

Dr. Holger Brand

GSI Helmholtzzentrum für Schwerionenforschung GmbH

Workshop

**Objekt-orientiertes Programmieren mit LabVIEW**





[Creative Commons Lizenzvertrag](http://creativecommons.org/licenses/by-nc-sa/3.0/de/)  
Workshop: Objekt-orientiertes Programmieren mit LabVIEW von Dr. Holger Brand steht unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 3.0 Deutschland Lizenz](http://creativecommons.org/licenses/by-nc-sa/3.0/de/). Über diese Lizenz hinausgehende Erlaubnisse können Sie unter [tt-info@gsi.de](http://creativecommons.org/choose/tt-info@gsi.de) erhalten.

Inhaltsverzeichnis

[Übersicht 3](#_Toc337823833)

[Übung 1: Projekt entpacken und öffnen 4](#_Toc337823834)

[Übung 2: Bibliothek erstellen 5](#_Toc337823835)

[Übung 3: Klasse erstellen 6](#_Toc337823836)

[Klasse erstellen 6](#_Toc337823837)

[Vererbung einstellen 7](#_Toc337823838)

[Attribute hinzufügen 8](#_Toc337823839)

[Items zur ini-Datei hinzufügen 8](#_Toc337823840)

[Übung 4: Klassen-*Methoden erstellen* 9](#_Toc337823841)

[Zugriffsberechtigungen 9](#_Toc337823842)

[Übung 4.1 Datenzugriffs-Methoden 9](#_Toc337823843)

[Übung 4.2 Arbeiten mit LabVIEW-Objekten 10](#_Toc337823844)

[Übung 4.3 *Overwrite*-Methoden 10](#_Toc337823845)

[Übung 4.4: *Dynamic Dispatch*-Methoden 12](#_Toc337823846)

[Übung 5: Weitere *Interface*-Klassen 13](#_Toc337823847)

[Referenzen 14](#_Toc337823848)

[Notizen 15](#_Toc337823849)

[Notizen 16](#_Toc337823850)

[Notizen 18](#_Toc337823851)

# Übersicht

In diesem Workshop wird der VIP 2012 Vortrag *Einführung in die objektorientierte Programmierung mit LabVIEW* aus dem Bereich *LabVIEW Power Programming* durch Übungen vertieft.

Es handelt sich um die Anwendung von LabVIEW-Klassen zum Lesen und Schreiben von Konfigurationsdaten, die oft in *ini*-Dateien gespeichert werden. Im Rahmen dieses Projekts werden die kompletten Konfigurationsdaten in einer Instanz der Klasse *Configuration* gespeichert. Die Sektionen werden als Objekte betrachtet, die durch LabVIEW-Klassen beschrieben werden, die von der Klasse *Section* abgeleitet sind. Das Speichermedium wird als Interface-Basisklasse abstrahiert. Abgeleitete Klassen implementieren den konkreten Zugriff auf das Speichermedium. Für den Fall der *ini*-Dateien sind die notwendigen Klassen und VIs bereits ausprogrammiert. Weitere *Interface*-Klassen, z.B. für den Zugriff auf XML-Dateien oder Datenbanken können bei Bedarf hinzugefügt werden.

Sie werden sich in diesem Workshop zunächst mit der Klassenbibliothek vertraut machen. Abbildung 1 zeigt das Projekt und die Klassenhierarchie. Danach werden Sie eine neue Klasse für Ihre eigene neue Sektion erstellen. Dabei werden Sie lernen, wie man in LabVIEW eine neue Klasse erstellt. Sie werden sie als Kind-Klasse der *Section*-Basisklasse definieren. Danach werden Sie Ihrer Klasse Attribute hinzufügen, in der die *Item*-Daten gespeichert werden. Anschließend werden Sie die *dynamic dispatch*-VIs der Basisklasse überschreiben, um mit Hilfe der *Interface*-Klasse Ihre neuen *Items* zu lesen und zu schreiben. Am Ende steht die Herausforderung, eine neue Funktion mit Hilfe eines *dynamic dispatch*-VIs und entsprechender *overwrite*-VIs zu realisieren.

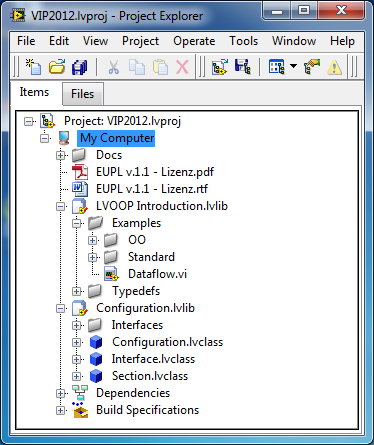
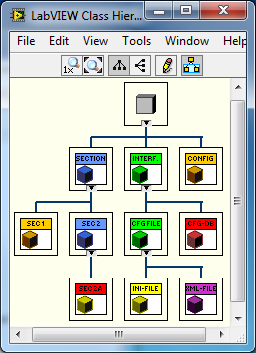
 

Abbildung 1: Projekt & Klassenhierarchie

# Übung 1: Projekt entpacken und öffnen

Im Folgenden werden Sie die Projektdateien entpacken (falls nicht schon vorinstalliert) und das enthaltene LabVIEW Projekt öffnen.

1. Wählen Sie im Kontextmenü der Datei *VIP 2012 LVOOP Workshop.zip*   
   → Dateien entpacken und wählen ein neues Verzeichnis *<Projektverzeichnis>[[1]](#footnote-1)*.
2. Wechseln Sie in das neue Verzeichnis *<Projektverzeichnis>\VIP2012*
3. Doppelklicken Sie auf *VIP2012.lvproj* um das Projekt zu öffnen.
   1. Alternativ starten Sie LabVIEW 2012  
      *Windows Start→Alle Programme→National Instruments LabVIEW 2012*
   2. Wählen Sie Menü*→File→Open Project…*
   3. Navigieren Sie zu der Projektdatei *VIP2012.lvproj* und klicken Sie OK, um das Projekt zu öffnen.
4. Das Projekt sollte einige Dokumente enthalten sowie die Lizenzdateien und natürlich die LabVIEW Bibliotheken, die die Klassen und Anwendungsbeispiele enthält.
   1. *Doc*  
      Vortrag und diese Anleitung
   2. *Configuration.lvlib*  
      Diese LabVIEW Bibliothek enthält die notwendigen Basisklassen für dieses Projekt
      1. *Configuration.lvclass*, Diese Klasse enthält ein Interface und nach dem Lesen die kompletten Konfigurationsdaten.
      2. *Interface.lvclass* und ihre Kind-Klassen, insbesondere die bereits fertig gestellte abgeleitete Klasse *Ini-File.lvclass* die den Zugriff auf eine ini-Datei implementiert.
      3. *Section.lvclass*, die Basisklasse für alle Anwendungsspezifischen Kind-Klassen.
   3. *LVOOP Introduction.lvlib*  
      Diese LabVIEW Bibliothek enthält die Beispiel-VI’s, Beispielklassen und Beispiele für ini-Dateien.
      1. Das Lesen von Konfigurationsdaten mit Hilfe der Standard-VIs und Typdefinitionen finden Sie in dem virtuellen Verzeichnis: *Examples\Standard*
      2. Das Beispiel zum Lesen von Konfigurationsdaten mit dem objekt-orientierten Ansatz finden Sie in dem virtuellen Verzeichnis: *Examples\OO*. Dort sind auch die auf das Beispiel bezogenen Kind-Klassen von *Section.lvclass* zu finden.
         1. *Section 1.lvclass*
         2. *Section 2.lvclass*
            1. *Section 2a.lvclass* eine Kind-Klasse von *Section 2.lvclass*.
   4. *Oracle\_Configuration.lvlib* wird in diesem Workshop nicht benutzt. Sie enthält einen Tabellenentwurf mit Beispielkonfigurationsdaten für eine Oracle-Datenbank.
5. In den ini-Dateien müssen die *ClassPath*-Einträge auf die aktuellen Pfade korrigiert werden.
6. Führen Sie die Beispiele aus, um sich mit der Problemstellung vertraut zu machen. Sie können die entsprechende ini-Datei mit *Drag&Drop* aus dem Projekt auf die Pfad-Kontrolle ziehen. Bemerken Sie die Unterschiede?

# Übung 2: Bibliothek erstellen

Die Ergebnisse Ihrer eigenen Übungen in diesem Workshop speichern Sie am besten in einem neuen Verzeichnis, z.B. *<Projektverzeichnis>\ihrName*.

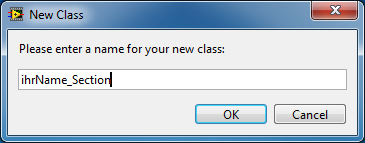
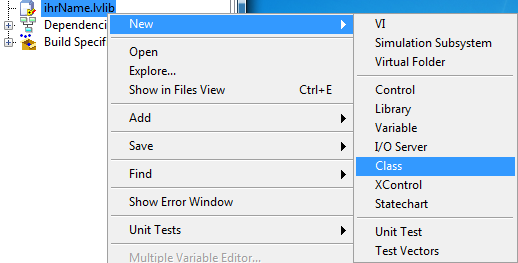
Am Ende des Workshops können Sie das komplette Verzeichnis auf einen USB-Stick kopieren und mit nach Hause nehmen, als Basis für weitere Übungen oder zur Wiederverwendung bei eigenen Projekten.

1. Erstellen Sie in dem Projekt eine neue Bibliothek.  
   (Kontextmenü von *My Computer>New>Library*)
2. Speichern Sie diese Bibliothek *<Projektverzeichnis>\ihrName\ihrName.lvlib*  
   (Kontextmenü von *Untitled Library 1>Save>Save as…*)
3. Öffnen Sie die Eigenschaften der Bibliothek  
   (Kontextmenü von *ihrName.lvlib>Properties*)
   1. Versehen Sie in der Kategorie *General Settings* das *VI Icon Template* mit Ihren Initialen. Dieses Template wird als Basis für neue VIs benutzt.
   2. Geben Sie in der Kategorie *Documentation* eine kurze Beschreibung ein.
   3. Schließen Sie den Dialog und speichern Sie das Projekt

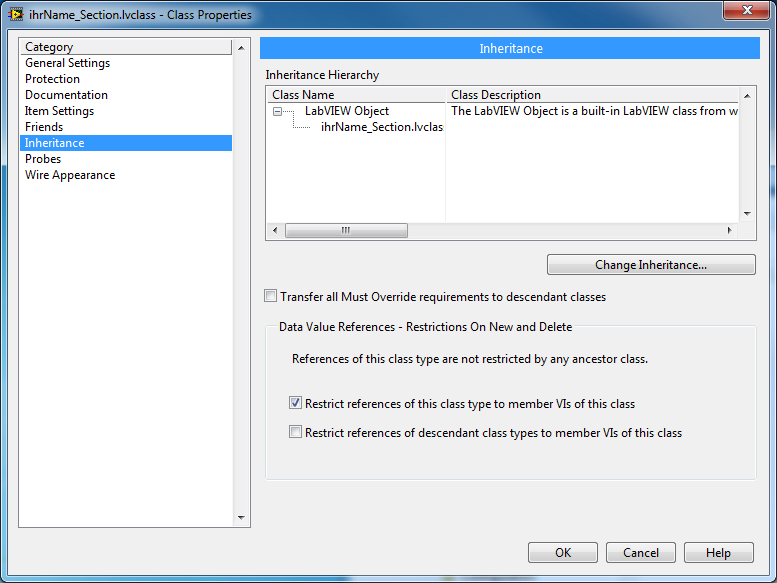
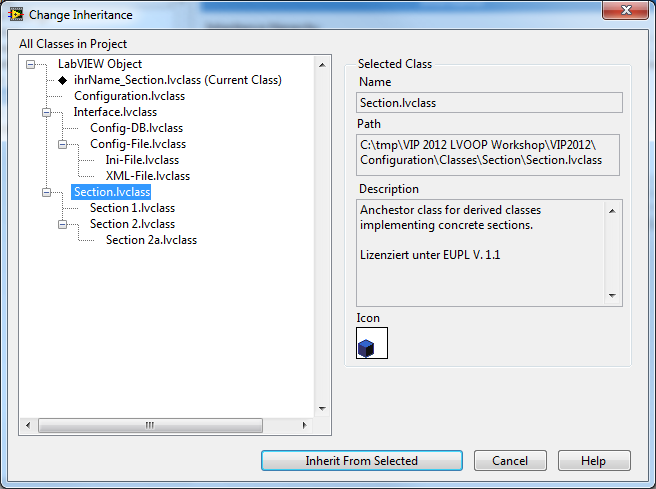
# Übung 3: Klasse erstellen

In dieser Übung werden Sie Ihre eigene Klasse erstellen. Sie soll eine eigene neue Sektion enthalten, also von *Section.lvclass* erben. Sie werden die notwendigen Attribute zu *Class private data* hinzufügen und die *dynamic dispatch*-VI‘s der Basisklasse überschreiben, um Ihre eigenen Items zu lesen bzw. zu schreiben. Sie werden die Konfigurationsdatei um die neue Sektion und ihre *Items* erweitern und mit dem vorhandenen Beispiel lesen.

## Klasse erstellen

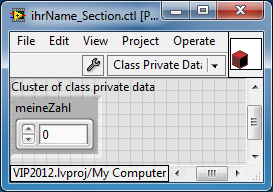
1. Erstellen Sie in der Bibliothek eine neuen Klasse *ihrName\_Section.lvclass*.  
   (Kontextmenü von *ihrName.lvlib>New>Class*) 
2. Öffnen Sie die Eigenschaften der Klasse  
   (Kontextmenü von *ihrName.lvclass>Properties*)
   1. Versehen Sie in der Kategorie *General Settings* das *VI Icon Template* mit Ihren Initialen. Dieses Template wird als Basis für neue VI‘s benutzt.
   2. Geben Sie in der Kategorie *Documentation* eine kurze Beschreibung ein.
   3. Schließen Sie den Dialog und speichern Sie das Projekt.

## Vererbung einstellen

1. Öffnen Sie die Eigenschaften der Klasse erneut.
2. In der Kategorie *Inheritance* sehen Sie die Vererbungshierarchie Ihrer Klasse. Eine neue Klasse hat immer von *LabVIEW Object* geerbt. 
   1. Lassen Sie vorerst alle Optionen unverändert. Diese werden in diesen einführenden Workshop nicht weiter behandelt.
   2. Ändern Sie die Vererbung, indem Sie den Knopf *Change Inheritance* anklicken. Wählen Sie in dem Dialog die gewünschte Eltern-Klasse aus, in unserem Beispiel ist das *Section.lvclass*. Klicken Sie *Inherit From Selected* um die Vererbungshierarchie zu ändern.
      1. Schließen Sie den Dialog und speichern Sie das Projekt.
3. Öffnen Sie die LabVIEW Class Hierarchie (*Menü>View*). Sie finden Ihre Klasse im Vererbungsbaum jetzt unterhalb der Klasse *Section.lvclass*.

## Attribute hinzufügen

Erweitern Sie Ihre Klasse und öffnen Sie das *ihrName\_Section.ctl*. (Es entspricht der Typedefinition beim nicht objekt-orientierten Programmieren.)

1. Fügen Sie dem *Cluster of class private data* die von Ihnen gewünschten Attribute von der *Controls*-Palette hinzu und vergeben einen sinvollen Name, z.B. ein *Numeric* Control vom Typ *Double* mit dem Namen *meineZahl*. 
2. Sie sollten auch die *Description* und *Tip* für alle Attribute ausfüllen.
3. Sie können auch die *VI-Description* in den *Control Properties* ausfüllen.
4. Schließen Sie die *Control* und speichern Sie das Projekt.

## Items zur ini-Datei hinzufügen

Öffnen Sie *LVOOP Introduction.lvlib:Examples\OO\Demo\_OO.ini* und speichern Sie eine Kopie in Ihr eigenes Verzeichnis, z.B. *<Projektverzeichnis>\ihrName\ihrName.ini*.

1. Fügen Sie eine neue Sektion hinzu.
2. Geben Sie den *ClassPath* zu Ihrer neuen Klasse an.
3. Fügen Sie für jedes Attribut in Ihrem *class private data* ein Item mit Wertzuweisung hinzu.
4. Das Ergebnis könnte so aussehen:  
   [ihrName\_Section\_0]  
   ClassPath="<Projektverzeichnis>\ihrName\ihrName\_Section\ihrName\_Section.lvclass"  
   meineZahl=987.654  
   [ihrName\_Section\_1]  
   ClassPath="<Projektverzeichnis>\ihrName\ihrName\_Section\ihrName\_Section.lvclass"  
   meineZahl=876.543
5. Speichern und schließen Sie die Datei.
6. Fügen Sie die neue Datei Ihrer Bibliothek als Beispiel hinzu und speichern Sie das Projekt.

# Übung 4: Klassen-*Methoden erstellen*

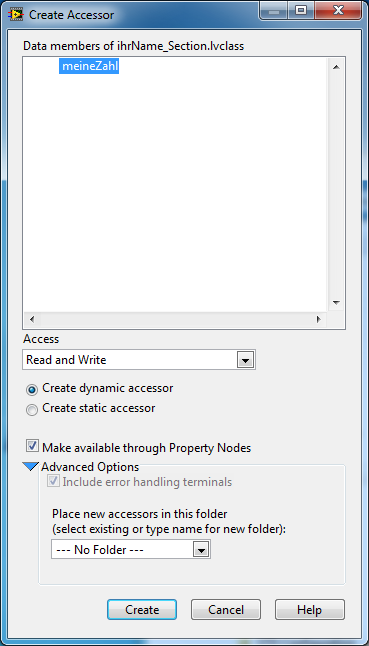
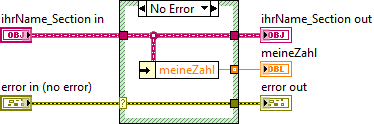
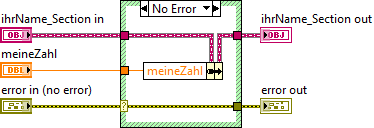
## Zugriffsberechtigungen

Da die **Attribute einer Klasse immer als privat deklariert sind**, kann man in VI‘s, die nicht der Klasse angehören, nicht mit *Un-/Bundle By Name* auf die Elemente des *Cluster of class private data* zugreifen. Das aufrufende VI wäre nicht ausführbar. Daher muss die Klasse gegebenenfalls Datenzugriffs-VI‘s bereitstellen, die mit der gewünschten Kategorie der Zugriffsberechtigung versehen werden. Die Zugriffsrechte können natürlich auch für alle anderen VI‘s der Klasse eingestellt[[2]](#footnote-2) werden.

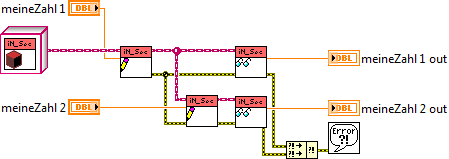
* *Public*: Kann von allen VIs benutzt werden
* *Protected*: Kann von Kind-Klassen und der Klasse selbst benutzt werden.
* *Private*: Kann nur von der Klasse selbst benutzt werden.
* *Community*: Kann von der Klasse selbst und von Ihren Freunden[[3]](#footnote-3) benutzt werden.

## Übung 4.1 Datenzugriffs-Methoden

Fügen Sie die VI‘s für das Lesen bzw. Schreibe eines Attributes Ihrer Klasse hinzu.

1. Wählen Sie im Kontextmenü der Klasse *New>VI for Data Member Access…*
   1. Wählen Sie im Dialog die gewünschten Attribute[[4]](#footnote-4) aus, Beispiel *meineZahl*.
   2. Wählen Sie die Zugriffsmethode aus: *Read and Write*.
   3. Behalten Sie die anderen Optionen unverändert bei.
   4. Klicken Sie Create, damit der Wizard die entsprechenden VIs erzeugt.
   5. Versehen Sie die neuen VIs mit Icon und Beschreibungen für VI und *Frontpanel*-Elemente.
   6. Speichern Sie die VIs mit dem voreingestellten Namen.
   7. Die Blockdiagramme der erzeugten VI‘s sehen in etwa so aus:   
      

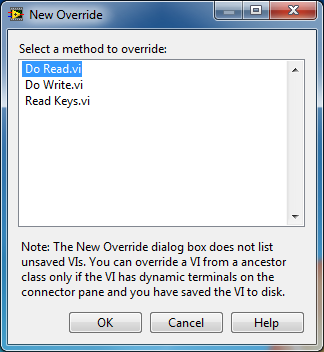
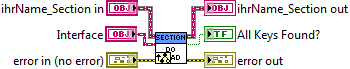
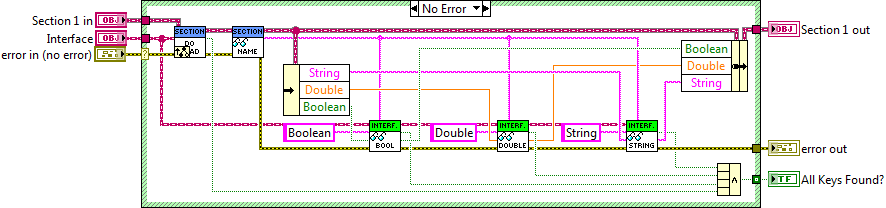
## Übung 4.2 Arbeiten mit LabVIEW-Objekten

1. Erstellen Sie ein normales VI in Ihrer Bibliothek, z.B. *Test\_DataAccess.vi*
2. Ziehen Sie Ihre Klasse aus dem Projekt auf das Blockdiagramm des neuen VI‘s
3. Ziehen Sie die beiden Datenzugriffs-VIs auf das Blockdiagramm.
4. Verdrahten Sie die VI, so dass Sie zuerst einen Werte in das Objekt schreiben und danach lesen.
5. Erzeugen Sie die zugehörigen Kontrollen und Indikatoren auf dem Frontpanel und verdrahten Sie diese mit den SubVI‘s.
6. Das VI könnte jetzt so aussehen: 
7. Stellen Sie auf dem Frontpanel ein Zahl ein und starten Sie das VI. Beobachten Sie das Ergebnis.
8. Erweitern Sie das VI mit einem abzweigenden Objekt und setzen Sie einen anderen Wert.   
   Beachten Sie, dass das Objekt am Drahtabzweig geklont wird.
9. Stellen Sie auf dem Frontpanel verschiedene Zahlen ein und starten Sie das VI. Beobachten Sie das Ergebnis.
10. Beachten Sie die Auswirkung des Datenflussprinzips!
11. Variieren Sie das VI. Arbeiten Sie z.B. mit zwei Schleifen.
12. Stellen Sie bei Bedarf Fragen.

## Übung 4.3 *Overwrite*-Methoden

Fügen Sie die *Overwrite*-VI’s, die von *Section.lvclass* als *dynamic dispatch*-VI’s definiert wurden, Ihrer Klasse hinzu.

* *Do Read.vi* liest Daten vom Speichermedium (Interface) und speichert Sie in den Objektattributen.
* *Do Write.vi* Schreibt Daten, die in den Objektattributen gespeichert sind, in das Speichermedium (Interface).
* *Read Keys.vi* konvertiert die Objektattribute in Strings und fügt Sie zusammen mit dem Key-Namen an eine Array an.

1. Wählen Sie im Kontextmenü der Klasse *New>VI for Overwrite…*
2. Selektieren Sie *Do Read.vi* und klicken Sie den OK-Knopf.  
    
   1. Der *Wizard* erstellt ein *Overwrite*-VI als Startpunkt für Ihre Implementierung, das nichts anderes tut, als die entsprechende Methode der *Ancestor*-Klasse aufzurufen, z.B. Do Read.vi.  
      
   2. Benutzen Sie ihre eigenen Klassen-VI’s und die der Elternklasse, sowie die Methoden der Basisklasse *Interface*, um Ihre Items zu lesen.
   3. Bei der Implementierung können Sie sich an den Klassen in der *LVOOP Introduction.lvlib* orientieren, z.B.:  
      
3. Wiederholen Sie die Schritte 1 und 2 auch für *Do Write.vi* und *Read Keys.vi*.
4. Führen Sie nun das erneut *Read Configuration OO.vi* aus der *LVOOP Introduction.lvlib* aus. Sie sollten nun Ihre eigenen Sektionen in dem angezeigten *Array* wiederfinden.
5. Speichern Sie eine Kopie des VI’s und modifizieren Sie es entsprechend Ihrer Wünsche, z.B. ändern des Wertes eines *Items* und schreiben Sie die Konfiguration in eine neue ini-Datei.

## Übung 4.4: *Dynamic Dispatch*-Methoden

Erstellen Sie analog zu der Methode *Configuration.lvclass:Read Keys.vi* bzw. *Section.lvclass:Read Keys.vi* die notwendigen Methoden, um alle *Items* vom Type *Double* als *Double* auszulesen.

**Hinweis**: Dazu erstellen Sie ein *static dispatch*-VI in *Configuration.lvclass* und ein *dynamic dispatch*-VI in *Section.lvcass* sowie die notwendigen *overwrite*-VI’s in den Kindklassen.

# Übung 5: Weitere *Interface*-Klassen

Als **Hausaufgabe** können Sie für Ihren Bedarf Kind-Klassen von *Interface.lvclass* erstellen um Konfigurationsdaten in anderen Datei-basierten Formaten, z.B. XML (*eXtensible Markup Language*), oder in Datenbanken zu speichern. Entsprechende Basisklassen sind in der *Configuration.lvlib* bereits angelegt.

* *Config-DB.lvclass*
* *Conig-File.lvclass*
  + *XML-File.lvclass*

Da es im Rahmen von XML verschiedenste Document Type Definitions (DTD) gibt, sollten Sie ihre eigene *XML-Interface*-Klasse als Kind von *XML-File.lvclass* erstellen.

Da es die verschiedensten Datenbanken gibt, sollten Sie ihre eigene Datenbank*-Interface*-Klasse als Kind von *Config-DB.lvclass* erstellen. Als Startpunkt für einen Tabellenentwurf gibt es in *Oracle\_Configuration.lvlib* ein SQL-Skript bzw. Hilfs-VI’s, um in einer Oracle-Datenbank Beispieltabellen zu erzeugen, Daten einzufügen und wieder zu löschen. Diese könnten als Startpunkt für Ihre eigene Konfigurationsdatenbank dienen. Diese Bibliothek enthält allerdings keine Implementierung einer entsprechenden *Interface*-Kind-Klasse.

# Referenzen

Nachfolgend finden Sie einige Hinweise auf Startpunkte für das Objekt-orientierte Programmieren in LabVIEW.

* LabVIEW Menue>Help>LabVIEW Help... -> Contents -> Fundamentals -> LabVIEW Object-Oriented Programming
* LabVIEW Menue>Help>Find Examples -> Browse by Task -> Fundamentals -> Object-Oriented
* [LabVIEW Object-Oriented Programming: The Decisions Behind the Design](http://zone.ni.com/devzone/cda/tut/p/id/3574)
* [LabVIEW Object-Oriented Programming FAQ](http://zone.ni.com/devzone/cda/tut/p/id/3573)
* [Applying Common OO Design Patterns to LabVIEW](http://decibel.ni.com/content/docs/DOC-2875)
* [HGF Baseclass Library](http://wiki.gsi.de/cgi-bin/view/NIUser/HGFBaseClassLibrary)
* [Mobile Agent System](http://wiki.gsi.de/cgi-bin/view/NIUser/LVMobileAgentSystem)
* [Actor Framework](https://decibel.ni.com/content/docs/DOC-17193)
* [Measurement Abstraction and Model-View-Controller (MVC) Project with Actor Framework in LabVIEW](https://decibel.ni.com/content/docs/DOC-21441)

# Notizen

# Notizen

# Notizen

1. Das konkret zu benutzende Projektverzeichnis wird zu Beginn des Workshops bekanntgegeben. [↑](#footnote-ref-1)
2. *Access Scope* im Kontextmenü eines Elements einer LabVIEW Bibliothek oder Klasse. [↑](#footnote-ref-2)
3. Die Freunde müssen in den Klasseneigenschaften explizit angegeben werden. [↑](#footnote-ref-3)
4. In jedem erzeugten VI wird jeweils nur ein Element benutzt. Wenn Sie in einem VI mehr als ein Element lesen oder schreiben wollen. Selektieren Sie nur ein Attribut-Element und modifizieren das VI manuell. [↑](#footnote-ref-4)