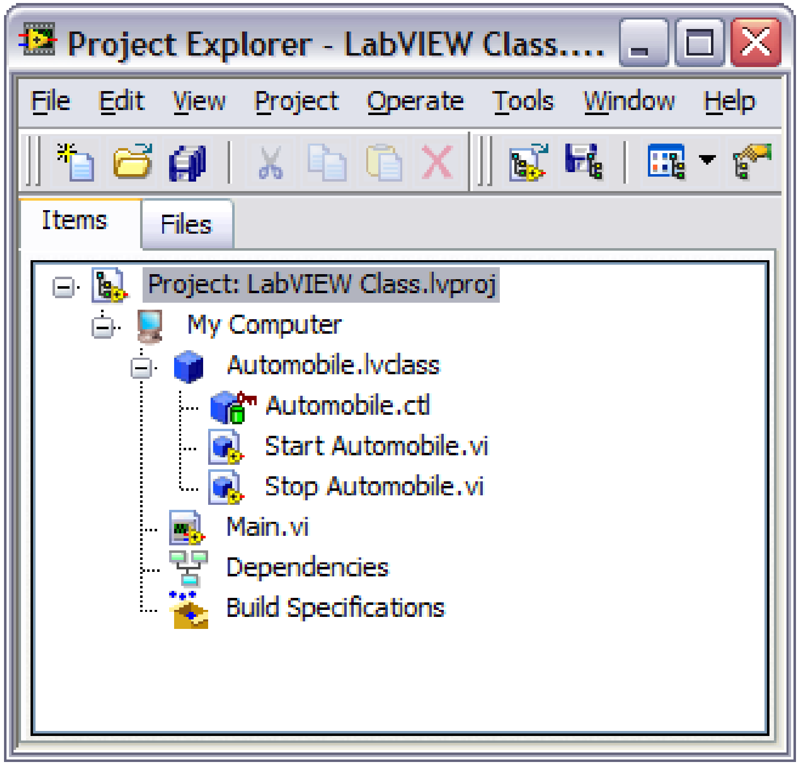
Example: Circuit Board Test

- Scenario
  - LabVIEW-based circuit board test system
- Requirements
  - Different types of boards must be tested
  - New types of boards will be added in the future
- Goals
  - Maximize code reuse and system scalability

What is a LabVIEW class?

- A glorified cluster
- A user-defined data type
- A type of Project Library

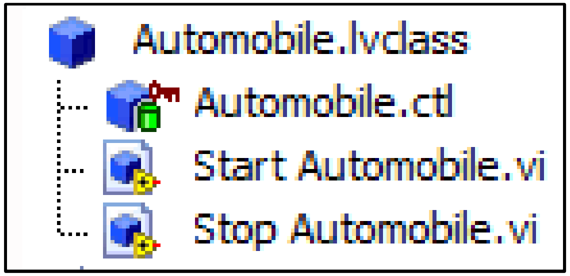



Project Explorer - LabVIEW Class....
File Edit View Project Operate Tools Window Help
Items
Files

- Project: LabVIEW Class.lvproj
  - My Computer
    - Automobile.lvclass
      - Automobile.ctl
      - Start Automobile.vi
      - Stop Automobile.vi
    - Main.vi
    - Dependencies
    - Build Specifications

Anatomy of a class

- Each LabVIEW class consists of:
  - A private data control (cluster)
  - Member VIs to access that data



Automobile.lvclass

Automobile.ctl

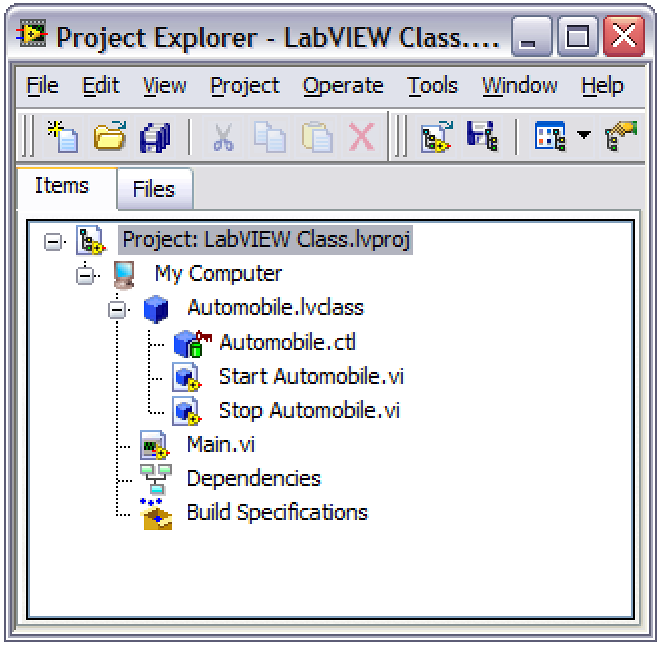
Start Automobile.vi

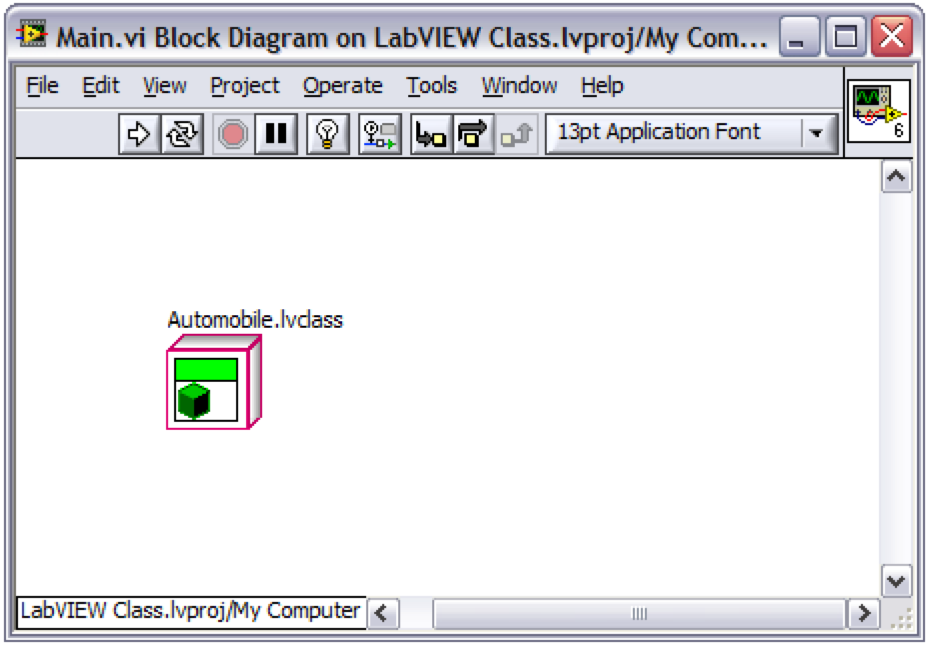
Stop Automobile.vi

- Class file (.lvclass) stores class information
  - Private data control definition
  - List of member VIs
  - Properties of member VI

What is an Object?

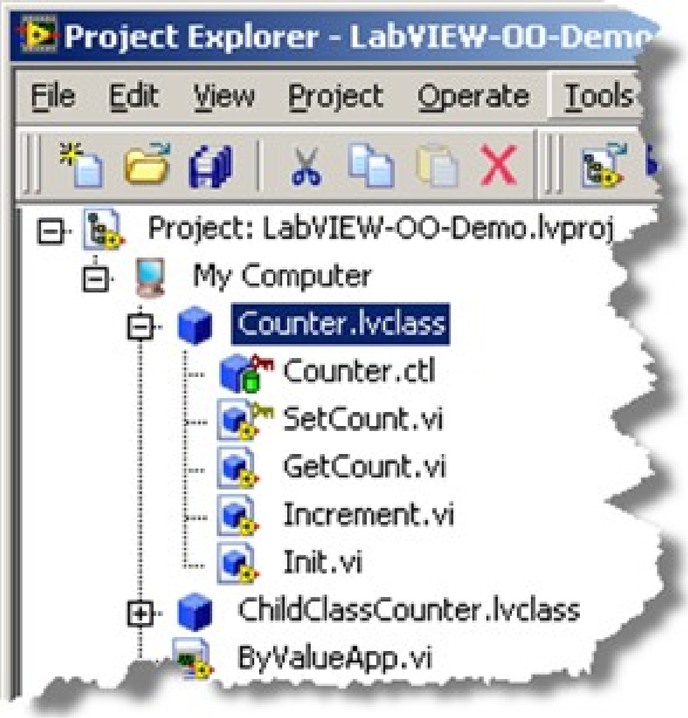
- An object is a specific instance of a class
- Object data and methods are defined by the class




Main.vi Block Diagram on LabVIEW Class.lvproj/My Com...
File Edit View Project Operate Tools Window Help
13pt Application Font
Automobile.lvclass
LabVIEW Class.lvproj/My Computer

DEMO: A class in LabVIEW

1. Create and explore a class
2. Class: Counter and the By ValueApp.vi
3. Class constant, read-write data
4. Class icon template and wire



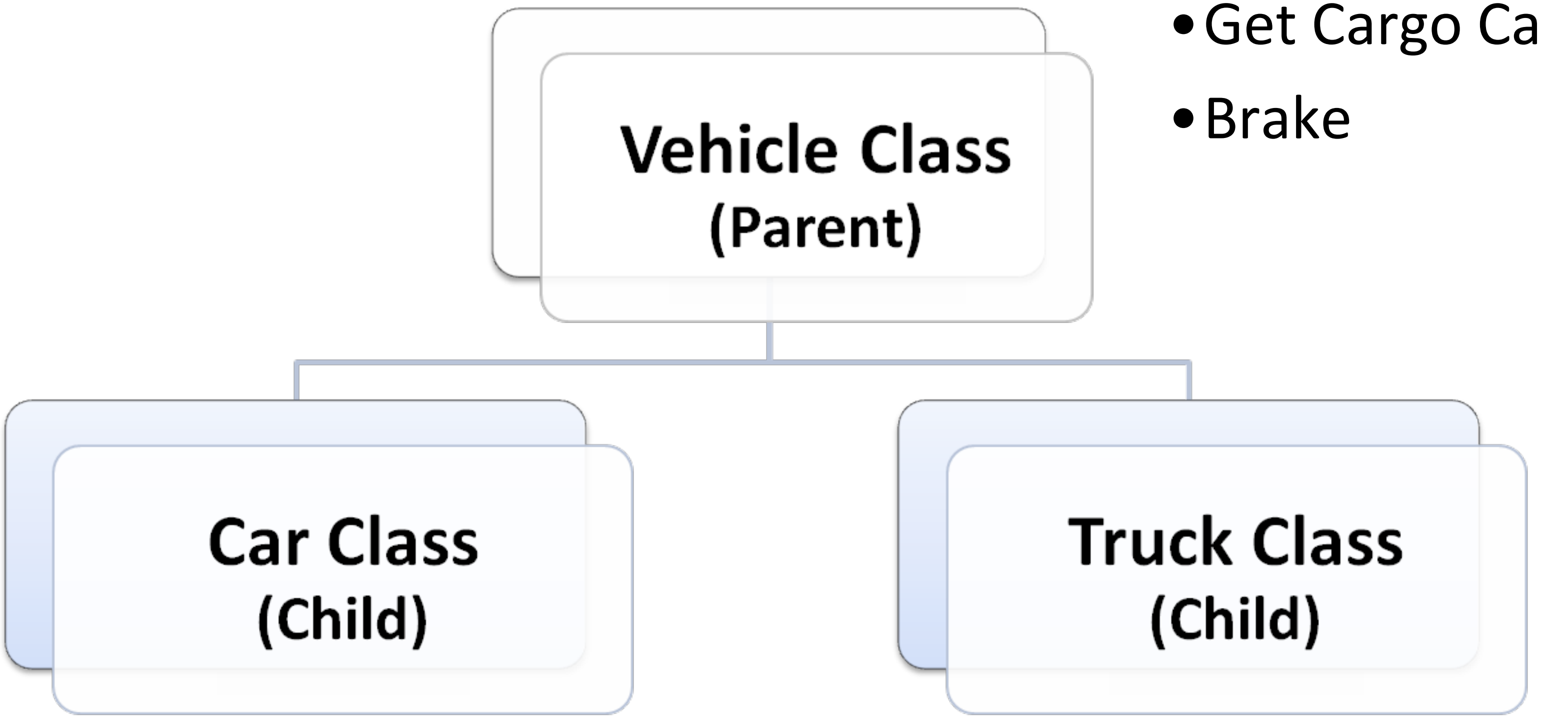
Project Explorer - LabVIEW-OO-Demo

```
Project: LabVIEW-OO-Demo.lvproj
  My Computer
    Counter.lvclass
      Counter.ctl
      SetCount.vi
      GetCount.vi
      Increment.vi
      Init.vi
    ChildClassCounter.lvclass
    ByValueApp.vi
```

What Is Inheritance?

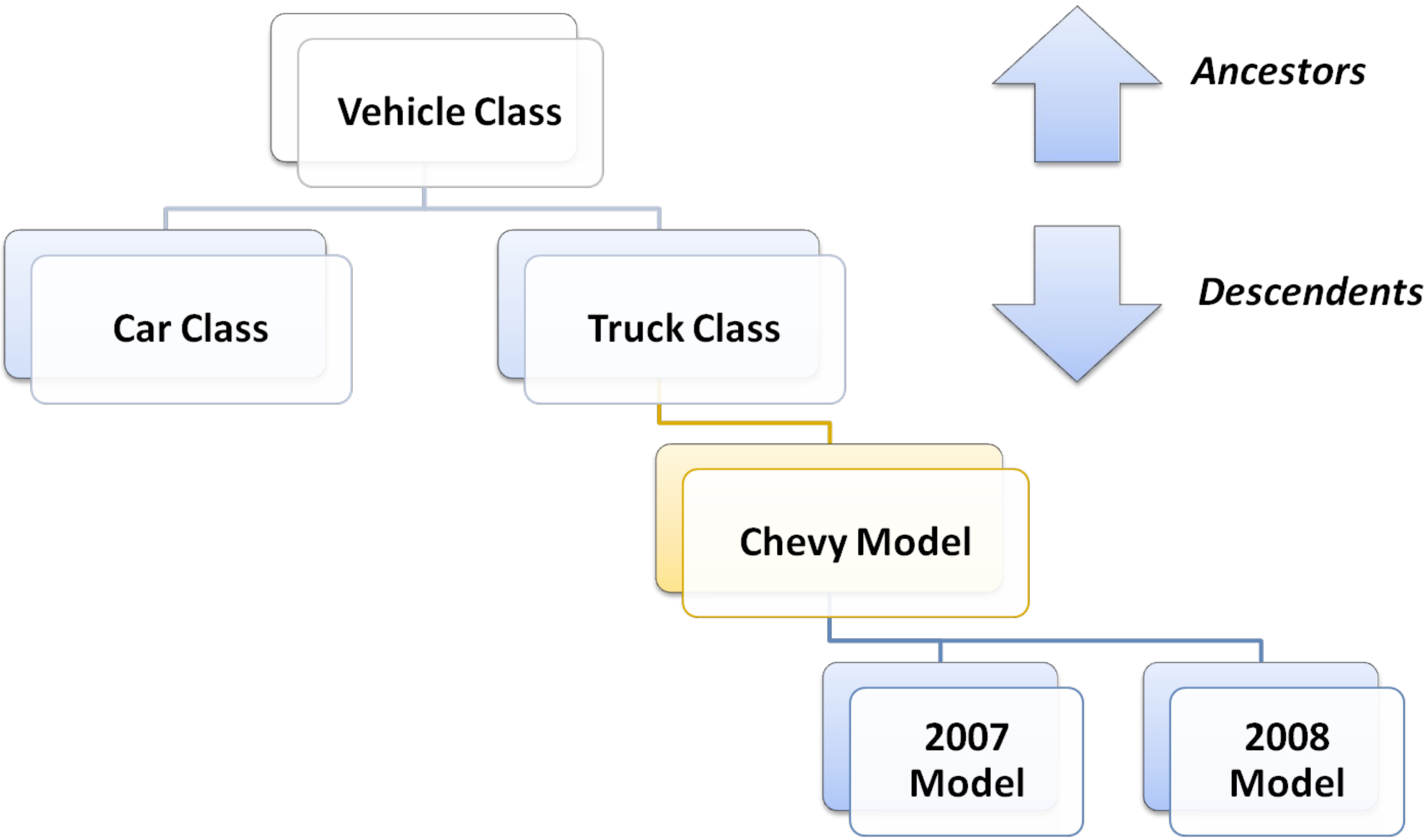
Example methods:

- Initialize
- Get Cargo Capacity
- Brake



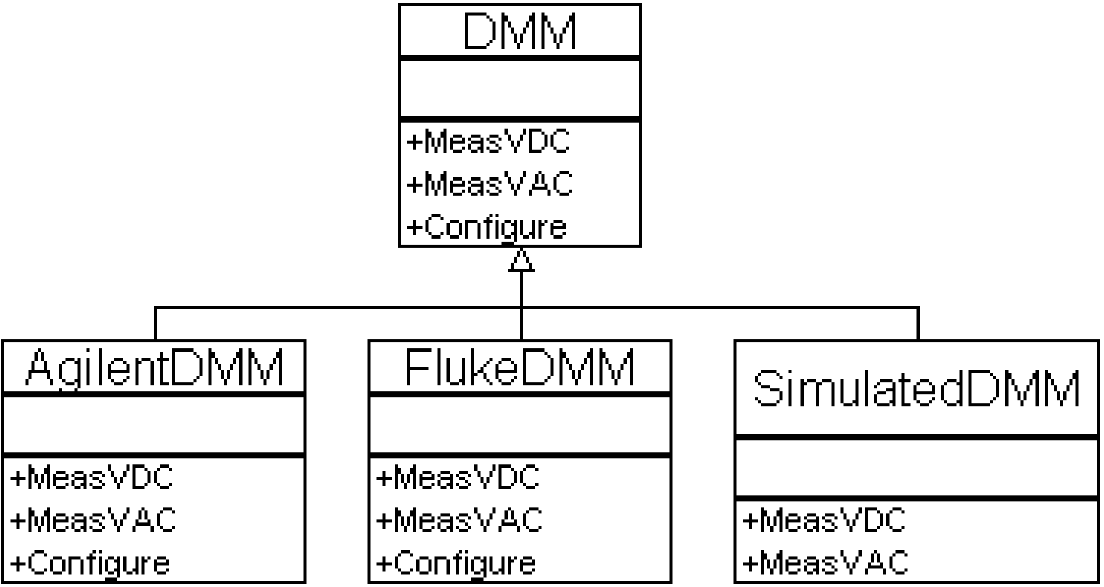
A car is a type of vehicle. *A truck is a type of vehicle.*

Inheritance example



Inheritance

- Creates replacability between classes which:
  - Inherit from the same ancestor
  - Have the same public VI's (methods)
- Benefits
  - Code reuse combined with specialization
  - Changes to parent propagate to children

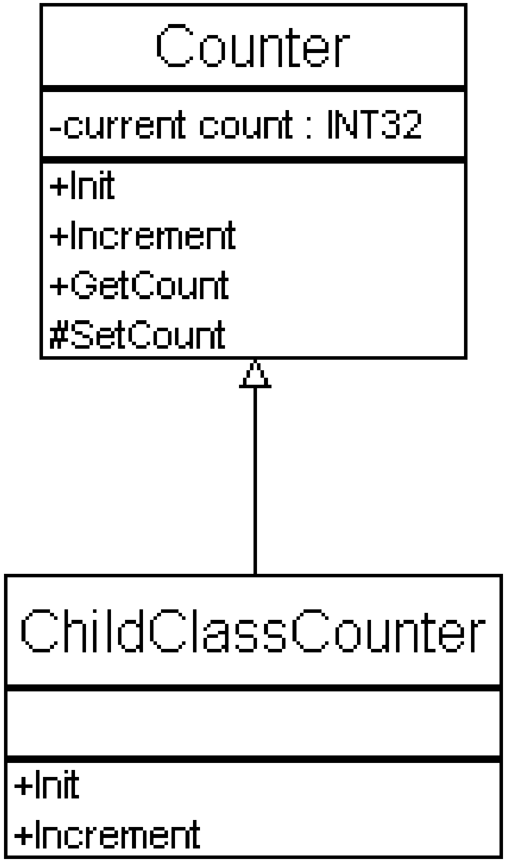


DEMO: Inheritance in LabVIEW

Init en Increment are “magic”
Dynamic dispatch VI's:

- Same VI name on each class
- Different block diagrams
- LabVIEW chooses which VI to run

DEMO: InheritanceApp.vi



Counter

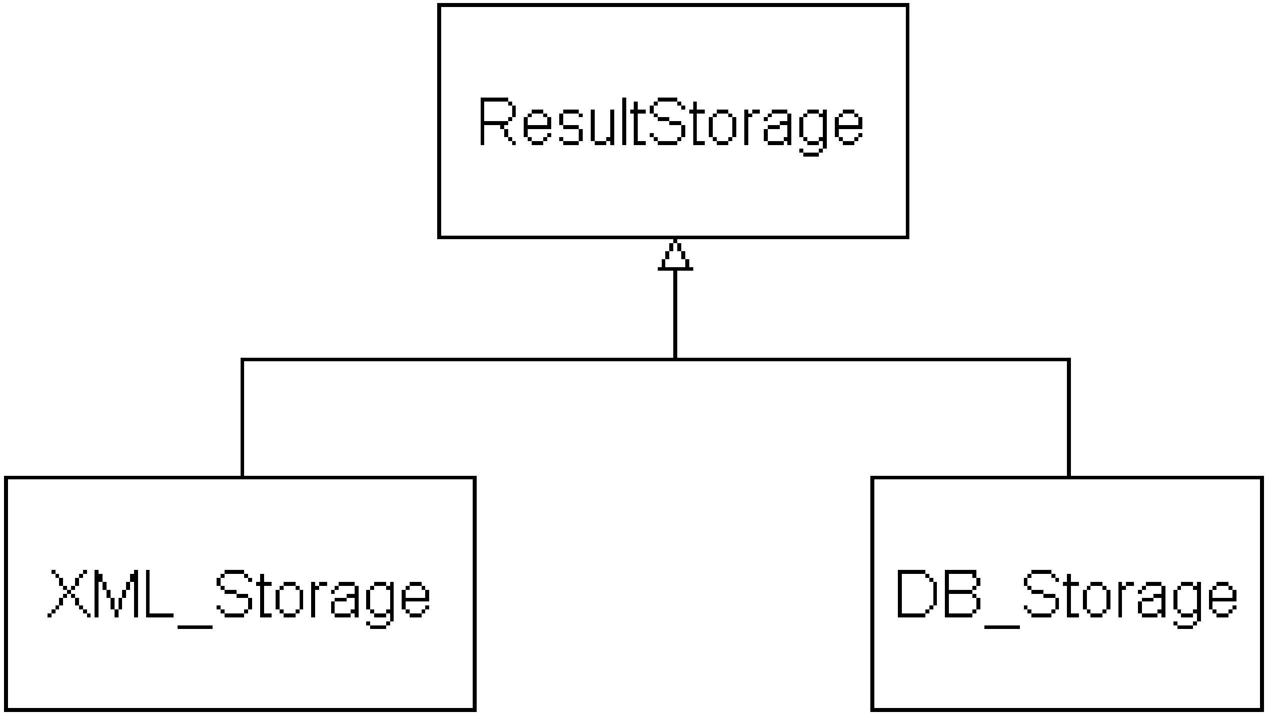
-current count : INT32

+Init
+Increment
+GetCount
#SetCount

ChildClassCounter

+Init
+Increment

Another example



Extension - GOOP

LabVIEW class + Reference

Instead of: Object in the wire
→ Reference in the wire

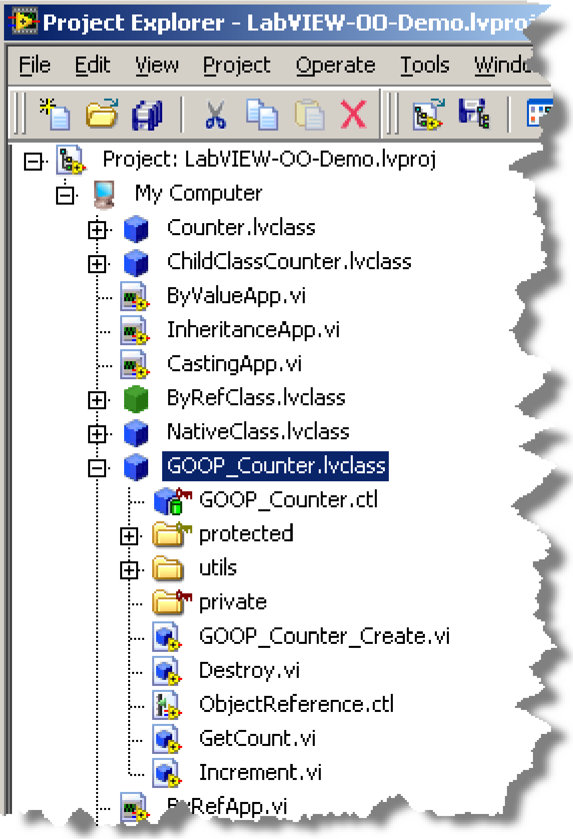
Gives us control of object creation and destruction

How?

- NI Example Finder → Fundamentals → Object-Oriented → ReferenceObject.lvproj
- 3rd Party reference frameworks and/or tooling

DEMO: GOOP

- ByRefApp.vi
- Creation of a GOOP class
- Explore the tools



Project Explorer - LabVIEW-OO-Demo.lvproj

```
Project: LabVIEW-OO-Demo.lvproj
  My Computer
    Counter.lvclass
    ChildClassCounter.lvclass
    ByValueApp.vi
    InheritanceApp.vi
    CastingApp.vi
    ByRefClass.lvclass
    NativeClass.lvclass
    GOOP_Counter.lvclass
      GOOP_Counter.ctl
      protected
      utils
      private
      GOOP_Counter_Create.vi
      Destroy.vi
      ObjectReference.ctl
      GetCount.vi
      Increment.vi
    ByRefApp.vi
```

Use Case Summary

- GOOP
  - Modeling of system resources / hardware
  - Parallel (R / W) access to object data
  - Tooling!
  - Object attributes (data) are protected instead of private
- LVOOP
  - Parallelle toegang tot data (zonder semaforen)
  - Dataflow (replacement of clusters)
  - Native dynamic dispatching

Resources and acknowledgments

LabVIEW Object-Oriented Programming FAQ
http://zone.ni.com/devzone/cda/tut/p/id/3573

Expressionflow – Blog by Tomi Maila
http://expressionflow.com/

GOOP on LAVA
http://forums.lavag.org/GOOP-f68.html

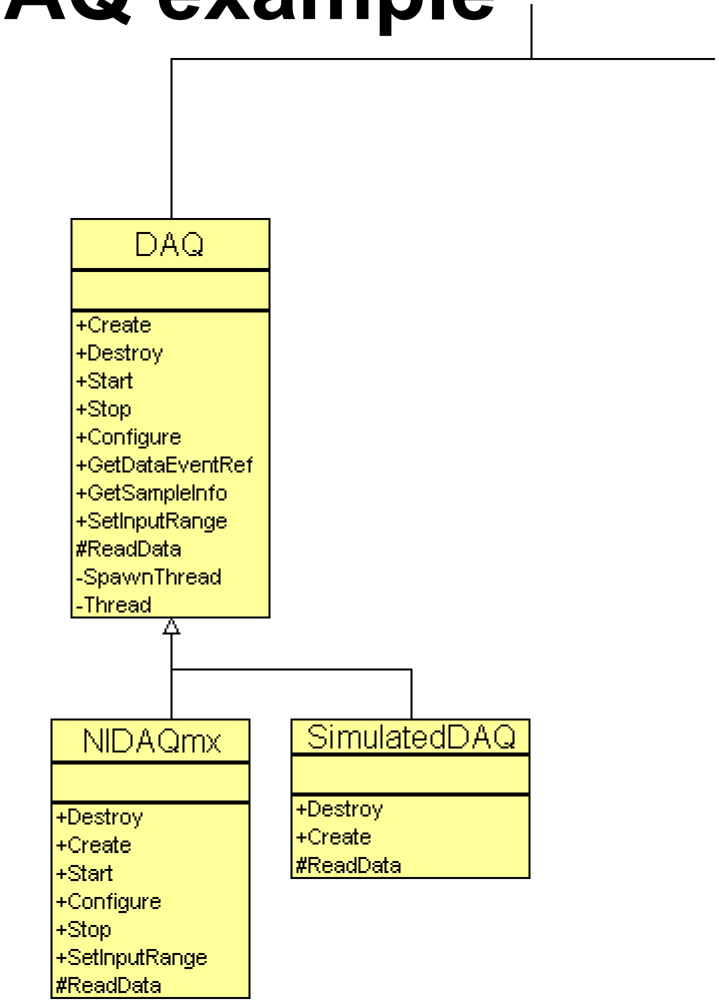
Endevo – Makers of Goop Development Suite and UML Modeller
http://www.endevo.se/content/blogcategory/18/103/lang,en/

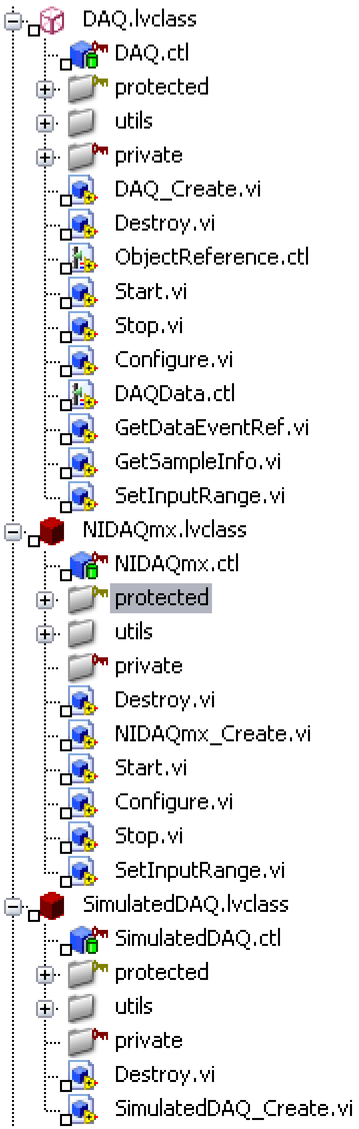
LabVIEW Examples – Fundamentals → Object-Oriented

VI Technologies (Training Graphical Object Oriented Programming 13/14-10-2008)
http://www.vi-tech.nl/

Stephen Mercer (LabVIEW R&D) – LabVIEW Classes:The State of the Art
http://forums.ni.com/ni/attachments/ni/170/353748/1/TS1304_Mercer_pptx.zip

Extra – DAQ example





- DAQ.lvclass
  - DAQ.ctl
  - protected
  - utils
  - private
  - DAQ_Create.vi
  - Destroy.vi
  - ObjectReference.ctl
  - Start.vi
  - Stop.vi
  - Configure.vi
  - DAQData.ctl
  - GetDataEventRef.vi
  - GetSampleInfo.vi
  - SetInputRange.vi
- NIDAQmx.lvclass
  - NIDAQmx.ctl
  - protected
  - utils
  - private
  - Destroy.vi
  - NIDAQmx_Create.vi
  - Start.vi
  - Configure.vi
  - Stop.vi
  - SetInputRange.vi
- SimulatedDAQ.lvclass
  - SimulatedDAQ.ctl
  - protected
  - utils
  - private
  - Destroy.vi
  - SimulatedDAQ_Create.vi

Extra - New Features

- LabVIEW 8.5
  - Choose Implementation dialog box
  - Create Accessor dialog box
  - Recursion!
- LabVIEW 8.6
  - Comparison functions work on classes
  - Better error reporting
  - List classes + dynamic members in VI-hierarchy
  - Un(flatten) XML support

FAQ: LabVIEW OOP Compared With C++

Q: How do LabVIEW classes compare with C/C++?

A: Some (but not all) of the differences include:

- LabVIEW has a value syntax only.
- C++ has constructors and destructors; LabVIEW has no need for them.
- C++ has multiple inheritance (LabVIEW does not).
- C++ has function overloading (LabVIEW does not).

FAQ: By-Value vs. By-Reference

Q: Why do LabVIEW classes use a by-value model instead of by-reference model?

A: By-value model is a better fit in a highly parallel programming environment. Examples:

- By-value avoids race conditions
- By-value allows the compiler to determine when copies of data need to be made

FAQ: Dynamic Dispatching Overhead

Q: Is there any overhead at run-time associated with dynamic dispatching?

A: Dynamic dispatching involves some small overhead as LabVIEW determines which subVI to invoke. The timing overhead is constant.