RainingData

Objektorientierte Programmierung mit Omnis Studio

von Richard Mortimer
Technical Consultant
richard.mortimer@rainingdata.com

RainingData

Themen

- Was ist Objektorientierung?
- Die Prinzipien der Objektorientierung
- Objektorientierung in Omnis Studio
- Benutzen von Vererbung und Subklassen
- Methoden und Nachrichten
- Benutzen von Subwindows
- Benutzen von Objektklassen

RainingData

Strukturiertes Programmieren

- Top-down Verfahren
- Daten und Funktionen sind separat gehalten
- Teilt ein Programm in Komponenten auf, bis die Komponenten nicht mehr weiter geteilt werden können.
- Verbesserte die Qualität der Software
- Nachträgliche Änderungen im Design, nachdem die Programmierung bereits gestartet hat, verursacht möglicherweise eine komplette Neustruktur der Applikation.

RainingData

Was ist Objektorientierung?

- Die Applikation basiert auf Softwareobjekte
- Diese simulieren Objekte aus der realen Welt.
- Es ist ein anderer Weg Software Systeme zu bauen.
- Es ist ein Paradigmenwechsel.
- Es ist eine Technologie.
- Viel wichtiger: Es ist eine Methodik.

RainingData

Ein wenig OO Geschichte

- Erfunden in den späten 1960'ern in in einer Sprache namens Simula von Ole-Johan Dahl und Kristen Nygaard des Norwegischen Computer Centers in Oslo
- Anfang 1970 unterstützte die Sprache SmallTalk von Alan Kay des Xerox PARC die Idee, Softwareobjekte zu benutzen, die reale Objekte simulieren und diese bei der Entwicklung und beim Prototype von Applikationen zu verwenden.
- Mitte der 1980'er erschienen andere OO Sprachen wie C++ und Eiffel
- Ende 1990 wurde Java und *Omnis Studio* entwickelt.

RainingData

Objekte

- Objekte sind "black boxes" die miteinander kommunizieren um Aufgaben auszuführen.
- Objekte kombinieren sowohl Zustände - "state" (z.B. Daten) und Verhalten - "behavior" (z.B. Proceduren oder "Methoden") in einer einzigen Entität.
- Das beschleunigt die Entwicklung neuer Programme und , und erhöht die
  - Konsistenz
  - Wartbarkeit
  - Wiederverwendbarkeit

RainingData

Prinzipien der Objektorientierung

- Abstraktion
- Klassen
- Kapselung
- Instanzen
- Nachrichtenversand
- Polymorphismus
- Vererbung

RainingData

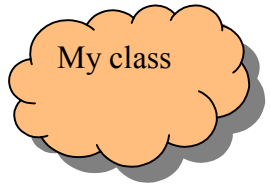
Abstraktion

- Abstraktion ist die Möglichkeit einer Sprache ein Konzept aufzunehmen und daraus in einem Programm eine vereinfachte Repräsentation zu erstellen .
- Es beinhalted die Identifizierung oder das Vereinfachen allgemeiner Funktionen von Objekten und Prozeduren um sie dann in einer einzigen, darstellbaren Entität zu kombinieren.
- Zum Beispiel, ein Programmierer würde Abstraktion benutzen um herauszustellen, das zwei Funktionen überwiegend dieselbe Aufgabe ausführen und somit in eine einzige Funktion integriert werden kann.
- Jedes Objekt im System dient als Modell eines vereinfachten “Actor” welches Aufgaben ausführen, Berichte abliefern und Zustände ändern und mit anderen Objekten im System kommunizieren kann.
- Ein Kundenobjekt z.B. ist eine abstrakte Repräsentation eines wirklichen Kunden.

RainingData

Klassen

- Sie könnten sich eine Klasse als eine Fabrik vorstellen, die genau eine Art von Objekt produzieren kann.
- Ein Objekt wird durch die Klasse definiert, die Alles über das Objekt festlegt.
- Klassen beinhalten die Spezifikationen für das Verhalten und die Eigenschaften des Objektes.
- Die Klasse ist eine Abstraktion einer Einheit aus der richtigen Welt.



RainingData

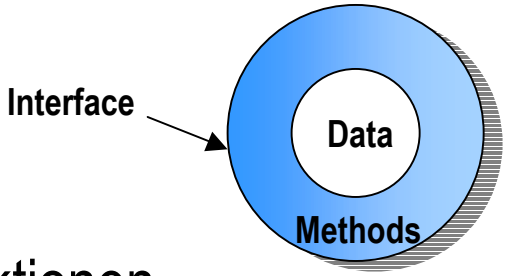
Instanzen

- Objekte sind individuelle Instanzen einer Klasse
- Jede Instanz hat einen eindeutigen Identifier
- Zum Beispiel: Sie könnten ein Objekt namens “Fanny” der Klasse “Hund” erstellen.
- Objektorientierte Sprachen haben eine Funktionalität um aus einer Klassendefinition eine Objekt Instanz zu erzeugen (Instanziierung).
- Sie könnten mehr als ein Objekt von einer Hundeklasse erzeugen und Ihnen unterschiedliche Namen geben.
- Instanzen kommunizieren untereinander mit Nachrichtenversand (Messaging)

RainingData

Kapselung

- Ein Objekt hat ein externes Interface und Internas, die nicht sichtbar sind von aussen.
- Diese versteckten Informationen und Funktionen nennt man Kapselung
- Versteckte Informationen und Funktionen
  - Daten (Variable)
  - Private Methoden
- Versteckte Daten sind geschützt
- nicht versteckte Informationen und Funktionen
  - Öffentliche Methoden
- Die Internas können verändert werden ohne die Aussenwelt zu beeinflussen.



RainingData

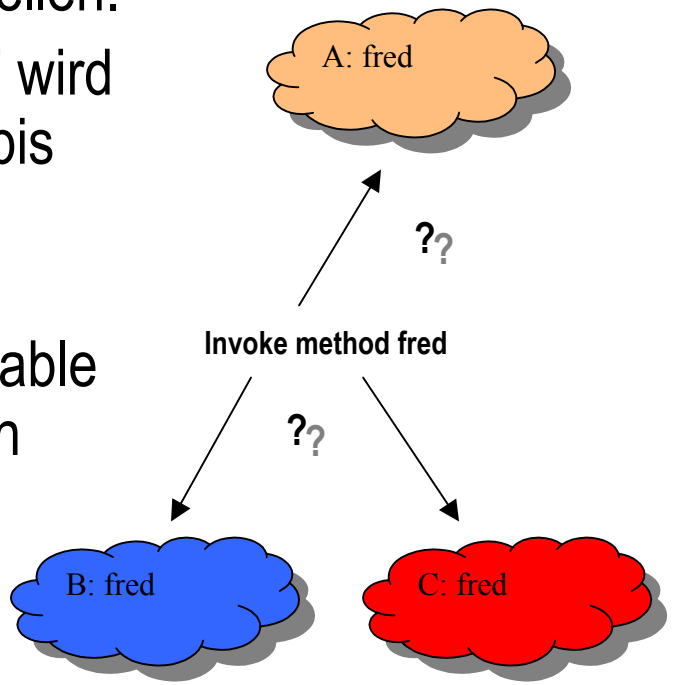
Nachrichtenversand

- Instanzen (Objekte) kommunizieren miteinander via Nachrichtenversand
- Die zulässigen Nachrichten sind dabei in der (Empfänger) Klasse definiert
- Wenn eine Instanz (Objekt) eine Nachricht empfängt, die es versteht, wird sie ausgeführt.
- Z.B: Die Hundeklasse definiert, was ein Hundeobjekt ist, so daß das Hundeobjekt versteht und auf Nachrichten wie “Sitz!”, “Platz!” usw. reagieren kann.
- Die Nachricht enthält:
  - Den Namen der Operation (Methodenname)
  - Jegliche zusätzliche Information, die die Operation benötigt (Parameter)
- Eine Operation kann Daten zurückliefern
  - Das ist der Weg um auf gekapselte Daten zuzugreifen.

RainingData

Late Binding

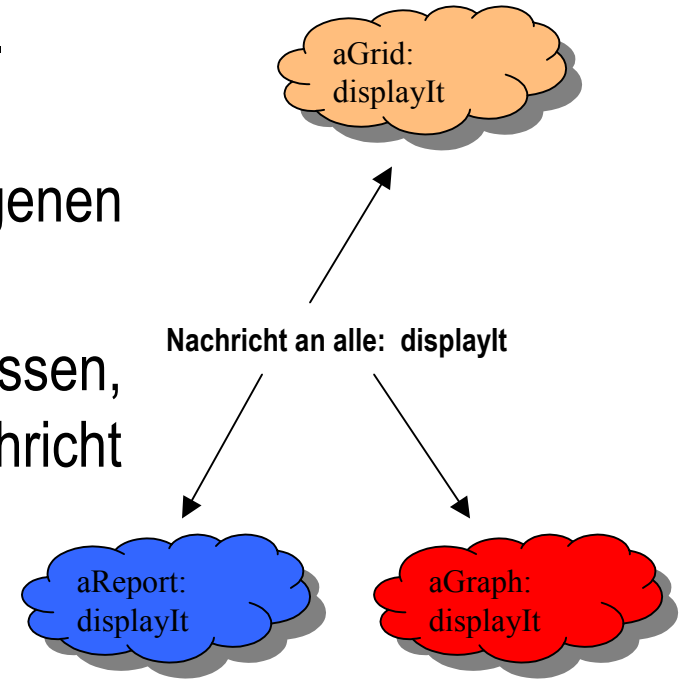
- Traditionelle Sprachen nutzen “Static binding” um eine Referenz zu einer bestimmten Variablen oder Methode zur Design (compile) Zeit herzustellen.
- Mit “Dynamic” oder “Late binding” wird die Entscheidung aufgeschoben bis zur Laufzeit des Programmes.
- Oftmals ist es unmöglich zur Designzeit zu sagen, welche Variable oder Methode zur Laufzeit wirklich benutzt wird.



RainingData

Polymorphismus

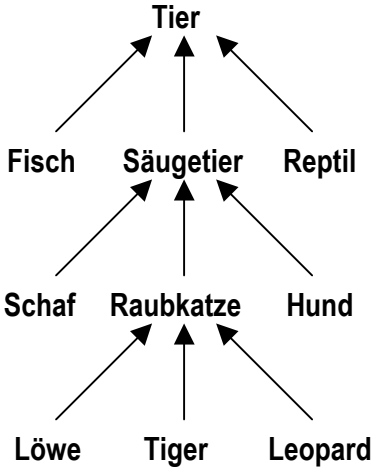
- Jede Klasse eines Objektes, das auf eine Nachricht reagiert, hat seine eigene Implementation einer Methode mit einem gemeinsamen Namen.
- Wenn die Methode gerufen wird, reagiert das Objekt mit seiner eigenen Implementation seiner Methode.
- Das rufende Objekt muß nicht wissen, welche Art von Objekt seine Nachricht empfängt.

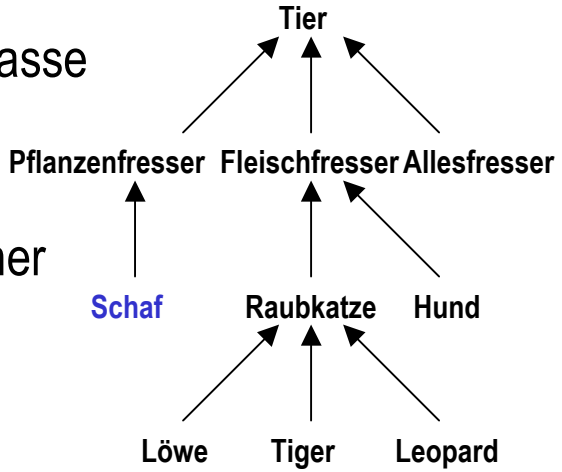


RainingData

Vererbung

- Die Möglichkeit eine Klasse als eine Spezialisierung einer anderen Klasse zu definieren
- Vererbung ist hirarchisch
- Subklassen vererben Eigenschaften von Superklassen
- Manchmal werden Ausdrücke wie “Abgeleitete Klasse” und “Basisklasse” anstatt “Subklasse” und “Superklasse” benutzt.
- Die Beziehung zwischen Subklasse und Superklasse ist “eine Art von”
- Z.B: Ein “Tiger” ist *eine Art* von “Raubkatze”
- Vererbungshierarchien können in unterschiedlicher Weise ausgedrückt werden.





RainingData

Software Klassenvererbung

- Was ist, wenn wir schon eine Klasse haben, die auf diverse Nachrichten reagieren kann?
- Was ist, wenn Sie eine ähnliche Klasse erstellen möchten, die nur ein paar zusätzliche Funktionen beinhalten soll.
- Warum sollten wir die gesamte Klasse neu schreiben?
- Vererbung ist ein einfacher und eleganter Weg um Code wiederzuverwenden und die reale Welt sinnvoll darzustellen.
- Einige zusätzliche Methoden können definiert werden um die Möglichkeiten der Klasse zu erweitern.

RainingData

Was ist vererbt?

- Sichtbare Eigenschaften
- Bestimmte Variablen
- Externes Interface
  - Öffentliche Methoden

RainingData

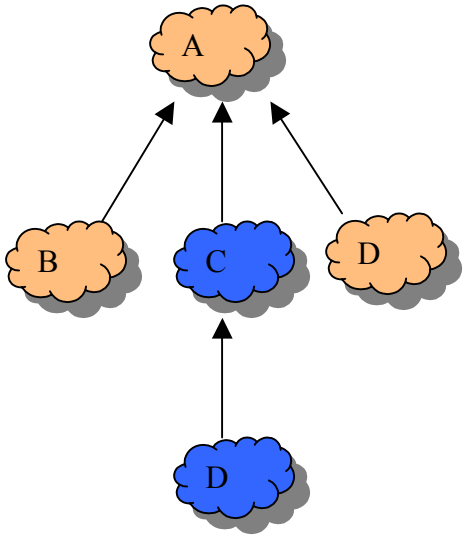
Überschreiben von Eigenschaften und Methoden

- Das Erstellen einer Subklasse ist Spezialisierung.
- Das Vereinen bestimmter Elemente von vererbten Klassen in eine gemeinsame Basis (oder Parent) ist Verallgemeinerung.
- Im OO Sprachgebrauch ist “Überschreiben” (“Overriding”) der Ausdruck für das Redefinieren einer Methode in einer abgeleiteten Klasse um spezialisiertes Verhalten zu ermöglichen.
- Die Eigenschaft oder Methode in einer Subklasse ersetzt diejenige aus der Superklasse.

RainingData

Einfache Vererbung

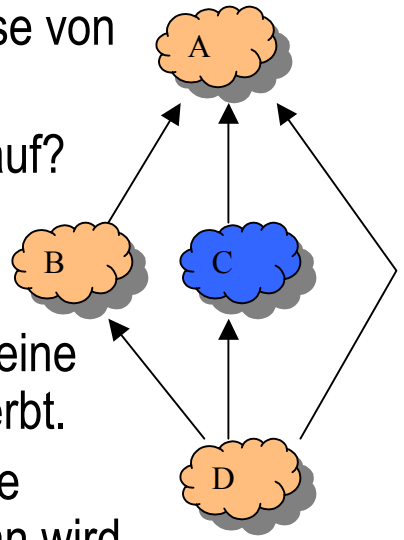
- In einer einfachen Vererbung kann eine Kind-Subklasse nur von einer einzigen Eltern-Superklasse vererbt sein.
- Jede Superklasse kann mehrere Subklassen haben.
- Einfache und saubere Mechanismen.



RainingData

Mehrfache Vererbung

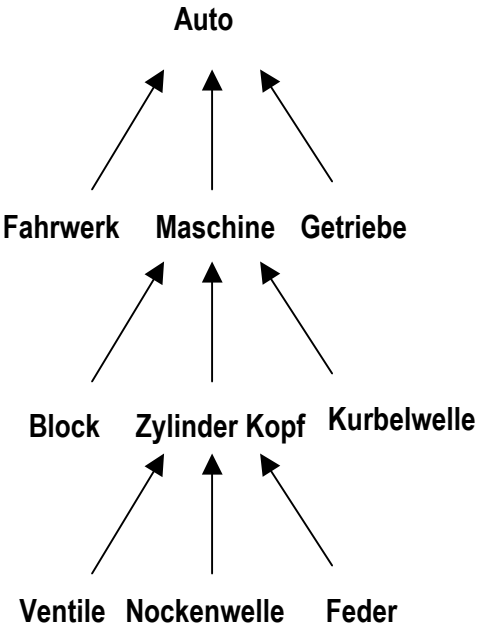
- Mehrfache Vererbung tritt dann auf, wenn eine Subklasse von mehreren Superklassen erbt.
- Welche Schwierigkeiten treten bei Mehrfachvererbung auf?
- A. hat Wiedersprüche durch mehrfaches Vererben von gleichen Eigenschaften und Methoden, z.B.
  - 2 Klassen B and C erben von Klasse A und es gibt eine andere Klasse D ,die von den Klassen A, B and C erbt.
  - Wenn jetzt Klasse A eine Methode fred() hat, welche sowohl von Klasse B und Klasse C geerbt wird, dann wird Klasse D sie ebenso von A,B und C erben.
  - Wenn fred() in D gerufen wird, welche fred Methode soll nun ausgeführt werden? Die von A, B oder C?



RainingData

Ansammlung und Container

- Eine alternative Hirarchie der Vererbung, wo Komponenten zusammengefasst werden.
- Nicht Vererbung (eine Maschine ist nicht ein spezielles Auto)
- Objekte in einer Ansammlung sind miteinander eng verbunden und arbeiten nur innerhalb der Ansammlung.
- Container sind wie Einkaufskörbe, die eine vielzahl unterschiedlicher Objekte beinhalten können.
- Objekte innerhalb eines Containers sind locker miteinander verbunden, jedes Objekt ist unabhängig der anderen benutzbar.



RainingData

Was ist eine Objektorientierte Sprache?

- 1987 schlug Peter Wenger eine Definition für Objektorientierte Sprachen vor.
- Damit eine Programmiersprache objektorientiert ist sollte sie folgende Eigenschaften haben:
  - Sie sollte auf Objekte basieren, damit der Programmierer auf einfache Weise gekapselte Programmobjekte erzeugen kann.
  - Sie sollte auf Klassen basieren, damit jedes Objekt zu einer Klasse gehört oder aus einer Klasse erzeugt werden kann.
  - Sie sollte Vererbung unterstützen, damit die Klassen in einer Superklassen-Subklassenhierarchie angeordnet werden können.

RainingData

Objektorientierung in Omnis Studio

- Jetzt werden wir uns die Möglichkeiten von Omnis Studio betrachten, das den Entwicklern erlaubt objektorientierte Applikationen zu erstellen.
  - Klassen
  - Vererbung
  - Instanziierung
  - Nachrichtenversand

RainingData

Omnis Studio Klassen

- GUI Klassen
  - Window*, Menu*, Toolbar*, Remote form*, Report*
- Datenklassen
  - Schema+, Query+, Table*, File, Search
- Nicht sichtbare, Verhaltensklassen
  - Object*, Task*, Remote task*, Code

* OO Klassen, die Instanziierung verlangen

\+ Schema und Query werden mit Tableklassen instanziiert.

RainingData

Was wird von einer Subklasse geerbt?

- Omnis Studio unterstützt nur die einfache Vererbung
- Öffentliche Methoden
- Klassenvariablen
- Instanzvariablen
- Eigenschaften (Properties)
- Untergeordnete Objekte
  - Felder auf Fensterklassen
  - Tools auf Toolbarklassen
  - Menüzeilen in Menüklassen

RainingData

Öffentliche und private Methoden

- Public Methoden haben immer ein vorangestelltes Dollarzeichen.
- Alle anderen Methoden sind privat.
- Rufen Sie eine private Methode indem Sie den Befehl: “Do method” verwenden. z.B:

```
Do method fred(p1,p2,p3)
```

  - Parameter werden innerhalb der runden Klammern übergeben
  - Das ist kein Nachrichtenversand!
- Verwenden Sie nicht den Befehl “Do method” um eine Öffentliche Methode zu rufen, selbst wenn sie in derselben Klasse ist.
- Öffentliche Methoden sollten über den Befehl “Do” gerufen werden.

RainingData

Instanziierung einer Klasse

- Verwenden des 4GL Befehls:

```
Open window instance myWind/Cen (p1,p2)
```

- Verwenden der Notation:

```
Do $windows.myWind.$open('*',kWindowCenter,p1,p2)
```

- Der Befehl referenziert zum Klassennamen.
- Der erste Parameter ist der Name der Instanz oder '*' um einen Instanznamen zu generieren.
- Die $construct Methode wird automatisch ausgeführt, wenn die Instanz erstellt wird.

RainingData

Andere Beispiele der Instanziierung

- Die Instanz des kaskadierenden Menüs wird zur gleichen Zeit wie die Instanz des zugehörigen Hauptmenüs erstellt.
- Menüinstanzen eines Fensters werden zur selben Zeit wie die Instanz des Fensters erstellt.
- Kontextmenüs werden instanziiert wenn der Anwender mit der rechten Maus auf ein Vaterobjekt (Fenster oder Feld) klickt.
- Bericht Instanz
  - Verwenden des 4GL Befehls

```
Set report name myRep
Prepare for print {* (#1,#2)}
```

  - Verwenden der Notation

```
Do $reports.myRep.$open('*',p1,p2)
```

  - Beachten Sie, dass der folgende Befehl auch einen Report instanziiert, sich aber unterschiedlich verhält:

```
Print report {* (p1,p2)}
```

RainingData

Instanzgruppen

- Es gibt eine Notationgruppe für instanziierte Klassen
- Für jede Instanz gibt es dort einen eigenen Eintrag
  - Geöffnete Fensterinstanzen - $iwindows
  - Task Instanzen - $itasks
    - Der Name ist identisch mit dem Librarynamen
  - Installierte Menüinstanzen - $imenus
    - Nur diejenigen Menüs, die in der Omnis Hauptmenübar installiert sind
  - Installierte Toolbarinstanzen - $itoolbars
    - Nur diejenigen Toolbars, die in der Omnis Haupttoolbar installiert sind
- Fensterinstanzen haben eigene Gruppen für Objekte, die darin instanziiert sind.
  - $iwindows.myWind.$menus.myMenu
  - $iwindows.myWind.$toolbars.myToolbar

RainingData

Schliessen der Instanz

- Benutzen eines 4GL Befehls

```
Close window instance myWind
```

- Benutzen der Notation:

```
Do $iwindows.myWind.$close()
```

- Der Befehl referenziert auf den Instanznamen
- Die Methode $destruct wird automatisch ausgeführt, wenn eine Instanz geschlossen wird, mit Ausnahme von Instanzen der Table- und Objektklassen.
  - Bei Table und Objektinstanzen können Sie die $destruct Methode manuell aus der $destruct Methode der übergeordneten Instanz ausführen (z.B. in einer Fensterinstanz aus der dortigen $destruct Methode)

```
Do myRow.$destruct()
```

RainingData

Versenden einer Nachricht

- Sie können eine Nachricht zu einem Objekt versenden mit Hilfe des “Do” Befehls.
- Eine Nachricht kann nur zu einer Instanz geschickt werden.
- Sie müssen eine Referenz zu dem Objekt haben, zu dem Sie die Nachricht schicken wollen (wie ein “Handler”).
  `Do $iwindows.myWind.$fred(p1,p2,p3)`
  - Das ist wie eine Adresse auf einem Brief
  - Parameter werden in Klammern übergeben
  - `$iwindows.myWind` ist eine Referenz
- Sie können keine Nachricht zu einer Kontextmenüinstanz senden, da es durch eine Notationsgruppe nicht erreichbar ist.

RainingData

Verteilen einer Nachricht

- Sie können die gleiche Nachricht zu einer Anzahl von Objekten innerhalb einer Gruppe mit Hilfe der $sendall Methode schicken.

```
Do $iwindows.$sendall($ref.$close())
```

  - Für jedes Mitglied der Gruppe wird die Nachricht wiederholt
  - Für jede Wiederholung referenziert $ref zu dem aktuellen Objekt
- Die Methode hat einen optionalen zweiten Parameter um selektieren zu können.

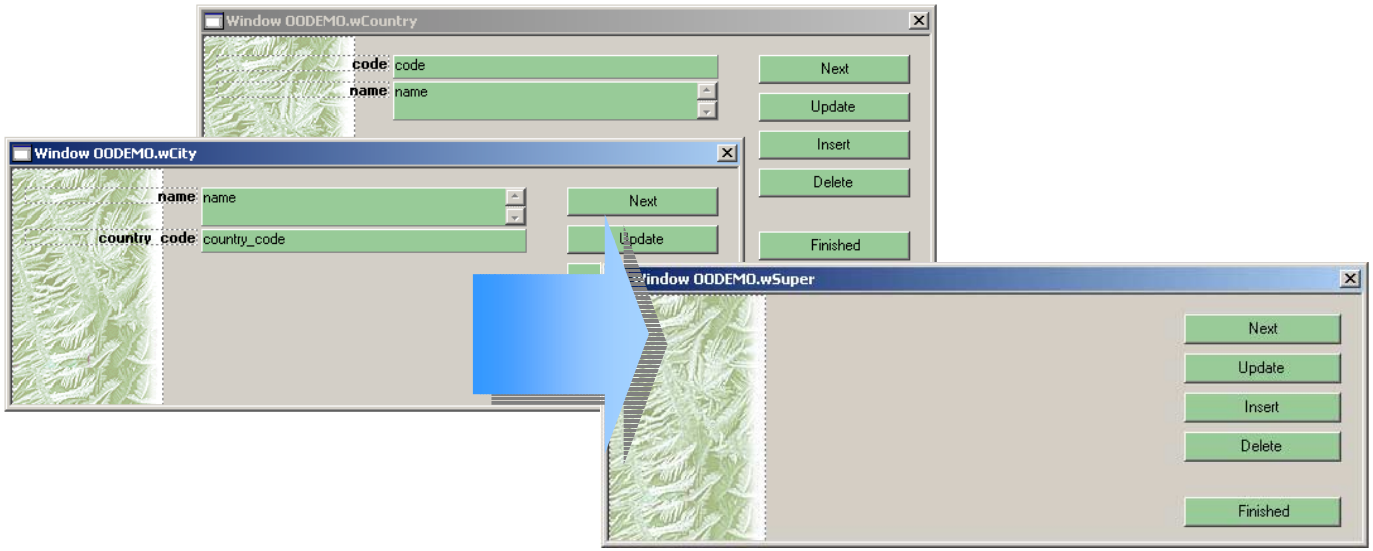
```
Do $iwindows.$sendall($ref.$close(),
  $ref.$class().$name='myClass')
```

- Die Nachricht wird nur dann zum Objekt gesendet, wenn der zweite Parameter wahr (kTrue) oder ungleich Null ist.

RainingData

Erstellen einer Superklasse

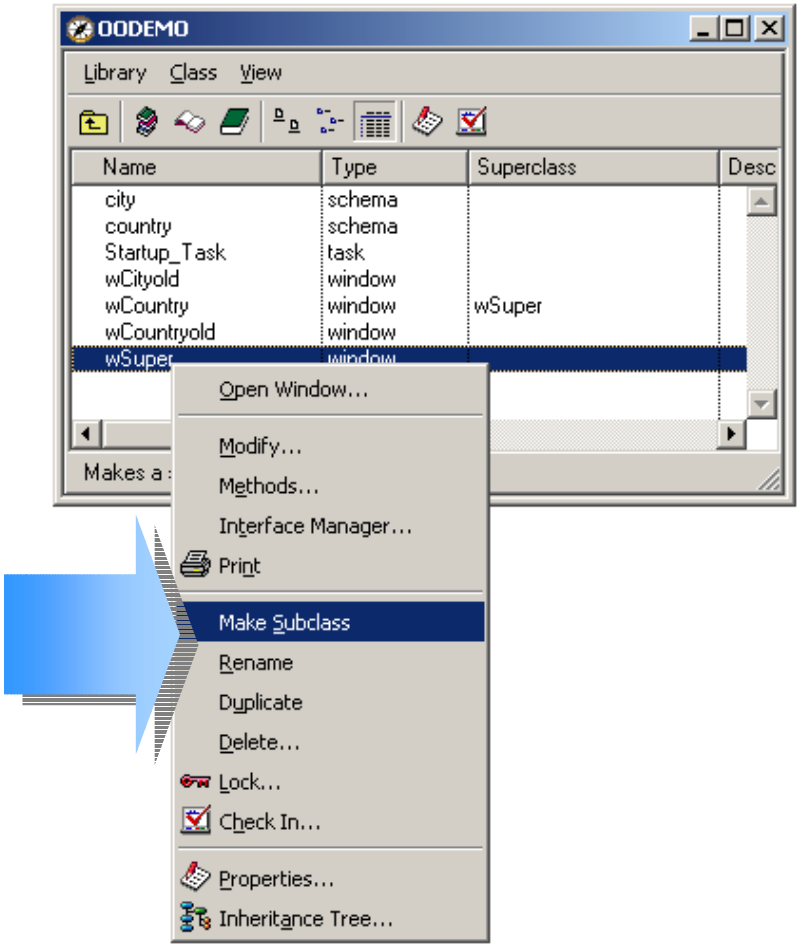
- Identifizieren gemeinsamer Merkmale und Verhaltensweisen.
- Erstellen Sie eine Abstraktion mit gemeinsamen Merkmalen.
- Das wird Ihre Basis für die Superklasse.



RainingData

Erstellen einer Subklasse

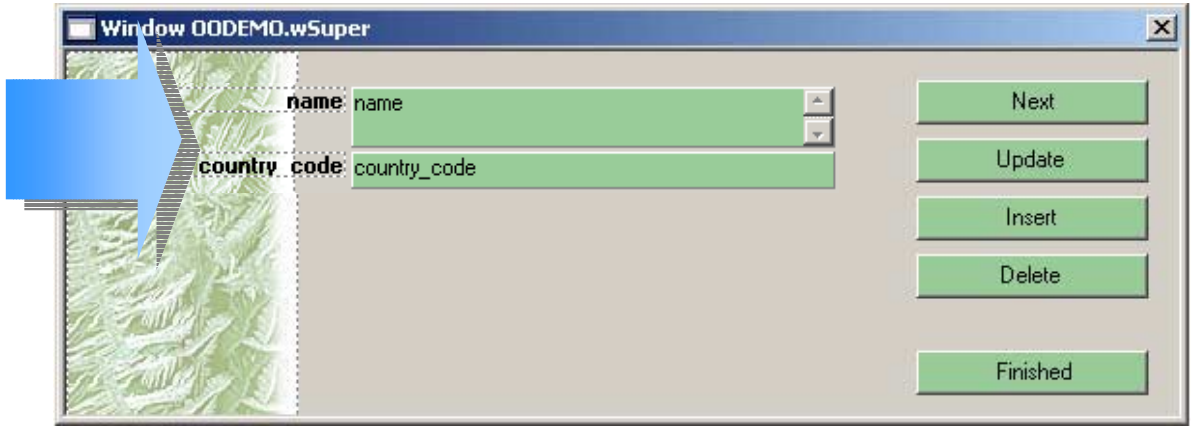
- Klicken Sie im Browserfenster mit der rechten Maus auf Ihre Superklasse.
- Wählen Sie “Make Subclass”.
- Geben Sie der Subklasse einen geeigneten Namen.
- Beachten Sie, dass die zugeordnete Superklasse im Browserfenster angezeigt wird.



RainingData

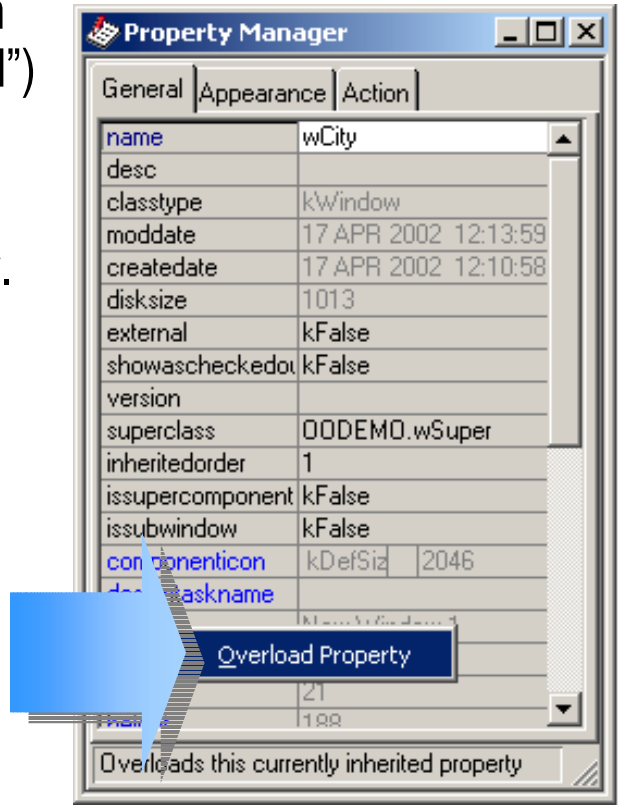
Machen Sie aus der Subklasse eine Spezialisierung

- Fügen Sie die Merkmale hinzu, die die Subklasse zu einer Spezialisierung der Superklasse machen.
- Fügen Sie Felder hinzu.
- Überschreiben Sie Eigenschaften und Methoden.



Überschreiben von Eigenschaften

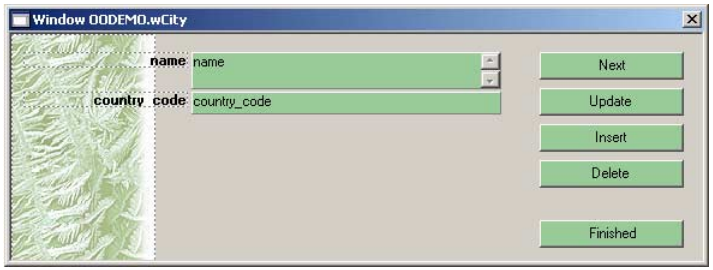
- Viele vererbte Eigenschaften können Sie im Property Manager überschreiben (“overload”)
- Vererbte Eigenschaften werden in Blau angezeigt.
- Selektieren Sie das Objekt im Fenstereditor.
- Klicken Sie mit der rechten Maus auf die benötigte Eigenschaft im Property Manager und selektieren Sie “Overload Property”
- Geben Sie wie gewohnt den neuen Wert rechts neben dem Eigenschaftsnamen ein.



Raining Data

Was Sie nicht überschreiben können

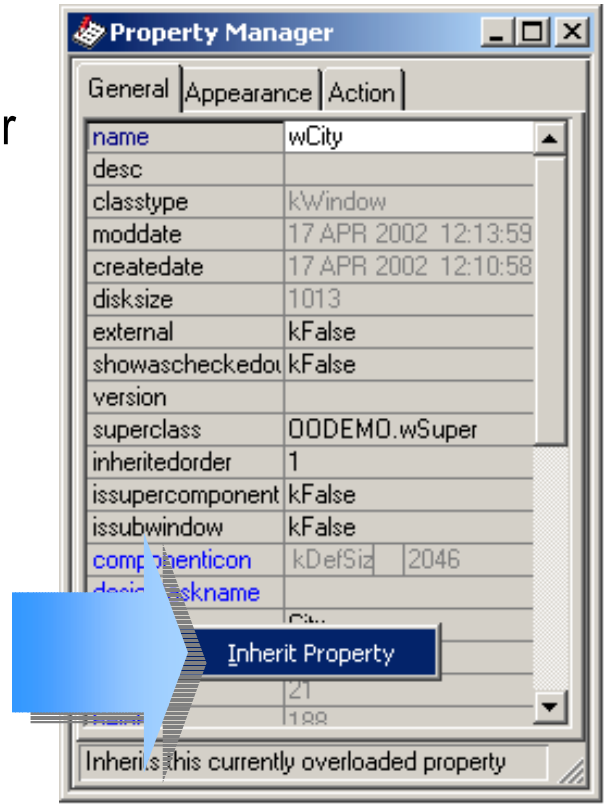
- Sie können keine Eigenschaften von Objekten überschreiben, die aus der vererbten Klasse stammen.
- z.B. die Eigenschaften $top, $left, $width und $height eines Knopfes aus der vererbten Klasse können nicht überschrieben werden.
- Hinweis: Benutzen Sie “floating field” Eigenschaften um automatisch die Felder neu anzuordnen, wenn ein Subclassfenster in der Größe verändert wird. Sie können auch die Felder per Notation in der $construct Methode beeinflussen.



RainingData

Vererben von Eigenschaften

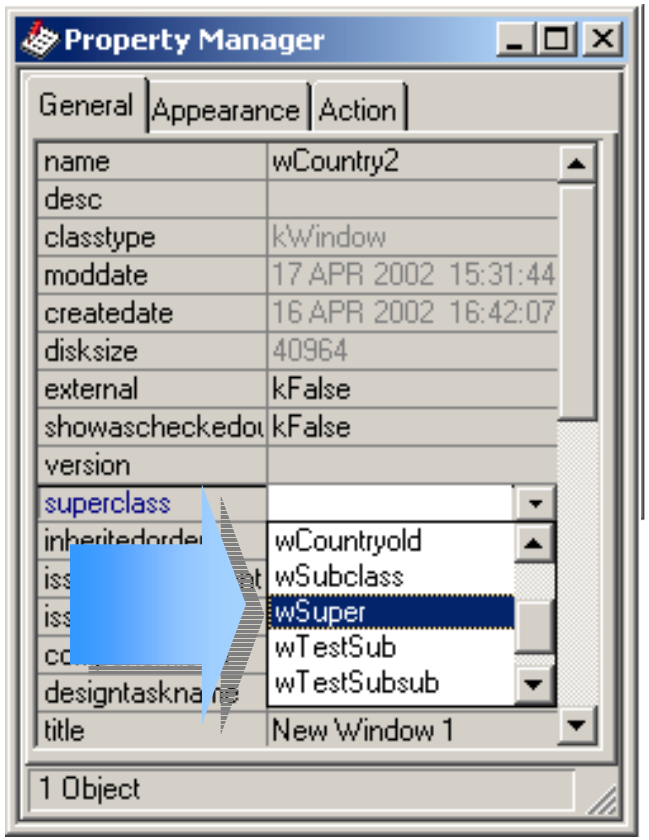
- Wenn eine Eigenschaft bereits überschrieben wurde, dann können Sie nachträglich die Eigenschaft wieder von Ihrer Superklasse erben mit Hilfe des Property Managers.
- Nicht vererbte Eigenschaften werden in Schwarz angezeigt.
- Selektieren Sie das Objekt im Fenstereditor.
- Klicken Sie mit der rechten Maus auf die benötigte Eigenschaft im Property Manager und selektieren Sie “Inherit Property”
- Der vererbte Wert wird auf der rechten Seite blau angezeigt.



RainingData

Setzen der Eigenschaft $superclass

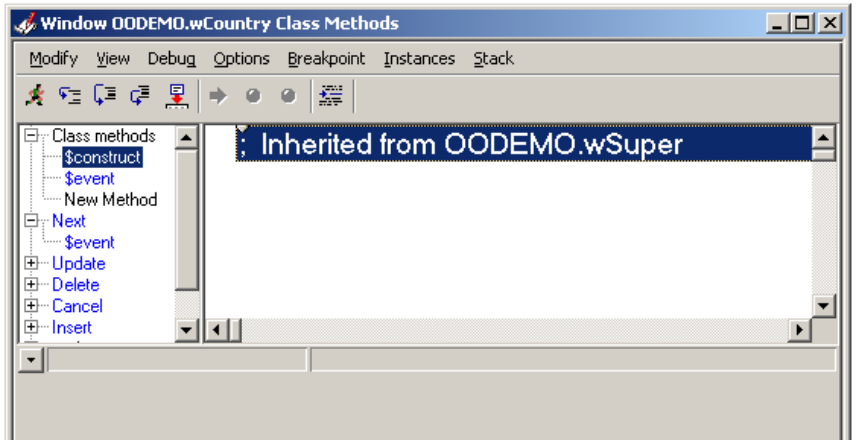
- Eine alternative Möglichkeit eine Subklasse zu erstellen ist dessen Superclass Eigenschaft zu setzen.
- Klicken Sie mit der rechten Maus im Browserfenster auf die Klasse und selektieren Sie “Properties”
- Selektieren Sie die Eigenschaft Superclass im Property Manager
- Nicht unbedingt empfehlenswert weil Standardeigenschaften nicht automatisch veerbt werden. (Sie müssen manuell vererbt werden)



RainingData

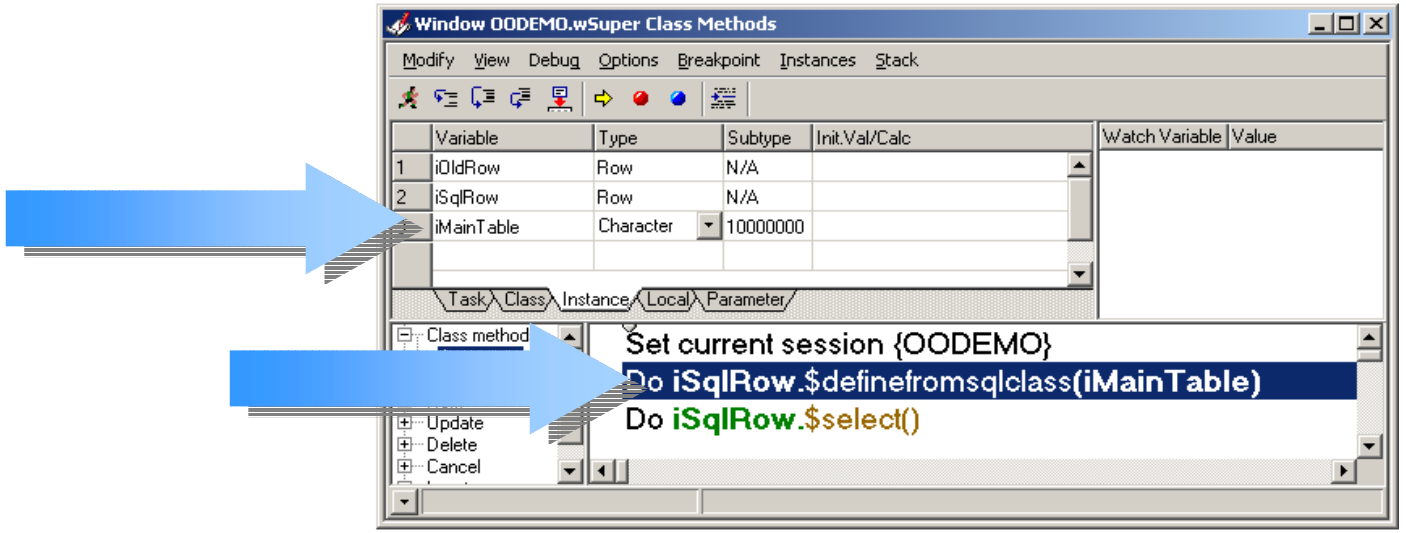
Vererbte Methoden

- Nur öffentliche Methoden mit einem voranstehenden $-Zeichen werden vererbt.
- Vererbte Methoden werden blau angezeigt.
- Wenn geerbte Methode durch eine Nachricht angestossen wurde, wird die Methode der Superklasse ausgeführt.



Stellen Sie sicher, dass Methoden abstrahiert sind.

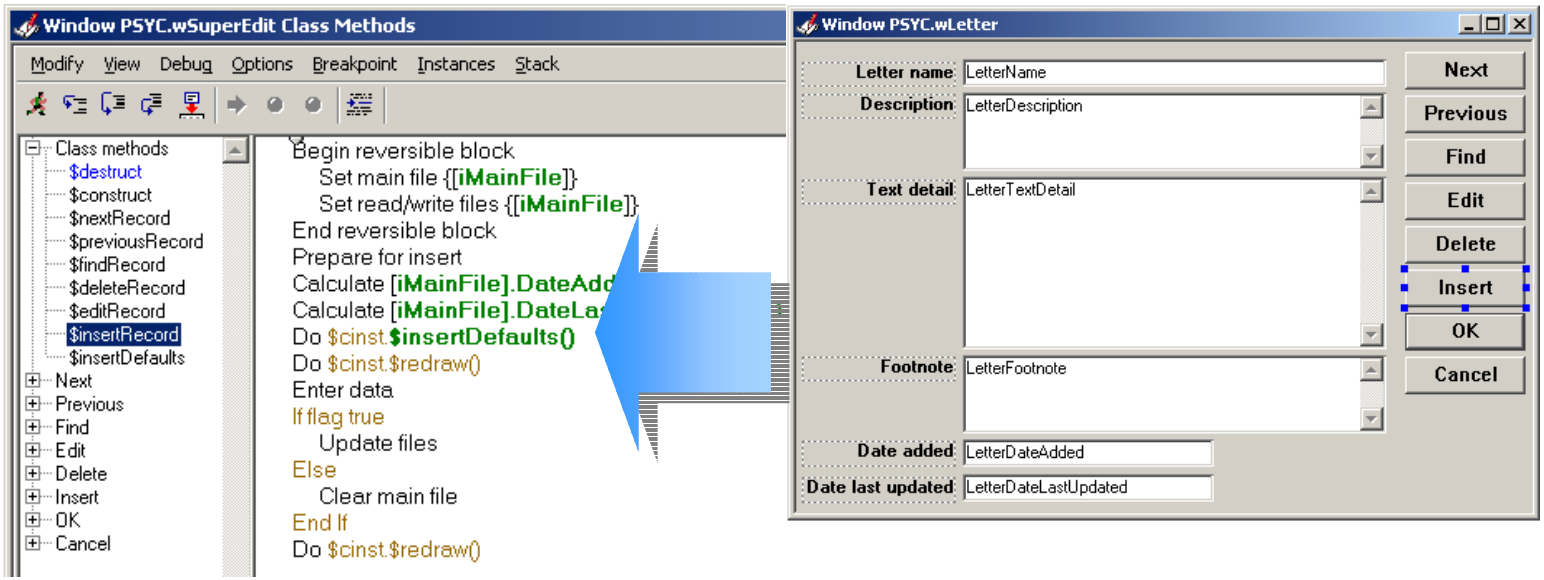
- Methoden in Superklassen müssen allgemeingültig vereinfacht werden.
- Code, der bestimmte Objekte anspricht, sollte in die Subklasse gelegt und die Methode in der Subklasse überschrieben werden.
- Alternativ können Sie eine Instanzvariable in der Superklasse anlegen, damit bestimmte Werte aus der Subklasse in der Methode der Superklasse verwendet werden können.



Raining Data

Aufrufpunkte für die Subklasse

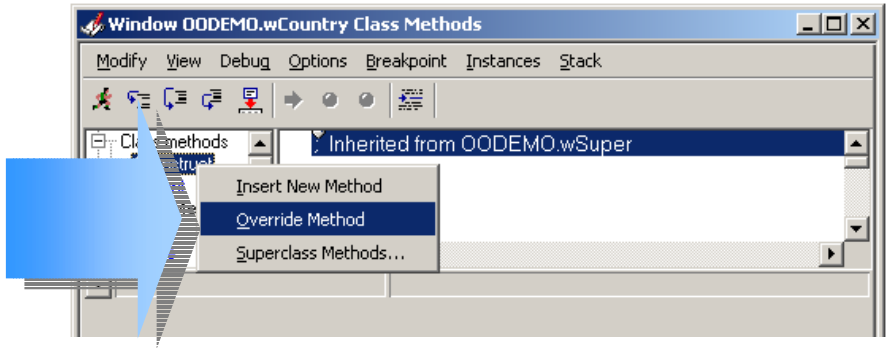
- Manchmal benötigt man den Aufruf einer Methode der Subklasse aus einer Methode der Superklasse.
- Erstellen Sie dazu eine öffentliche “Dummy” Methode in der Superklasse, aus anderen Superklassenmethoden aufgerufen werden kann.
- Überschreiben Sie diese Methode in der Subklasse bei Bedarf.



RainingData

Überschreiben von Methoden

- Sie können die vererbte Methode in der Subklasse überschreiben.
- Klicken Sie mit der rechten Maus auf die Methode im Methodeditor und selektieren Sie “Override Method”.
- Fügen Sie wie gewohnt den neuen Code auf der rechten Seite ein.
- Wenn die Methode durch eine Nachricht angestossen wird, wird die Methode aus der Subklasse ausgeführt.



Anstossen der Superklassenmethode

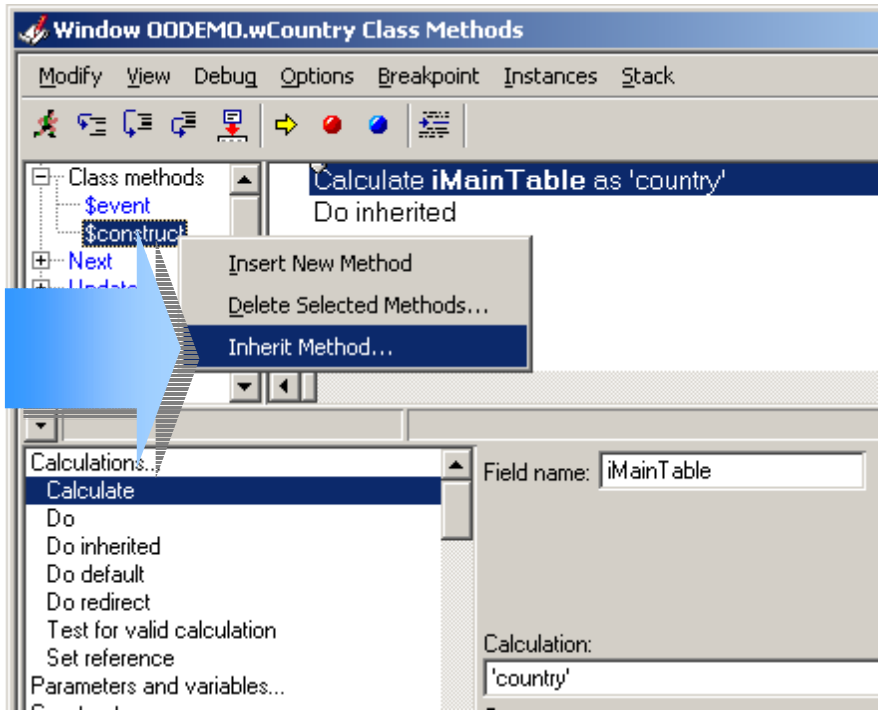
- Um den Code der überschriebenen Methode der Superklasse auszuführen können Sie den Befehl

  `Do inherited` verwenden.
  - Übergebene Parameter werden an beide Methoden weitergereicht.
  - Unterschiedliche Parameter können Sie mit dieser Technik nicht weitergeben.
  - Beide Methoden können Parameter vom Typ Field Reference benutzen, um auf die gleichen Daten zuzugreifen.
- Wenn Sie die Parameter überschreiben wollen, die normalerweise zur Methode der Superklasse gesendet würde, dann benutzen Sie:

  ```
  Do $inherited.$myMethod(p7,p8,p9)
  ```

Vererben von Methoden

- Eine Methode, die bereits in der Subklasse überschrieben wurde, kann wieder vererbt werden.
- Klicken Sie mit der rechten Maus im Methodeneditor auf die Methode und selektieren Sie “Inherit Method”
- Beachten Sie, dass dabei alle bereits eingegebenen Methodenzeilen der Subclassmethode gelöscht werden.



Statische versus Dynamische Variablenreferenzierung

- Statische Referenz

```
Calculate iCust as 'Fred'
Calculate myVar as iCust
```

  - Die Referenz zu iCust ist tokenisiert.
- Dynamische Referenz

```
Calculate $cinst.iCust as 'Jill'
Calculate myVar as $cinst.iCust
```

  - Achtung: iCust ist einfacher Text und wird erst zur Laufzeit als Variable erkannt.
- Achten Sie auf die richtige Schreibweise bei dynamischer Referenz. (Anders als statische Referenzen werden sie nicht von der IDE geprüft.)

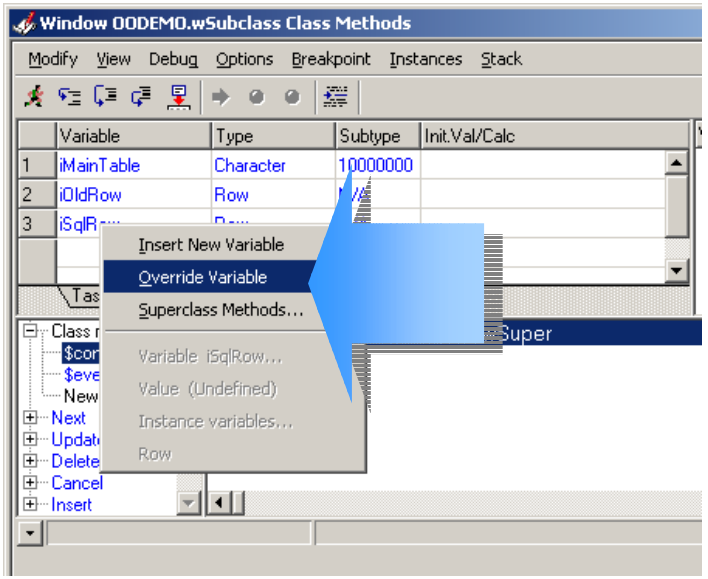
```
Calculate $cinst.iCus as 'Sam'
Calculate myVar as $cinst.iCus
```

  - In diesem Fall wird myVar einen “NULL” Wert beinhalten.

RainingData

Überschreiben von Variablen

- Vererbte Variablen werden in Blau angezeigt.
- Klicken Sie mit der rechten Maus im Methodeditor auf die Variable und selektieren Sie “Override Variable”.
- Die Variable in der Subklasse ist nun eine völlig andere Variable als die in der Superklasse.
- Das kann mit Klassen- und Instanzvariablen geschehen.
- Beachten Sie, dass Copy und Paste **automatisch** Variablen überschreiben kann. Um dies zu vermeiden können Sie Code zunächst kommentieren und erst dann mit Copy und Paste einfügen und wieder unkommentieren. (Vergessen Sie nicht die Kommentierung Ihres ursprünglichen Codes wieder rückgängig zu machen.)



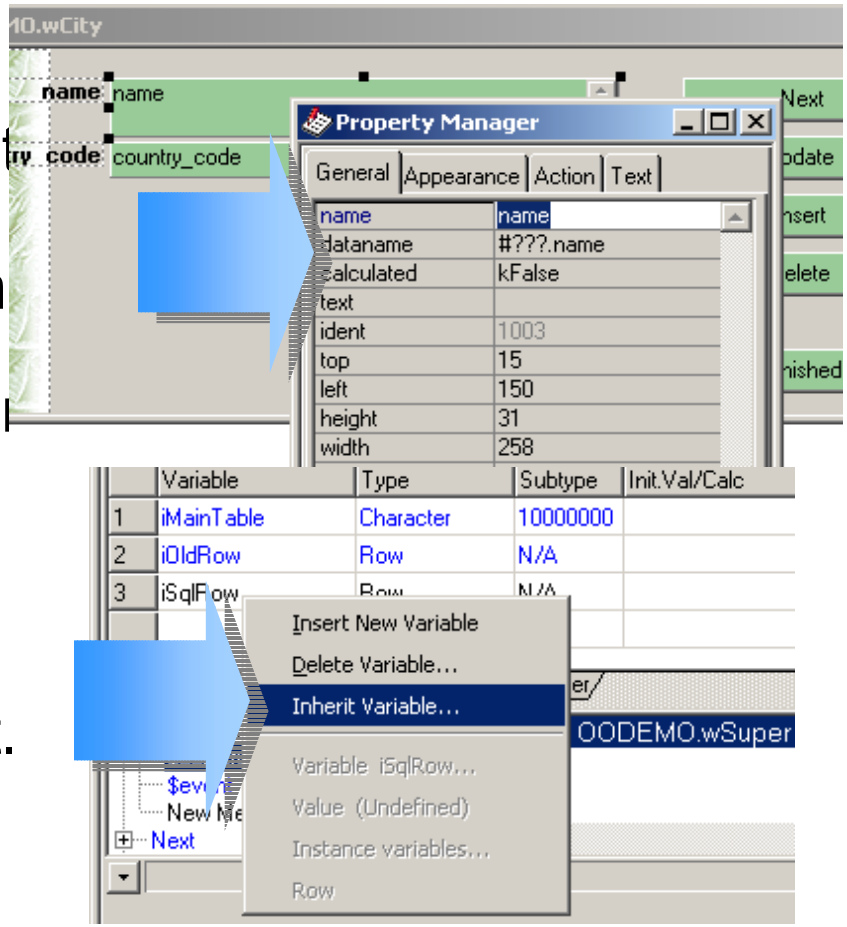
RainingData

Auf überschriebene Variablen zugreifen.

- Überschriebene Variablen haben immer noch Gültigkeit.
- Wenn Sie auf eine Superklassenvariable von einer Subklassenmethode zugreifen wollen, verwenden Sie:
  `$inherited.myVar`
  - Beachten Sie, dass sie damit auf die vererbte Variable der jeweils nächsten Ebene des Vererbungsbaumes zugreifen.
- Um auf eine Variable der Subklasse aus einer Superklassenmethode zuzugreifen, benutzen Sie:
  `$cinst.myVar`
  - Beachten Sie, dass Sie damit auf die Variable der jeweils untersten Ebene des Vererbungsbaumes zugreifen.

Vererben von Variablen

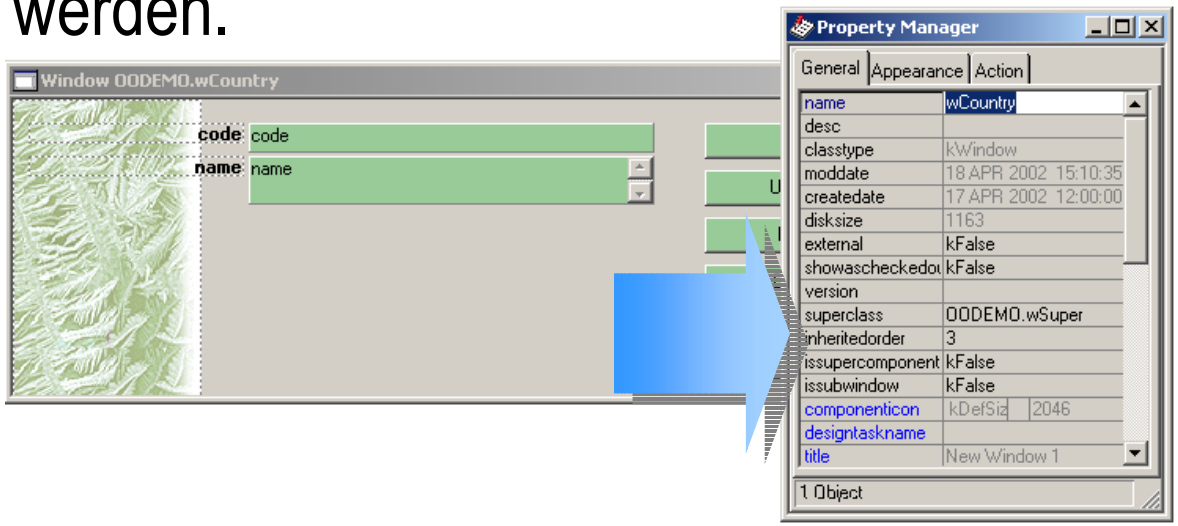
- Eine Variable, die in einer Subklasse überschrieben wurde, kann wieder vererbt werden.
- Klicken Sie mit der rechten Maus auf die Variable im Methodeneditor und wählen sie “Inherit Variable”
- Die Subklassenvariable wird dann entfernt und Referenzen zu dieser Variablen werden gelöscht. (Sie wechseln zu #???)



RainingData

Vererbung und Taborder

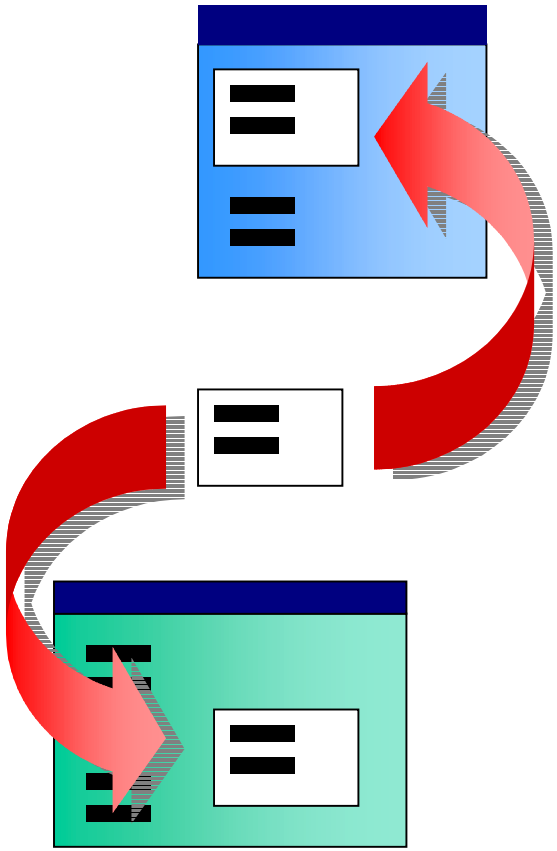
- Vererbte Felder erscheinen zuerst in der Taborder
- Die Taborder des ersten vererbten Feldes kann über die Eigenschaft $inheritedorder gesetzt werden.
- Das sollte für jede Ebene im Vererbungsbaum gemacht werden.



RainingData

Subwindows

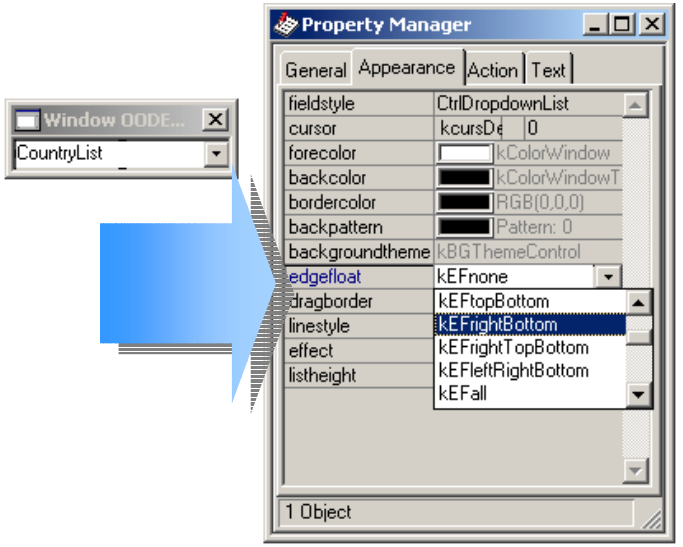
- Ein Subwindow ist eine Ansammlung
- Es ist eine Fensterklasse und kann normale Felder und Hintergrundobjekte beinhalten.
- Es kann viele Felder oder auch nur ein oder zwei beinhalten.
- Ein einzelnes Subwindow kann beliebig oft in anderen Fensterklassen wiederverwendet werden.
- Es ist eine Alternative zur Vererbung um Programmteile wiederzuverwenden.
- Die Omnis Studio “Version” eines ActiveX Controls



RainingData

Erstellen eines Subwindows

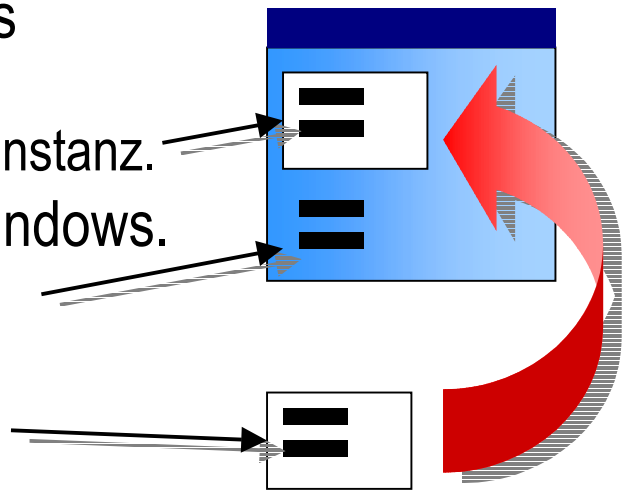
- Erstellen Sie ein normales Fenster.
- Vergrößern Sie das Feld so, dass es das Fenster ausfüllt.
- Setzen Sie die Eigenschaft $issubwindow auf kTrue
- Hinweis: Für Subwindows, die Metafelder implementiert haben, benutzen Sie Floating Eigenschaften auf den Objekten um sicherzustellen, dass sich die Objekte automatisch ausrichten.



RainingData

Subwindow Schnittstelle

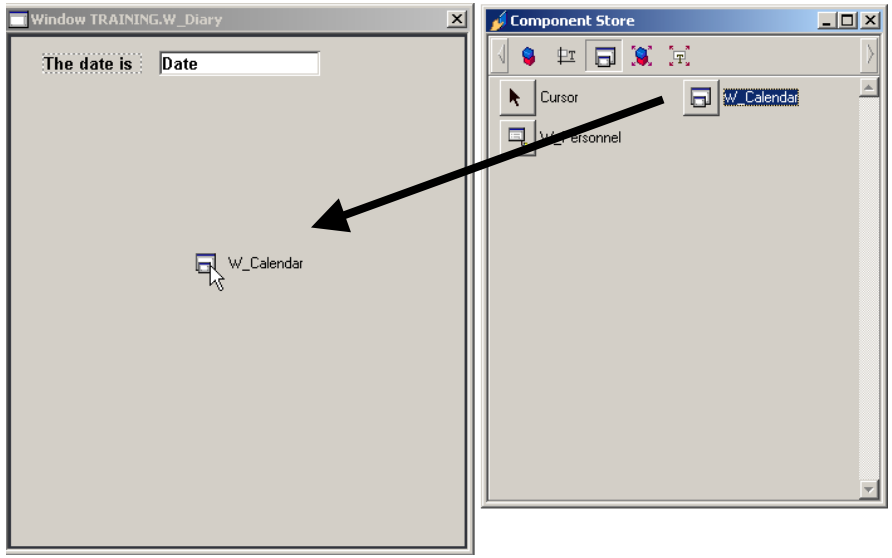
- Ein Subwindow muss mit seinem Parentwindow und umgekehrt kommunizieren können.
- Wir brauchen eine Schnittstelle mit öffentlichen Methoden.
- Innerhalb des Subwindowfeldes des Parentwindows.
  - $cfield referenziert zur Subwindowinstanz.
- Innerhalb einer Methode des Subwindows.
  - $cwind referenziert zur Instanz des Parentwindow.
  - $cinst referenziert zur Instanz des Subwindows.



RainingData

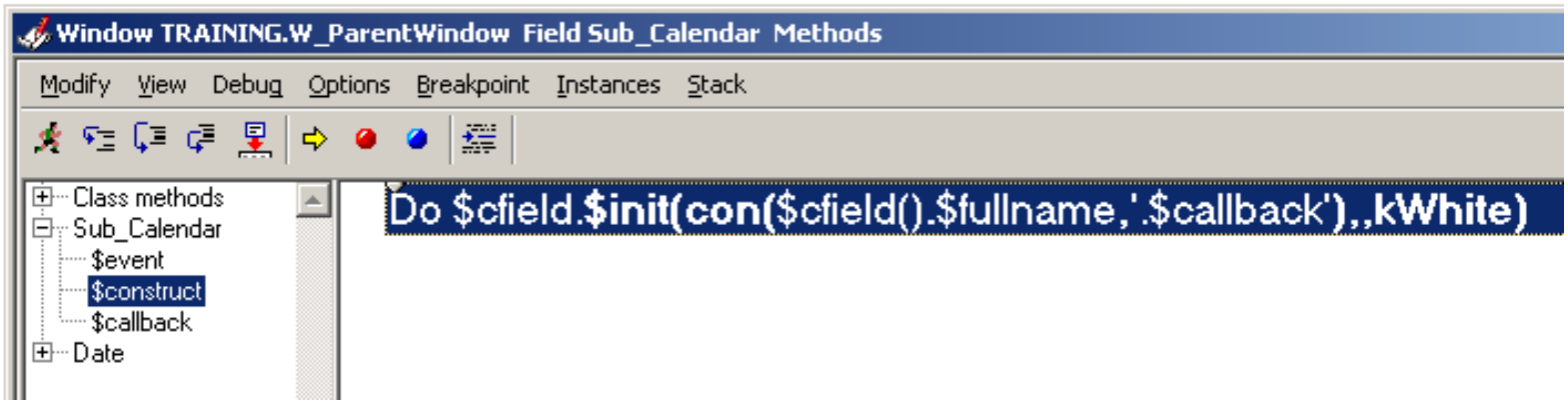
Benutzen eines Subwindows

- Selektieren Sie das Subwindow Icon in der Toolbar des Component Stores.
- Ziehen Sie das benötigte Subwindow auf Ihr Fenster.
- Richten Sie die benötigten Schnittstellenaufrufe zwischen dem Hauptfenster und dem Subwindow ein.



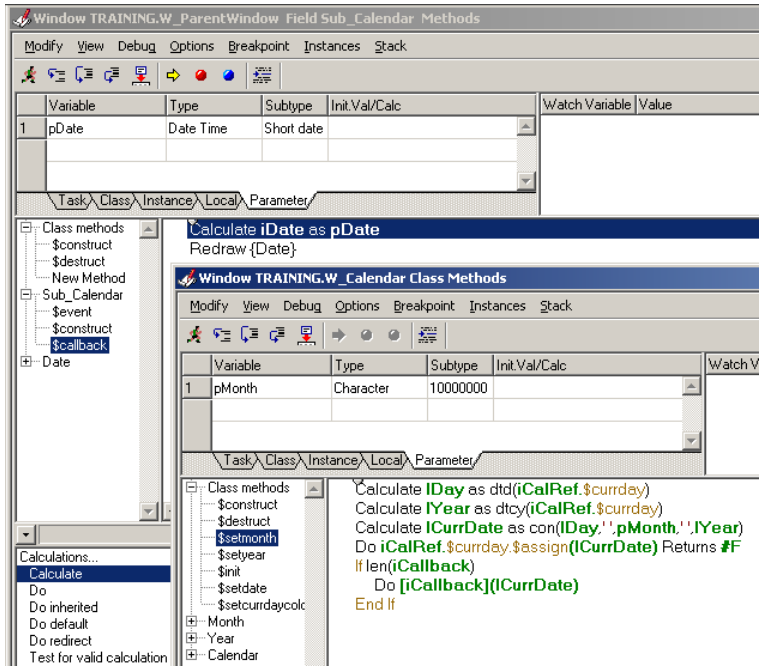
Initialisierung eines Subwindows

- Initialisieren Sie das Subwindow, indem Sie aus der $construct Methode des Container Feldes eine öffentliche Methode des Subwindows rufen.
- Übergeben Sie die benötigten Parameter.
- Diese Parameter können dazu benutzt werden um Daten innerhalb des Subwindows und andere visuelle Aspekte des Subwindows zu initialisieren. (z.B. die Farben der Felder, aktivieren/deaktivieren etc.)



Subwindow Callback Methode

- Wenn ein Event innerhalb des Subwindows auftritt, kann er nicht im Hauptfenster abgefangen werden.
- Wir brauchen einen Callback zu unserem Hauptfenster.
- Wenn die Initialisierung des Subwindows gerufen wird, geben Sie dort den kompletten Pfad für die Callback Routine an, die aufgerufen werden sollte, wenn ein Event im Subwindow auftritt.
- Speichern Sie den Pfad der Callback Methode in einer Instanzvariablen des Subwindows.
- Die Callback Methode kann Daten als Parameter vom Subwindow erhalten.



RainingData

Kombinieren von Subwindows

- Subwindows können mit anderen Feldern auf dem Fenster kombiniert werden.
- Andere Objekte kommunizieren mit dem Subwindow über das Public Interface (öffentliche Schnittstelle)
- Das Subwindow kann mit anderen Objekten über die Callback Routine kommunizieren.

RainingData

Tableklassen

- werden instanziiert als Row- oder Listvariablen
  - Table Instanzen und Row- oder Listvariablen sind ein und dasselbe.
  - Die Instanz wird erstellt, wenn die Row- oder Listvariable definiert wird.

```
Do myRow.$definefromsqlclass('myTab',,p1,p2)
```

  - Definieren Sie von einer Tableklasse, nicht Schema- oder Queryklasse.
  - Parameter können zur $construct Methode weitergereicht werden, Beachten Sie: ',,'
- vereinen ein Dateninterface via einer Row- oder Listvariablen.
- Methoden sind eine Abstraktion von SQL.
- ermöglichen Ihnen die Datenlogik von Ihrem GUI zu trennen.
- Public Methoden sind zugänglich via Row- oder Listvariable:

```
Do myRow.$select(con("where myCol = '",myRow.myCol,"'"))
```

- In Tableklassenmethoden können Sie via $cinst auf die Instanz, also auf die Row- oder Listvariable referenzieren.
- Die Instanz wird vernichtet, wenn die Variable geleert wird.
  - Die $destruct Methode wird nicht automatisch angestossen.

RainingData

Objectklassen

- Ähnliche Fähigkeiten wie Instanzen von Tableklassen aber kein Dateninterface (Row- oder Listvariable)
- Wird instanziiert in einer Variablen vom Typ Objekt.
- Kann statisch oder dynamisch definiert werden.
- Die Instanz wird zerstört, wenn die Variable geleert wird.
  - Die $destruct Methode wird nicht automatisch angestossen.

RainingData

Object Instanzen

- Statische Definition
  - Definieren Sie eine Variable mit dem geeigneten Gültigkeitsbereich.
  - Wählen Sie den Datentyp Objekt und setzen Sie den Subtype auf Ihre Objektklasse.
  - Die $construct Methode wird ausgeführt, wenn Sie das erste Mal eine Methode aufrufen.
- Dynamische Definition
  - Definieren Sie eine Variable mit dem geeigneten Gültigkeitsbereich.
  - Wählen Sie den Datentyp Objekt aber lassen Sie den Subtype leer.

    ```
    Do $clib.$objects.myObj.$new() Returns myVar
    ```
  - Die $new Methode hat die gleiche Aufgabe wie $open
  - Die $construct Methode wird ausgeführt, wenn die Instanz mit $new erstellt wird.

RainingData

Speichern von Objekten in der Datenbank

- Da Objekt ein Standartdatentyp ist, können Objekte in einer Spalte der Datenbanktabelle abgespeichert werden.
- Es können nicht alle Objektinstanzen, die bestimmte Werte enthalten identifiziert werden, ohne das alle Instanzen gelesen werden.
- Standardmäßig
  - Die Daten des Objektes (Instanzvariablen) werden in der Datenbank gespeichert.
  - Die Methoden werden aus der Objektklasse verwenden und können sich im Laufe der Zeit ändern.
- Um Methoden in der Datenbank zu speichern.
  - Setzen Sie die Eigenschaft der Objektklasse $selfcontained auf kTrue
  - Das macht einen “Schnappschuss” der Methoden.
  - Gespeicherte Objekte brauchen mehr Platz.

RainingData

Zusammenfassung

- Die Applikation basiert auf Softwareobjekte.
- Es ist ein anderer Weg um Systeme zu bauen.
- Es ist ein Paradigmenwechsel.
- Es ist eine Technologie.
- Es ist vielmehr eine Methodik.
- Vorteile:
  - Wiederverwendbarkeit
  - Konstistenz
  - Wartbarkeit
  - Sehr kurze Entwicklungszeiten (nachdem die Superklassen erstellt wurden)