# Using OODialog without the Resource Workshop

# Introduction

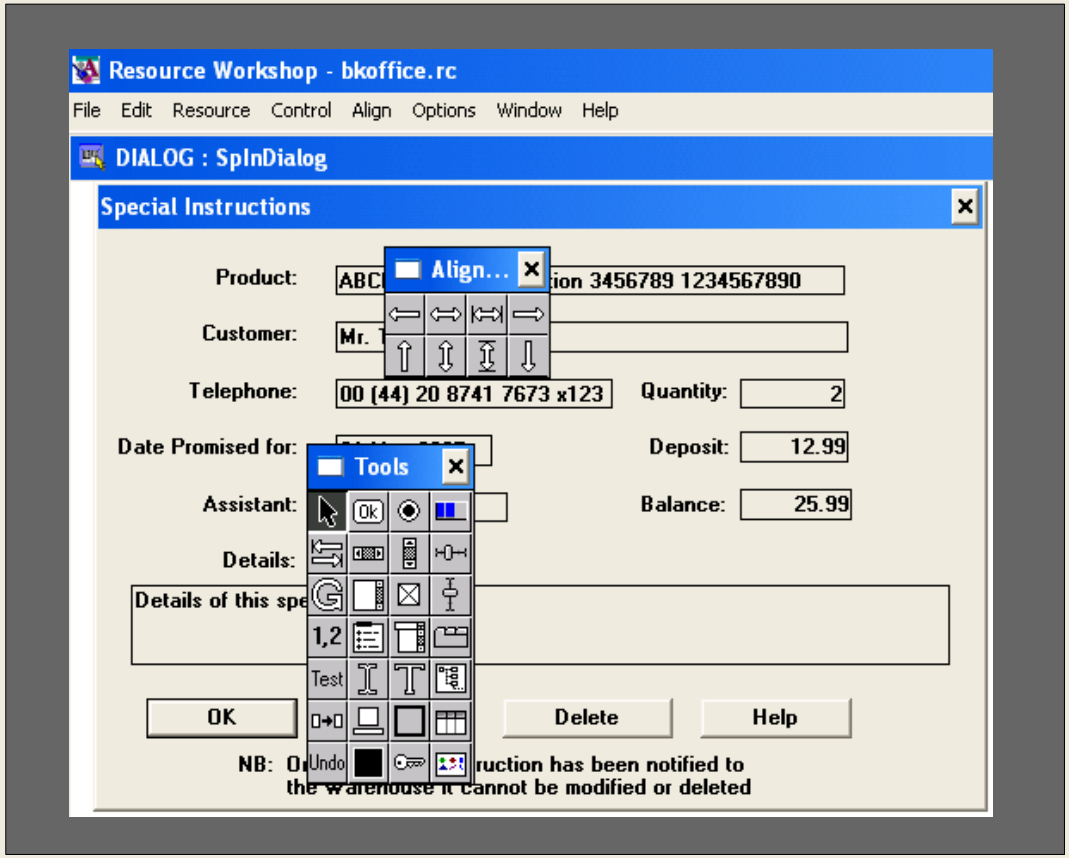
**OODialog**

- Object Rexx Windows Gui Manager

# Introduction

**OODialog**

- Object Rexx Windows Gui Manager
- Came with an IDE Application



# Introduction

**OODialog**

- Object Rexx Windows Gui Manager
- Came with an IDE Application
- IDE did not come across to ooRexx

# Introduction

**OODialog**

- Object Rexx Windows Gui Manager
- Came with an IDE Application
- IDE did not come across to ooRexx

**Me**

- Object Rexx developer since 2001

# Introduction

**OODialog**

- Object Rexx Windows Gui Manager
- Came with an IDE Application
- IDE did not come across to ooRexx

**Me**

- Object Rexx developer since 2001
- Designed 80+ dialogs using ooDialog

# Introduction

**OODialog**

- Object Rexx Windows Gui Manager
- Came with an IDE Application
- IDE did not come across to ooRexx

**Me**

- Object Rexx developer since 2001
- Designed 80+ dialogs using ooDialog
- No expert on ooDialog code or history

# Introduction

**OODialog**

- Object Rexx Windows Gui Manager
- Came with an IDE Application
- IDE did not come across to ooRexx

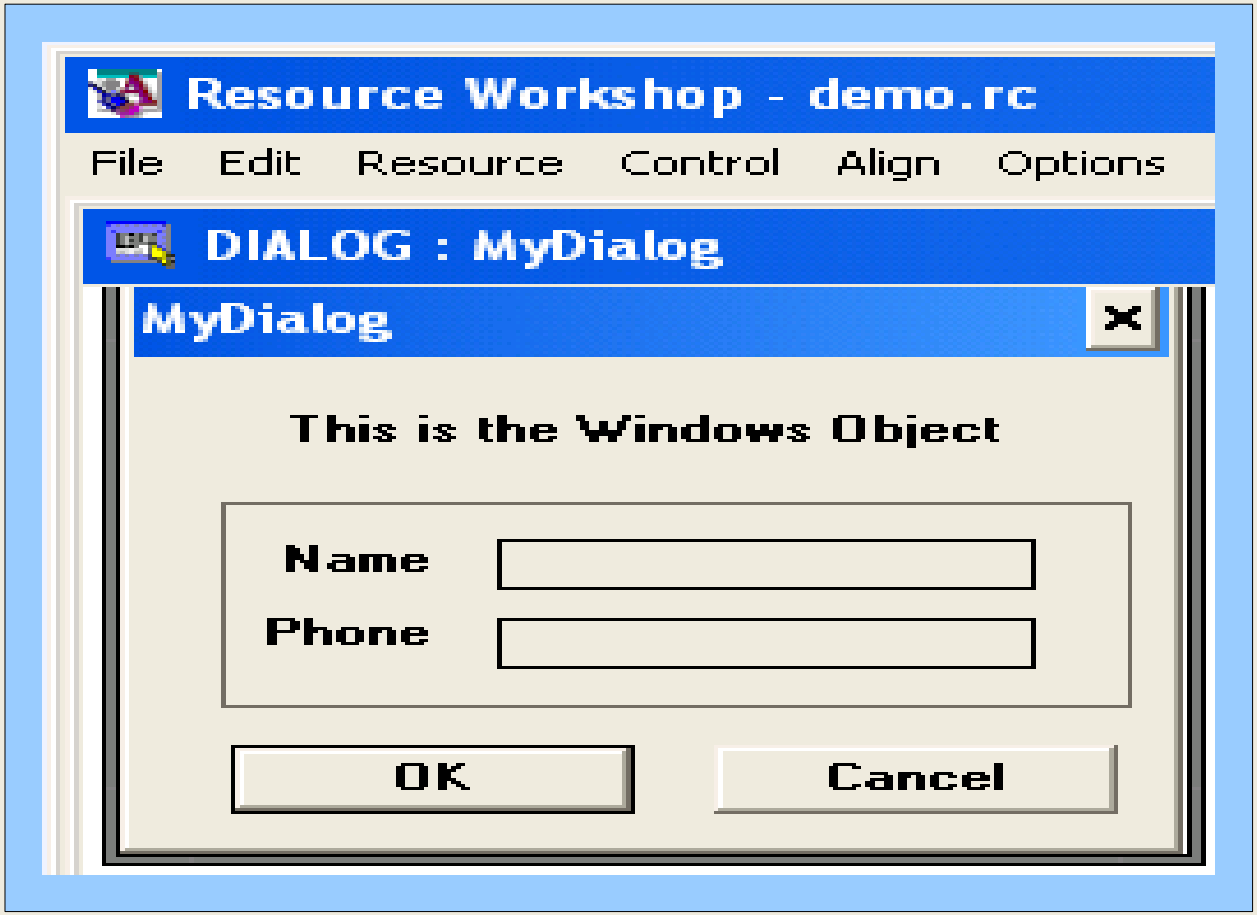
**Me**

- Object Rexx developer since 2001
- Designed 80+ dialogs using ooDialog
- No expert on ooDialog code or history
- Stopped using Resource Workshop

# The development cycle with The Resource Workshop

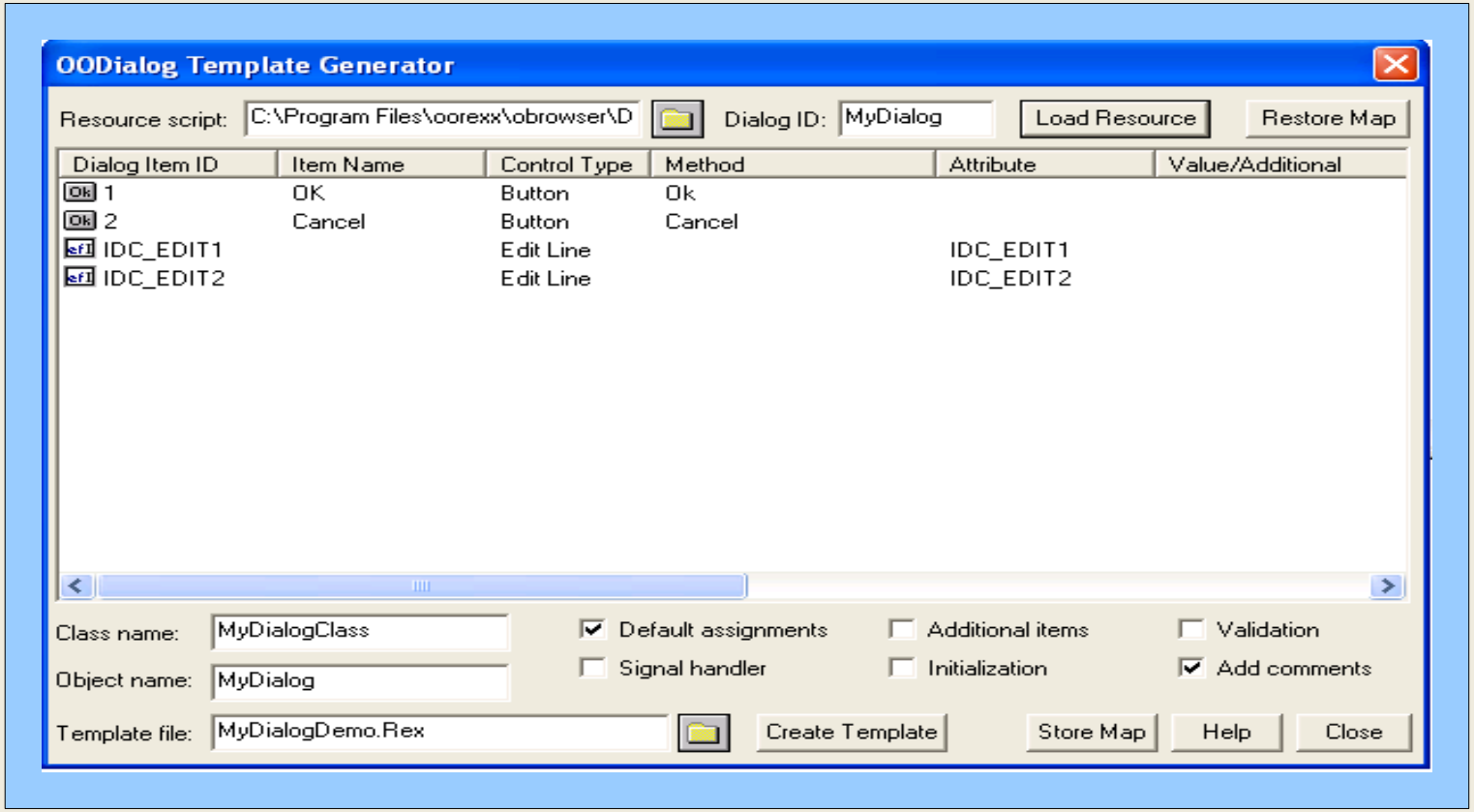
# Resource Workshop development cycle

- Design Windows GUI Object

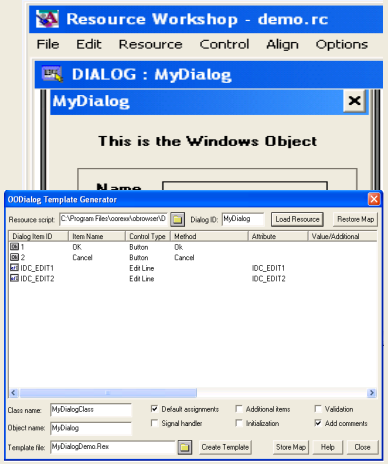


# Resource Workshop development cycle


- Design Windows GUI Object
- Run OODialog Template Generator



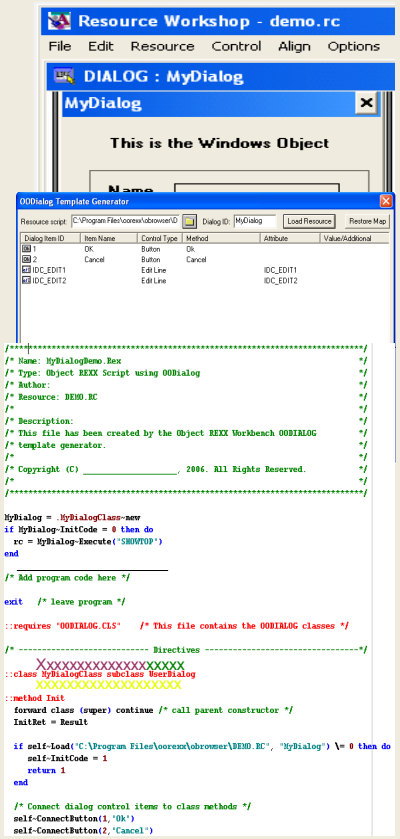
# Resource Workshop development cycle




- Design Windows GUI Object
- Run OODialog Template Generator
- Generator creates Code Template

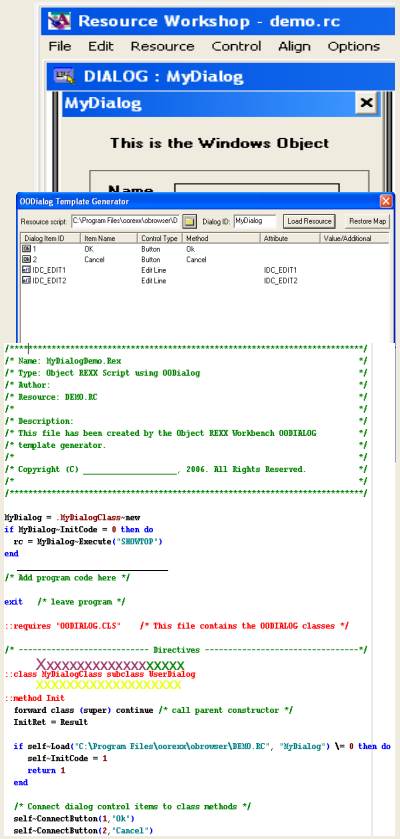
```
/**************************************************************************/
/* Name:  MyDialogDemo.Rex                                                */
/* Type:  Object REXX Script using OODialog                               */
/* Author:                                                                */
/* Resource:  DEMO.RC                                                     */
/*                                                                        */
/* Description:                                                           */
/* This file has been created by the Object REXX Workbench OODIALOG       */
/* template generator.                                                    */
/*                                                                        */
/* Copyright (C) _____, 2006. All Rights Reserved.                        */
/*                                                                        */
/**************************************************************************/

MyDialog = .MyDialogClass~new
if MyDialog~InitCode = 0 then do
   rc = MyDialog~Execute("SHOWTOP")
end

/* Add program code here */

exit   /* leave program */

::requires "OODIALOG.CLS"     /* This file contains the OODIALOG classes */

/* ----------------------------- Directives ---------------------------------*/

::class MyDialogClass subclass UserDialog

::method Init
   forward class (super) continue /* call parent constructor */
   InitRet = Result

   if self~Load("C:\Program Files\oorexx\obrowser\DEMO.RC", "MyDialog") \= 0 then do
      self~InitCode = 1
      return 1
   end

   /* Connect dialog control items to class methods */
   self~ConnectButton(1,"Ok")
   self~ConnectButton(2,"Cancel")
```

# Resource Workshop development cycle




- Design Windows GUI Object
- Run OODialog Template Generator
- Generator creates Code Template
- You amend Code to add functionality

# Resource Workshop development cycle




- Design Windows GUI Object
- Run OODialog Template Generator
- Generator creates Code Template
- You amend Code to add functionality
- Now you can't amend the GUI!

# Problems with the Resource Workshop way of developing OODialog GUIs

# Resource Workshop problems

- Hard to amend GUI

# Resource Workshop problems

- Hard to amend GUI
- Difficult to debug when goes wrong

# Resource Workshop problems

- Hard to amend GUI
- Difficult to debug when goes wrong
- Never get to understand how OODialog classes work

# Resource Workshop problems

- Hard to amend GUI
- Difficult to debug when goes wrong
- Never get to understand how OODialog classes work
- It's not there in ooRexx!

# Resource Workshop problems

- Hard to amend GUI
- Difficult to debug when goes wrong
- Never get to understand how OODialog classes work
- It's not there in ooRexx!
- Solution – Create a Static template

# Resource Workshop problems

- Hard to amend GUI
- Difficult to debug when goes wrong
- Never get to understand how OODialog classes work
- It's not there in ooRexx!
- Solution – Create a Static template
- Template for subclassing UserDialog class covers most bases

# What does a UserDialog Subclass look like?

# Anatomy of a UserDialog Subclass

Init

- Init method run when you create a new instance of an ooRexx object

# Anatomy of a UserDialog Subclass

Init

- Init method run when you create a new instance of an ooRexx object
- `self~Init:super` runs UserDialog Init Method

Could equally well be

```
forward class (Super) continue
```

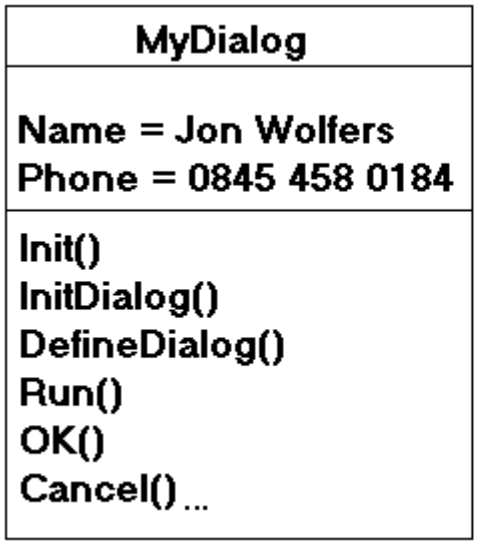
# Anatomy of a UserDialog Subclass

Init

- Init method run when you create a new instance of an ooRexx object
- `self~Init:super` runs UserDialog Init Method
- `Self~Create` or `~CreateCenter` creates the Windows Object

# The ooRexx & Windows Objects

Init
- Objects

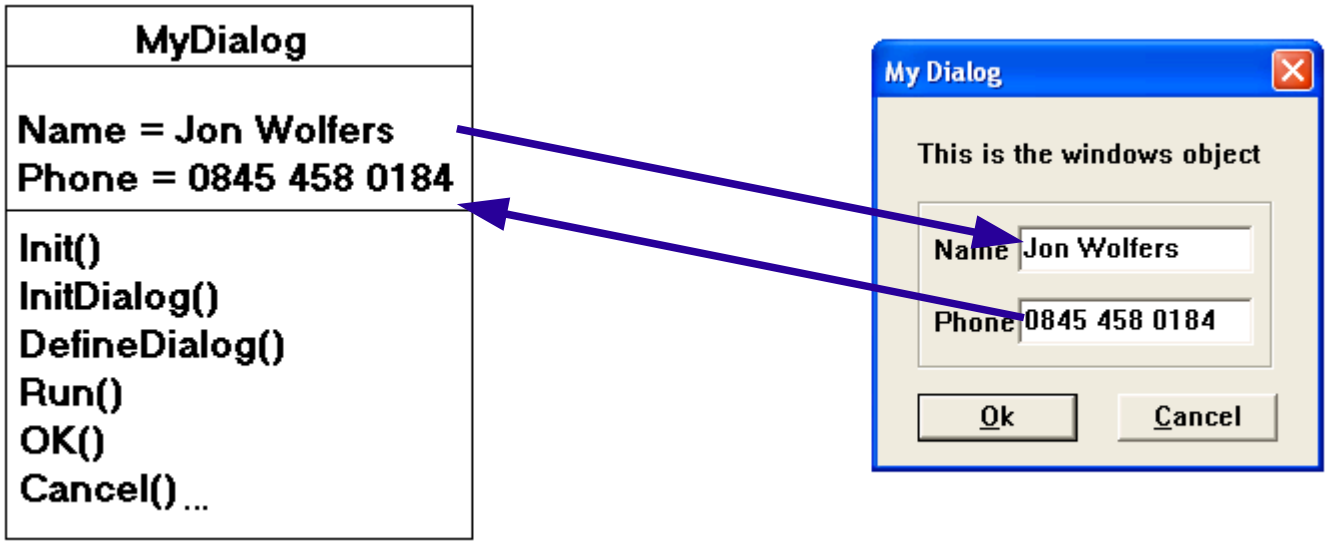




- OoRexx Object exists before & after Windows Object

# The ooRexx & Windows Objects

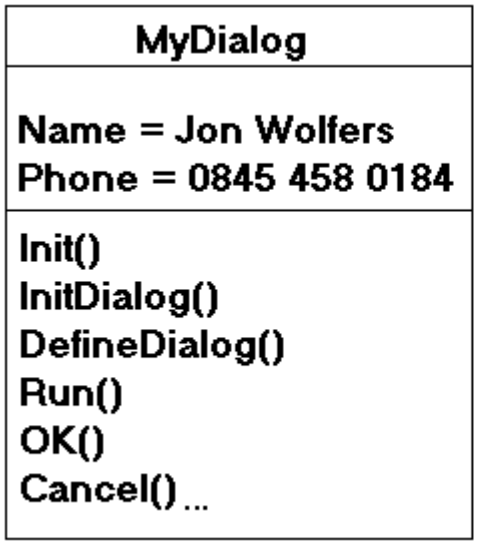
Init
- Objects

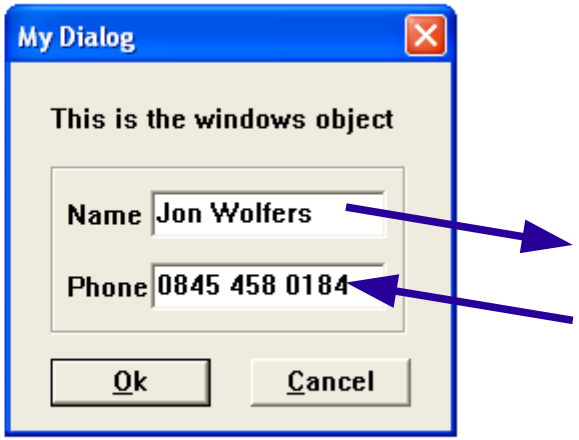



- OoRexx Object exists before & after Windows Object
- `self~GetData` & `~SetData` transfer data between them

# The ooRexx & Windows Objects

Init
- Objects




- OoRexx Object exists before & after Windows Object
- `self~GetData` & `~SetData` transfer data between them
- Possible to interact with Windows object Directly

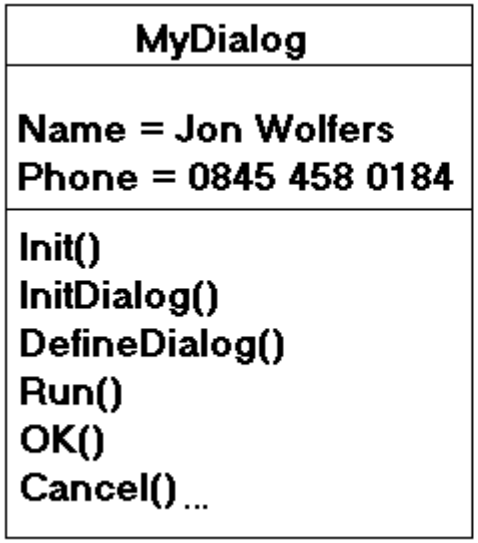
# Anatomy of a UserDialog Subclass

Init

- Init method run when you create a new instance of an ooRexx object
- `self~Init:super` runs UserDialog Init Method
- `Self~Create` or `~CreateCenter` creates the Windows Object
- Establish Connections in init method

# OODialog Connections

Init
Connections

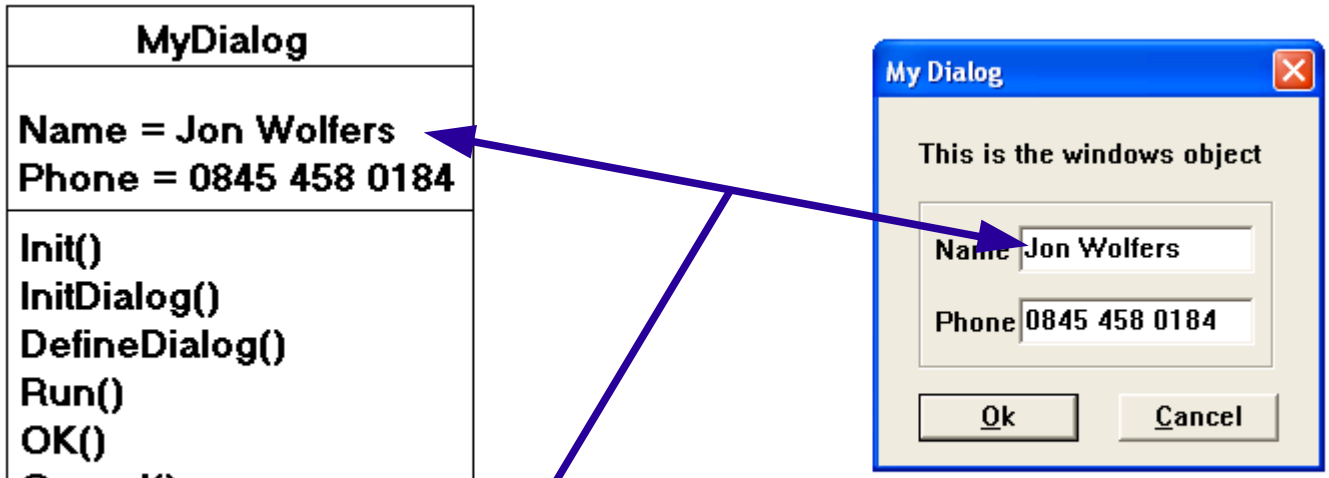




- Two types of Connections:

# OODialog Connections

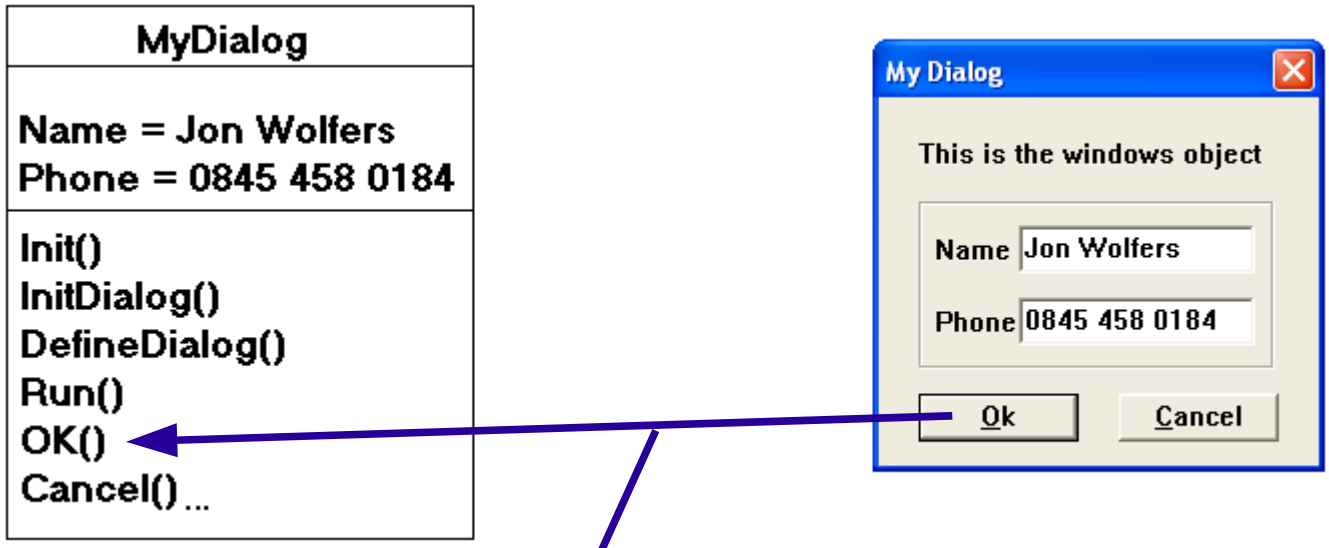
Init
Connections




- Two types of Connections:
  - **Data Connection**

# OODialog Connections

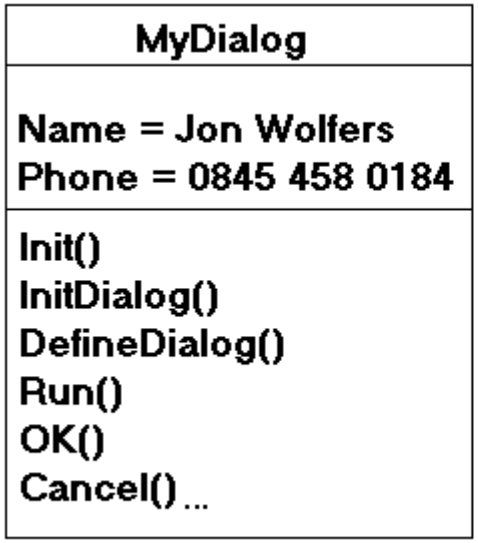
Init
Connections




- Two types of Connections:
  - **Data Connection**
  - **Event Notify Connection**

# OODialog Connections

Init
Connections





- Two types of Connections:
  - **Data Connection**
  - **Event Notify Connection**
- Many are added implicitly

# Anatomy of a UserDialog Subclass

Init

**DefineDialog**

- Where we add controls to our Dialog

# Anatomy of a UserDialog Subclass

Init

**DefineDialog**

- Where we add controls to our Dialog
- Called Automatically by ~Create

# Anatomy of a UserDialog Subclass

Init

**DefineDialog**

- Where we add controls to our Dialog
- Called Automatically by ~Create
- Use Add... Methods

# Anatomy of Add... Methods

Init

**DefineDialog**

└► AddMethods

- self~Add**Control**(**id**,**x**,**y**,**cx**,**cy**,**text**, **Options**,**MsgToRaise**,**Attribute**)
- Not all parameters required for each type of control

# Anatomy of Add... Methods

Init

DefineDialog

↳ AddMethods

- self~Add**Control**(**id**,**x**,**y**,**cx**,**cy**,**text**, **Options**,**MsgToRaise**,**Attribute**)
- What sort of a widget ie:
  - Button
  - ListControl
  - ComboBox
  - Tree Control
  - EntryLine
  - Static Text
  - Frame etc...

# Anatomy of Add... Methods

Init

DefineDialog

↳ AddMethods

- self~AddControl(id,x,y,cx,cy,text, Options,MsgToRaise,Attribute)
- Unique Identifier for this control
- Nos 1 – 9 reserved
  - 1 for OK Button
  - 2 for Cancel Button
  - 9 for Help Button
- IDs may be symbolic

# Anatomy of Add... Methods

Init

DefineDialog

└► AddMethods

- self~Add**Control**(**id**,**x**,**y**,**cx**,**cy**,**text**, **Options**,**MsgToRaise**,**Attribute**)
- Coordinates of control relative to dialog
- Measured in 'Dialog Units'
- Dialog Units depend on screen resolution & Active fonts
- A runtime conversion to pixels may be made using UserDialog attributes FactorX & FactorY

# Anatomy of Add... Methods

Init

DefineDialog

↳ AddMethods

- self~Add**Control**(**id**,**x**,**y**,**cx**,**cy**,**text**, **Options**,**MsgToRaise**,**Attribute**)
- Control Width & Height in Dialog Units

# Anatomy of Add... Methods

Init

DefineDialog

↳ AddMethods

- self~Add**Control**(**id**,**x**,**y**,**cx**,**cy**,**text**, **Options**,**MsgToRaise**,**Attribute**)
- Title associated with Control where appropriate
  - Button Text
  - Initial value of an entry line
  - Static Text
  - Title of a Radio Button/Checkbox ...

# Anatomy of Add... Methods

Init

DefineDialog

↳ AddMethods

- self~Add**Control**(**id**,**x**,**y**,**cx**,**cy**,**text**, **Options**,**MsgToRaise**,**Attribute**)
- Controls style or behaviour
  - Make Pushbutton Default
  - Size of Icons in a list
  - etc..

# Anatomy of Add... Methods

Init

DefineDialog

↳ AddMethods

- self~Add**Control**(**id**,**x**,**y**,**cx**,**cy**,**text**, **Options**,**MsgToRaise**,**Attribute**)
- Creates a Notify-Connection for Buttons

# Anatomy of Add... Methods

Init

**DefineDialog**

└► AddMethods

- self~Add**Control**(**id**,**x**,**y**,**cx**,**cy**,**text**, **Options**,**MsgToRaise**,**Attribute**)
- Initiates connection to (& creates) an attribute for Control Title or Selected element

# Anatomy of a UserDialog Subclass

Init

**DefineDialog**

- Where we add controls to our Dialog
- Called Automatically by ~Create
- Use Add... Methods

# Anatomy of a UserDialog Subclass

Init

DefineDialog

- Where we add controls to our Dialog
- Called Automatically by ~Create
- Use Add... Methods
- Realisation of a Dialog is facilitated Using My DlgArea Class

# Anatomy of a UserDialog Subclass

Init

DefineDialog

**InitDialog**

- Run After windows object Created but before it is populated or displayed

# Anatomy of a UserDialog Subclass

Init

DefineDialog

**InitDialog**

- Run After windows object Created but before it is populated or displayed
- Start with `self~InitDialog:Super`

# Anatomy of a UserDialog Subclass

Init

DefineDialog

**InitDialog**

- Run After windows object Created but before it is populated or displayed
- Start with `self~InitDialog:Super`
- Populate Combo Boxes and lists, add columns to reports

# Anatomy of a UserDialog Subclass

Init

DefineDialog

**InitDialog**

- Run After windows object Created but before it is populated or displayed
- Start with `self~InitDialog:Super`
- Populate Combo Boxes and lists, add columns to reports
- A SetData is run after this method

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

**Run**

- 'Run' method runs after SetData
  (Not Asynchronous Dialogs)

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

**Run**

- 'Run' method runs after SetData
  (Not Asynchronous Dialogs)
- Dialog is shown but not yet active

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

**Run**

- 'Run' method runs after SetData
  (Not Asynchronous Dialogs)
- Dialog is shown but not yet active
- Must contain `self~Run:Super`

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

**Run**

- 'Run' method runs after SetData
  (Not Asynchronous Dialogs)
- Dialog is shown but not yet active
- Must contain `self~Run:Super`
- Possible uses:
  - Disable/Enable Buttons
  - Set Initial Values for checkboxes
  - Long initialisation with dialog visible
  - Set Control Colours etc...

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

Run

**OK/Cancel**

- One may subclass OK and/or Cancel methods

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

Run

**OK/Cancel**

- One may subclass OK and/or Cancel methods
- `self~finished` regulates whether dialog closes

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

Run

**OK/Cancel**

- One may subclass OK and/or Cancel methods
- `self~finished` regulates whether dialog closes
- One must call super class

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

Run

OK/Cancel

**Validate**

- Called by OK Method

# Anatomy of a UserDialog Subclass

Init

DefineDialog

InitDialog

Run

OK/Cancel

**Validate**

- Called by OK Method
- Return
  - 0 to prevent dialog closing
  - 1 to allow dialog to close

# Anatomy of a UserDialog Subclass



- Your class should be headed

```
::class MyDialog SubClass UserDialog
```

# Anatomy of a UserDialog Subclass



- Your class should be headed
  ```
  ::class MyDialog SubClass UserDialog
  ```
- You may need to inherit from
  - AdvancedControls
  - MessageExtensions or
  - VirtualKeyCodes (requires winSystm.cls)

# Anatomy of a UserDialog Subclass



- Your class should be headed
  ```
  ::class MyDialog SubClass UserDialog
  ```
- You may need to inherit from
  - AdvancedControls
  - MessageExtensions or
  - VirtualKeyCodes (requires winSystm.cls)
- Template gives guidance on inheritance.

# Anatomy of a UserDialog Subclass



- `Dlg = .MyDialog~new`

# Anatomy of a UserDialog Subclass



- `Dlg = .MyDialog~new`
- Can pass a stem variable to Init

# Anatomy of a UserDialog Subclass



- `Dlg = .MyDialog~new`
- Can pass a stem variable to Init
- Now we can access attributes i.e.
  - `dlg~UserName='Jon'`

# Anatomy of a UserDialog Subclass



- `Dlg = .MyDialog~new`
- Can pass a stem variable to Init
- Now we can access attributes i.e.
  - `dlg~UserName='Jon'`
- `dlg~execute('showtop')`
  (could be ExecuteAsync)

# Anatomy of a UserDialog Subclass



- `Dlg = .MyDialog~new`
- Can pass a stem variable to Init
- Now we can access attributes i.e.
  - `dlg~UserName='Jon'`
- `dlg~execute('showtop')`
  (could be ExecuteAsync)
- Now the Dialog runs till OK or Cancel

# Anatomy of a UserDialog Subclass



- `Dlg = .MyDialog~new`
- Can pass a stem variable to Init
- Now we can access attributes i.e.
  - `dlg~UserName='Jon'`
- `dlg~execute('showtop')`
  (could be ExecuteAsync)
- Now the Dialog runs till OK or Cancel
- `dlg~deInstall`

# Anatomy of a UserDialog Subclass



- `Dlg = .MyDialog~new`
- Can pass a stem variable to Init
- Now we can access attributes i.e.
  - `dlg~UserName='Jon'`
- `dlg~execute('showtop')`
  (could be ExecuteAsync)
- Now the Dialog runs till OK or Cancel
- `dlg~deInstall`
- OoRexx object still available!

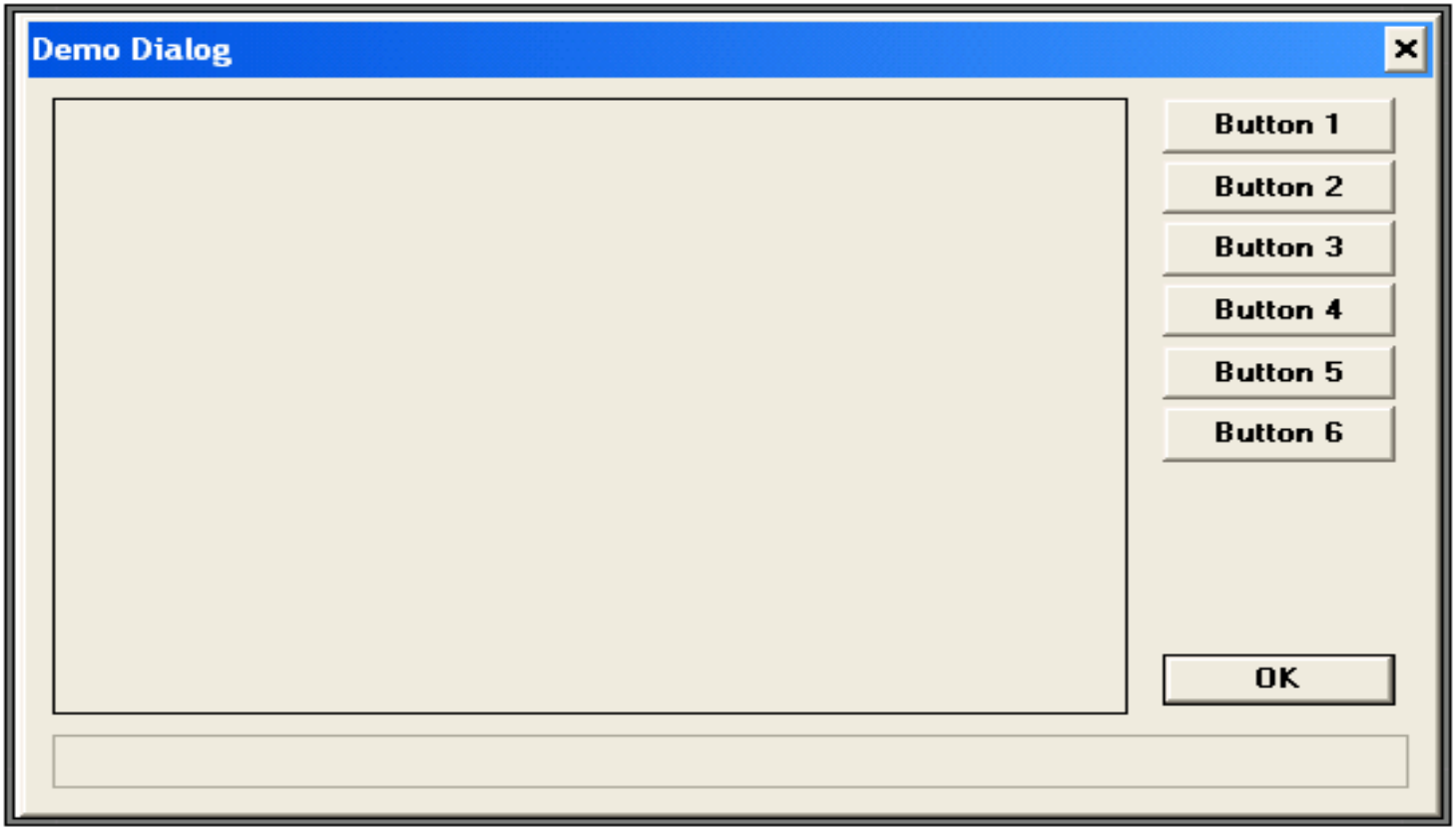
# Anatomy of a UserDialog Subclass

**That's all there is to subclassing the OODialog UserDialog class!**

# Using The Template

A live demo!

# Using The Template

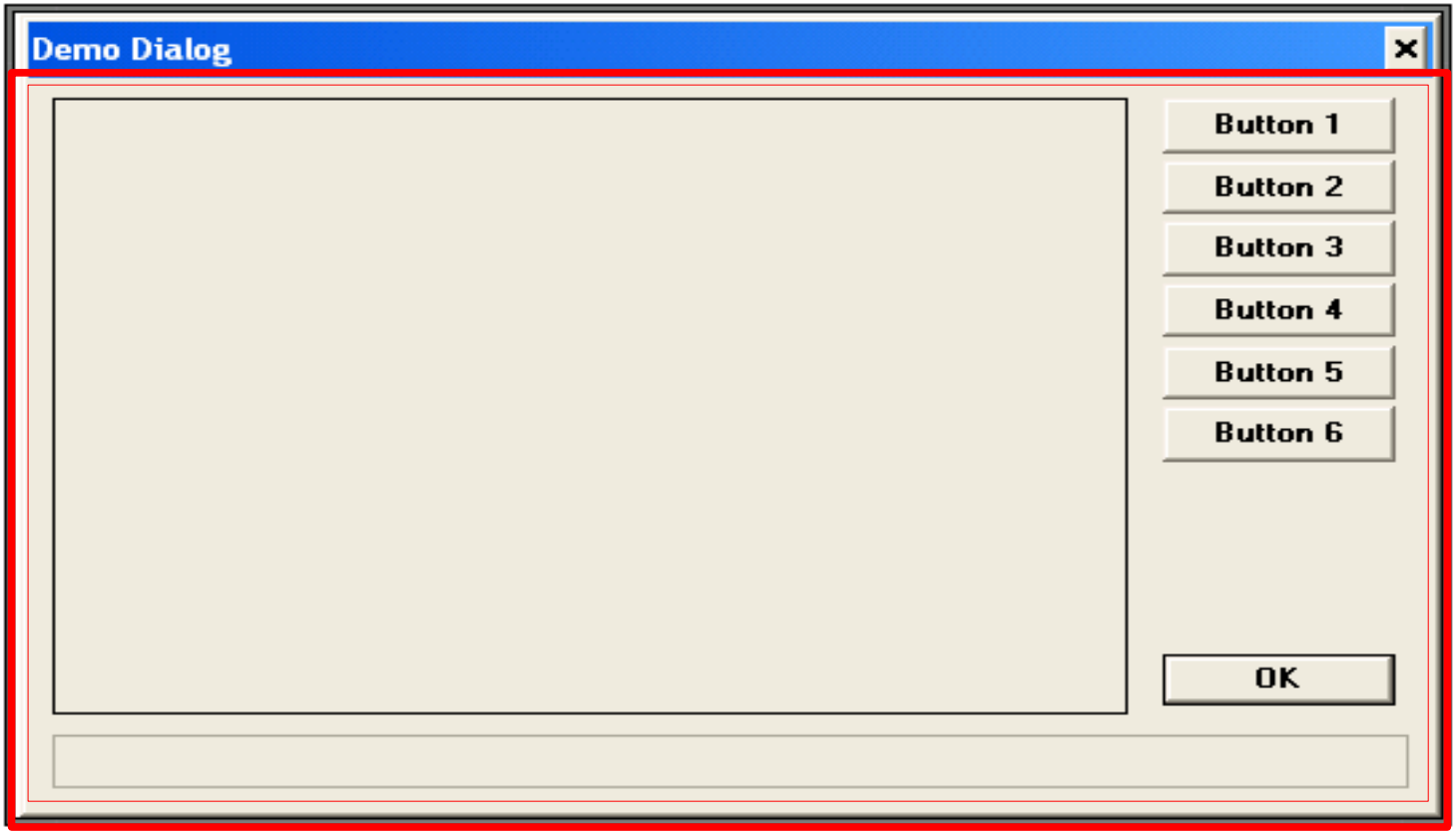



Template

- What we want to achieve

# Using The Template

DlgAreaU



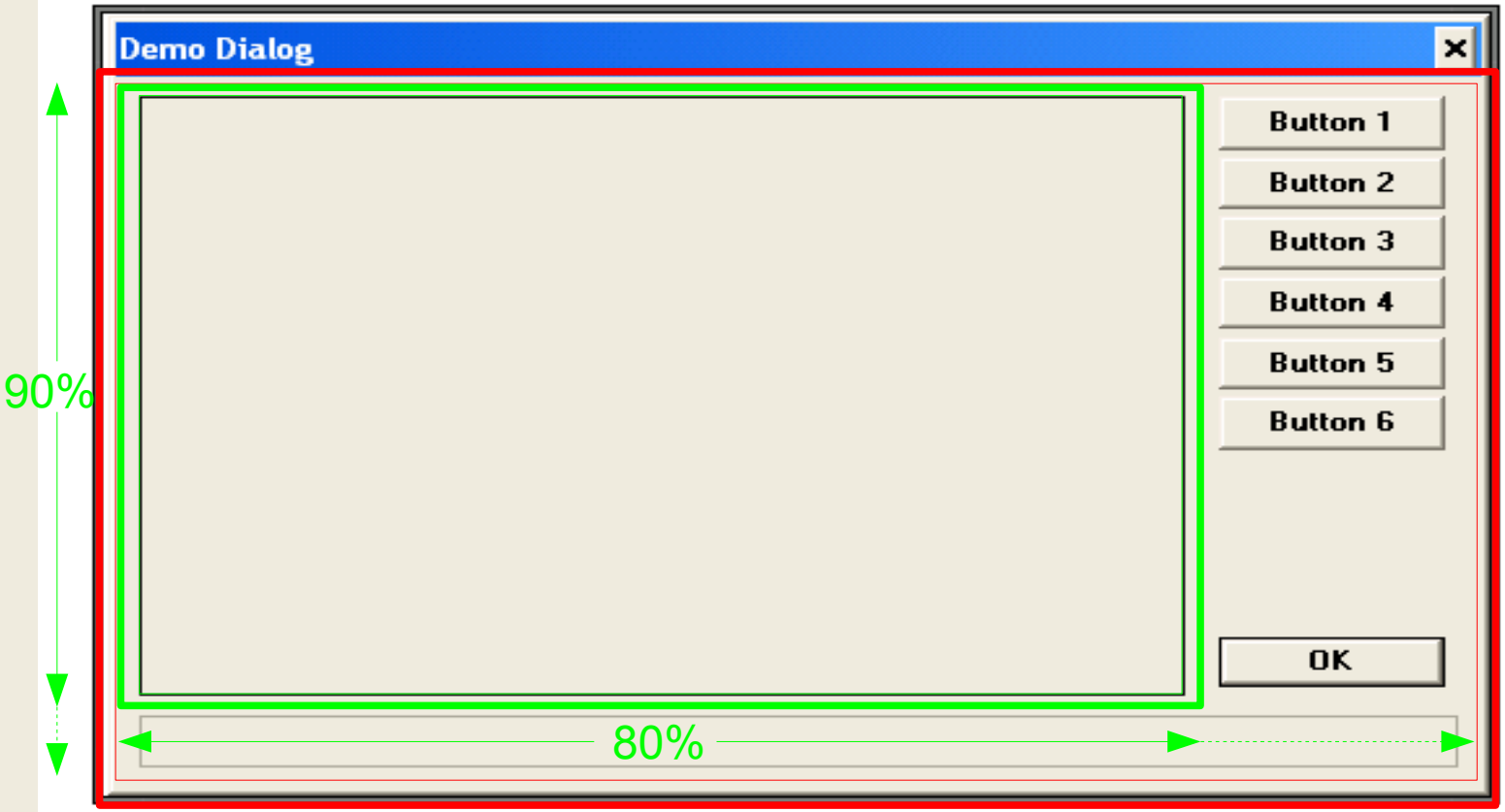

Template

**DlgAreaU class – coterminous with the dialog**

# Using The Template

DlgAreaU

DlgArea e




Template

**DlgArea created within inner margin of DlgAreaU**

# Using The Template

DlgAreaU

DlgArea e

DlgArea b


Template

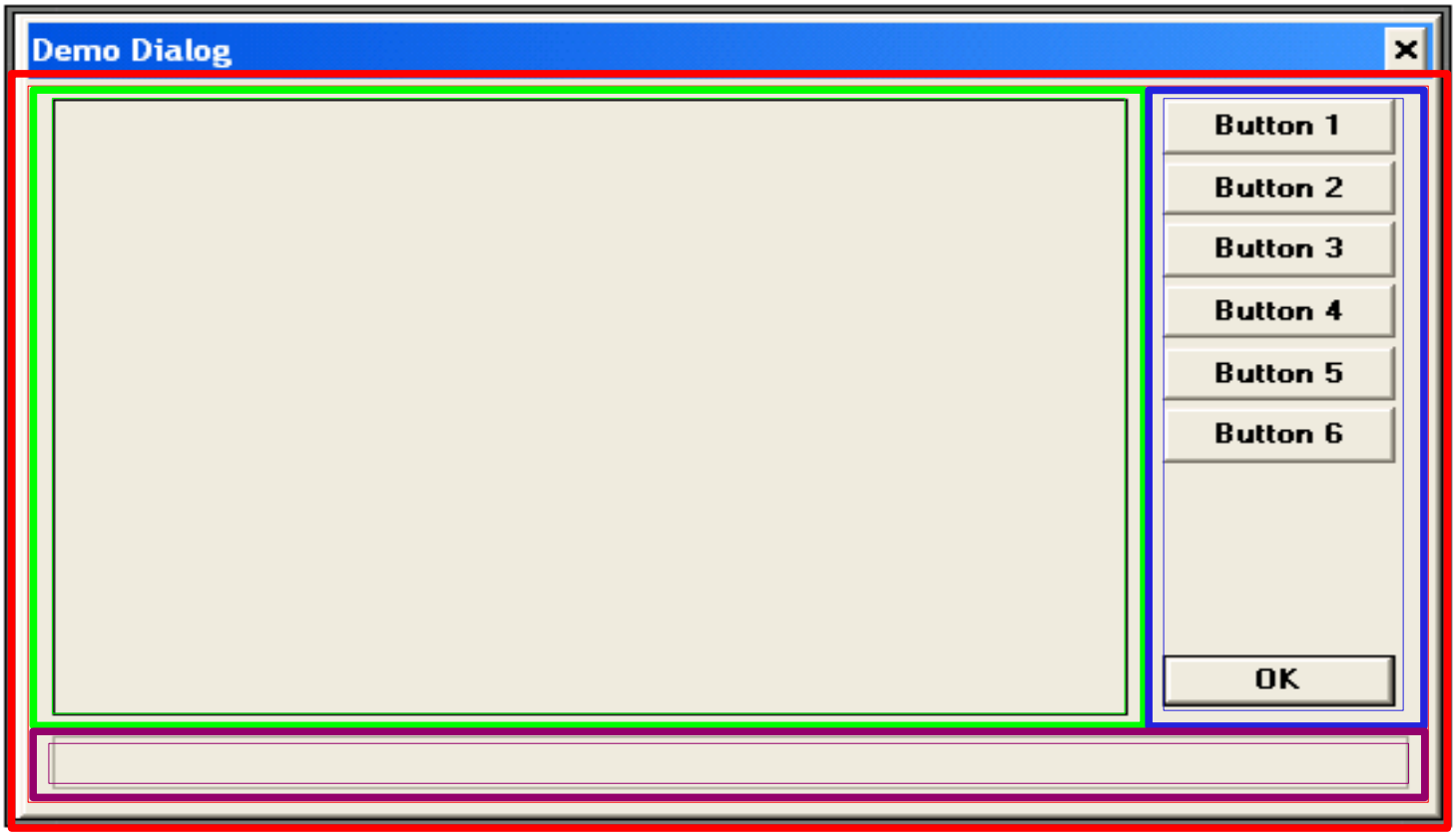
# Using The Template

DlgAreaU

DlgArea e

DlgArea b

DlgArea s




Template

# The end?

Scripts of interest:

- Minimum code to run a dialog
- Context Menu (Sort of)
- SubClassing GetData & SetData
- Resizable Dialog

Template
Demo Code