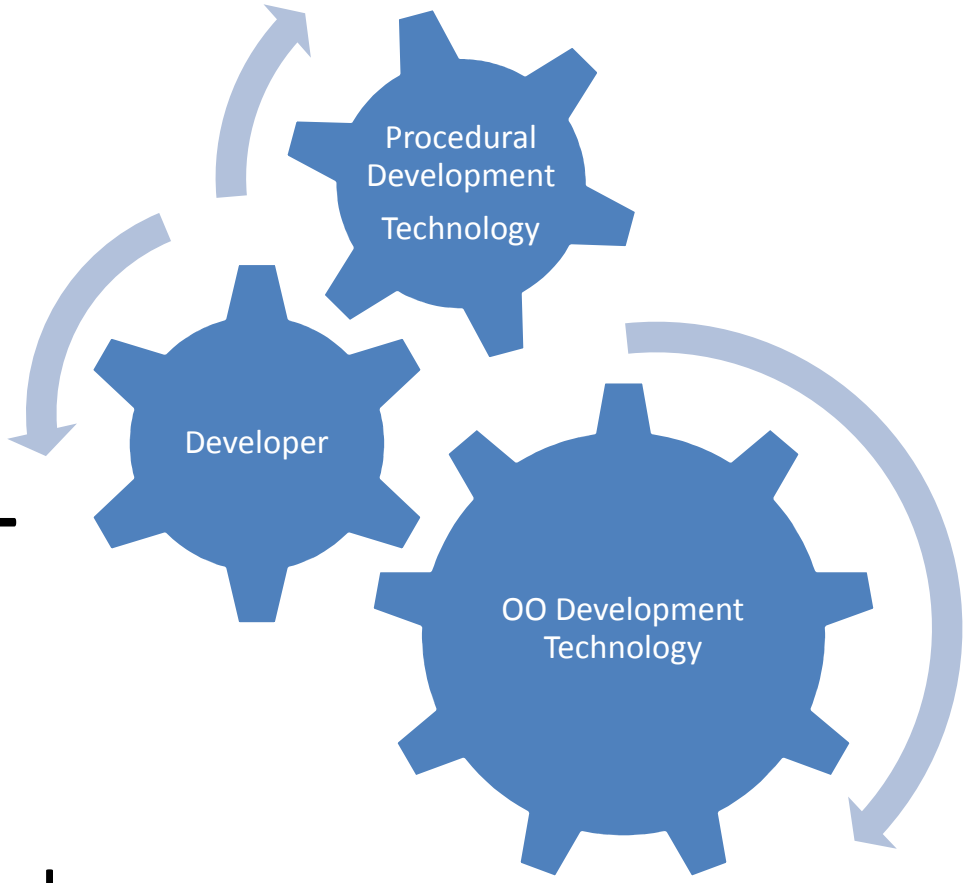
OO Language Concepts for Procedural 4GL Developers



Timothy D. Kuehn

Senior Consultant

TDK Consulting Services Inc

tim.kuehn@gmail.com

OO Language Concepts for 4GL Developers
What is This Presentation About?

Intended as a introduction for procedural developers looking to learn about OO programming.

- OO Language Element Definitions
- Survey of OO language elements and talk about how they are paralleled by procedural language elements.
- Have more material than I can cover in one session
- All code samples in these slides are available.

OO Language Element Definitions

OO Language Concepts for 4GL Developers
OO Language Element Definitions

What is a Class?

A class is a set of methods and properties grouped together in a single definition file for the purpose of accomplishing a task.

```
CLASS presentation.classes.ClassWrapper:
   /* contents */
END CLASS.
```

Note: Make sure your class names distinct from db table field names!

OO Language Concepts for 4GL Developers
OO Language Element Definitions

What is a Method?

A method is a function that is encased in a class.

```
CLASS presentation.classes.MethodWrapper:

METHOD CHARACTER MethodName():
    /* stuff */
RETURN("character string").
END METHOD.

END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Element Definitions

What is a Property?

A property is a variable encased in a class that can be directly accessed by objects and programs outside of the class.

```
CLASS presentation.classes.PropertyWrapper:
DEFINE PROPERTY character-property AS CHARACTER NO-UNDO
        GET.
        SET.
END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Element Definitions

What is an Object?

An object is the actualization of a class, with all of its default data and any actions that can be performed by its functions.

class = definition of how to do something.
object = actualization of that definition in a distinct setting.

In other words:

- classes are to OO what a persistent or super procedure files are to procedures,
- objects are to OO what procedure instances are to persistent and super procedures.
- Nothing in the OO world corresponds to straight ".p" procedures.

Survey of OOABL Language Elements

OO Language Concepts for 4GL Developers
OO Language Elements – Making an Object

How does one make an object?

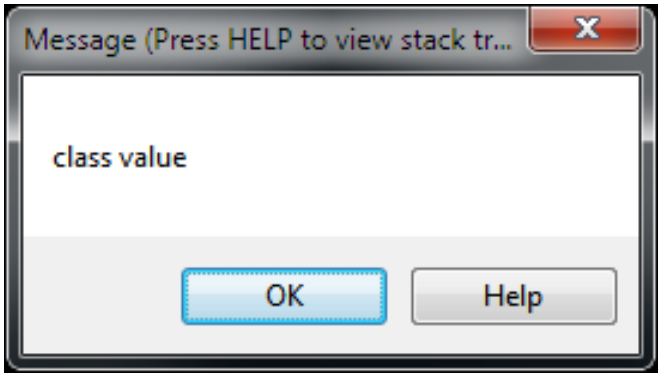
```
Define variable oClass as presentation.classes.ClassWrapper
        no-undo.
oClass = NEW presentation.classes.ClassWrapper().
```

OO Language Concepts for 4GL Developers
OO Language Elements – Classes and Methods

```
/* presentation/classes/Method.cls */

CLASS presentation.classes.Method:

DEFINE VARIABLE ExampleVar AS CHARACTER NO-UNDO.

METHOD VOID SetVariable(newvalue AS CHARACTER):
ExampleVar = newvalue.
END METHOD.

METHOD CHARACTER GetVariable():
RETURN(examplevar).
END METHOD.

END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements – Classes and Methods

```
/* presentation/examples/example01-method.p  */

DEFINE VARIABLE oClass AS presentation.classes.ExampleMethod NO-UNDO.

oClass = NEW presentation.classes.ExampleMethod().
oClass:SetVariable("class value").
MESSAGE oClass:GetVariable() VIEW-AS ALERT-BOX.
```

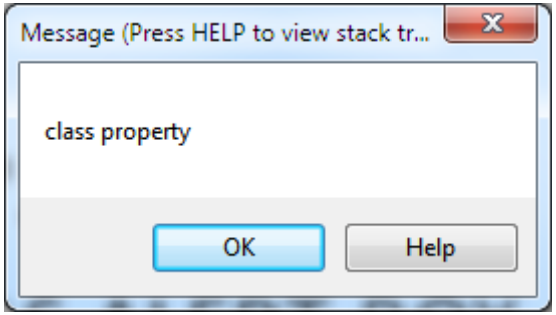


OO Language Concepts for 4GL Developers
OO Language Elements – Classes and Properties

```
/* presentation/classes/ExampleProperty.cls */

CLASS presentation.classes.ExampleProperty:

DEFINE PUBLIC PROPERTY ExampleProperty AS CHARACTER NO-UNDO
   GET.
   SET.

END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements – Classes and Properties

```
/* presentation/classes/example02-property.p */

DEFINE VARIABLE oClass AS presentation.classes.ExampleProperty NO-UNDO.

oClass = NEW presentation.classes.ExampleProperty().

oClass:ExampleProperty = "class property".
MESSAGE oClass:ExampleProperty VIEW-AS ALERT-BOX.
```



OO Language Concepts for 4GL Developers
OO Language Elements – Using.*

```
/* presentation/classes/example03-property.p */

DEFINE VARIABLE oClass AS presentation.classes.Property NO-UNDO.
oClass = NEW presentation.classes.Property().
```

Newer, Faster Way:

```
USING presentation.classes.*.
DEFINE VARIABLE oClass AS Property NO-UNDO.
oClass = NEW Property().
```

OO Language Concepts for 4GL Developers
OO Language Elements – PolyMorphism

```
 /* presentation/classes/SignatureMatch.cls */
CLASS presentation.classes.SignatureMatch:
DEFINE VARIABLE ExampleVar AS CHARACTER NO-UNDO.

METHOD VOID SetVariable(prefixvalue AS INTEGER,
                        suffixvalue  AS INTEGER):
SetVariable(STRING(prefixvalue, '999') + "." + STRING(suffixvalue, '999')).
END METHOD.

METHOD VOID SetVariable
        (newvalue AS CHARACTER):
ExampleVar = newvalue.
END METHOD.
```

```
METHOD CHARACTER GetVariable():
RETURN(ExampleVar).
END METHOD.

END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements – Method Signatures

Method Signatures provide a way to describe interface functionality that allows for calls from different contexts.

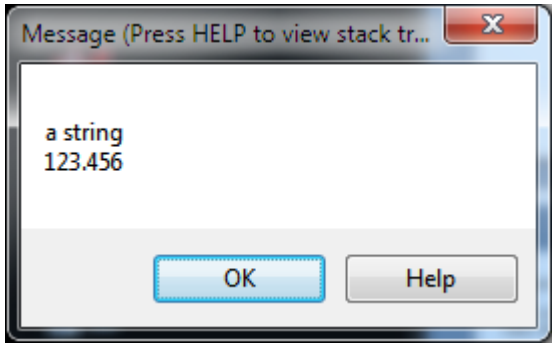
```
ProductionOrder:Create (sales-order-number,
                        backorder-number).

ProductionOrder:Create(kv-sales-order).
```

It's also possible to pass objects as parameters:

```
ProductionOrder:Create(oSalesOrder).
```

OO Language Concepts for 4GL Developers
OO Language Elements – Method Signatures

```
/* presentation/examples/example04-signatures.p */
USING presentation.classes.*

DEFINE VARIABLE oClass1 AS SignatureMatch NO-UNDO.
DEFINE VARIABLE oClass2 AS SignatureMatch NO-UNDO.

oClass1 = NEW SignatureMatch().
oClass2 = NEW SignatureMatch().

oClass1:SetVariable("a string").
oClass2:SetVariable(123, 456).

MESSAGE oClass1:GetVariable()   SKIP
        oClass2:GetVariable()
    VIEW-AS ALERT-BOX.
```

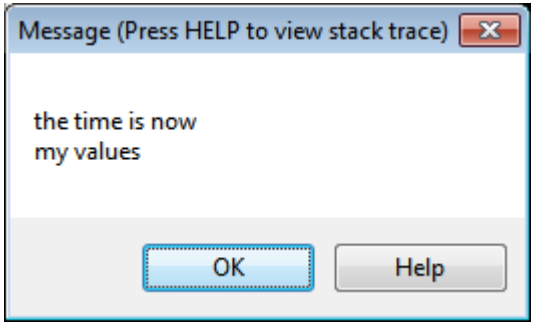


OO Language Concepts for 4GL Developers
OO Language Elements – Constructors

```
/* presentation/classes/ConstructorClass.cls */

CLASS presentation.classes.ConstructorClass:

DEFINE VARIABLE cur-value AS CHARACTER.

CONSTRUCTOR ConstructorClass():
ASSIGN cur-value = "the time is now".
END CONSTRUCTOR.

CONSTRUCTOR ConstructorClass(new-value AS CHARACTER):
ASSIGN cur-value = new-value.
END CONSTRUCTOR.

METHOD CHARACTER GetVariable():
RETURN(cur-value).
END METHOD.

END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements – Constructors

```
/* presentation/examples/example05-constructor.p */

USING presentation.classes.*.

DEFINE VARIABLE aClass AS ConstructorClass NO-UNDO.
DEFINE VARIABLE bClass AS ConstructorClass NO-UNDO.

aClass = NEW ConstructorClass().
bClass = NEW ConstructorClass("my values").

MESSAGE aclass:GetVariable()    SKIP
        bclass:GetVariable()
VIEW-AS ALERT-BOX.
```



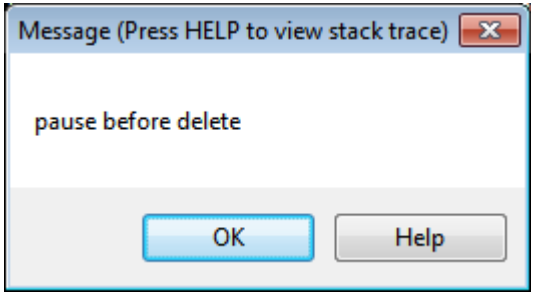
OO Language Concepts for 4GL Developers
OO Language Elements – Destructors

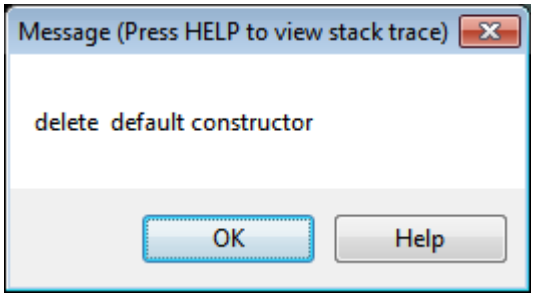
```
/* presentation.classes.DestructorClass */
CLASS presentation.classes.DestructorClass:
DEFINE VARIABLE cur-name AS CHARACTER.

DESTRUCTOR DestructorClass():
MESSAGE "delete " cur-name
   VIEW-AS ALERT-BOX.
END DESTRUCTOR.
```

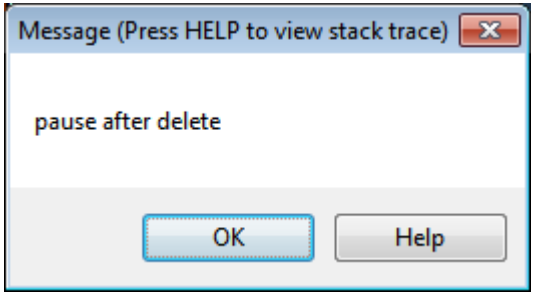
```
CONSTRUCTOR DestructorClass():
ASSIGN cur-name = "default constructor".
END CONSTRUCTOR.

CONSTRUCTOR DestructorClass(cur-parm AS
CHARACTER):
ASSIGN cur-name = cur-parm.
END CONSTRUCTOR.

END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements – Destructors

```
/* presentations/examples/example06a-destructor.p    */

USING presentation.classes.*.
DEFINE VARIABLE oClass AS DestructorClass NO-UNDO.

oClass = NEW DestructorClass().

MESSAGE "pause before delete" VIEW-AS ALERT-BOX.
DELETE OBJECT oClass.

MESSAGE "pause after delete" VIEW-AS ALERT-BOX.
```







OO Language Concepts for 4GL Developers
OO Language Elements – Garbage Collection

```
USING presentation.classes.*.
DEFINE VARIABLE oClass AS DestructorClass NO-UNDO.
 oClass = NEW DestructorClass().
RETURN.
```

- This looks like a classic memory leak.
- It's also a common practice in the OO world.
- Why?
- Garbage Collection.

Garbage collection frees programmers from explicitly manage object memory allocation by deleting objects when they're no longer in use.

In the ABL world, "no longer in use" = "an object with no references to it".

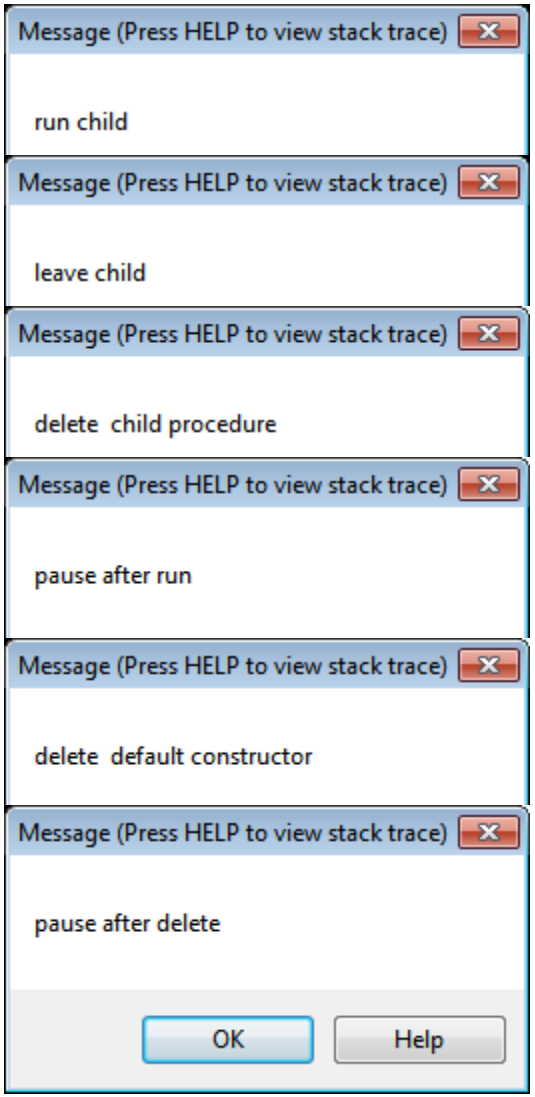
Read KB P155379 for more information.

OO Language Concepts for 4GL Developers
OO Language Elements – Garbage Collection

```
/* presentations/examples/example07a-garbage-collection.p    */

USING presentation.classes.*.
DEFINE VARIABLE oClass AS DestructorClass NO-UNDO.

oClass = NEW DestructorClass().

MESSAGE "run child" VIEW-AS ALERT-BOX.
RUN presentation/examples/example07b-garbage-collection.p.

MESSAGE "pause after run" VIEW-AS ALERT-BOX.

DELETE OBJECT oClass.

MESSAGE "pause after delete" VIEW-AS ALERT-BOX.
```

OO Language Concepts for 4GL Developers
OO Language Elements – Garbage Collection

```
/* presentations/examples/example07b-garbage-collection.p */

USING presentation.classes.*.
DEFINE VARIABLE oClass AS DestructorClass NO-UNDO.

oClass = NEW DestructorClass("child procedure").

MESSAGE "leave child"
   VIEW-AS ALERT-BOX.
```



OO Language Concepts for 4GL Developers
OO Language Elements – Access Modifiers

What does Private, Protected, Public mean?

They are **access modifiers** which help implement encapsulation (or information hiding). They tell the compiler what kind of access is allowed for the method, variable, buffer, temp-table, query, or other item that's being defined.

- Private – access is restricted to the current class.
- Protected – access is restricted to the current class and any sub-classes.
- Public – access is given to anything which has a reference to the class.

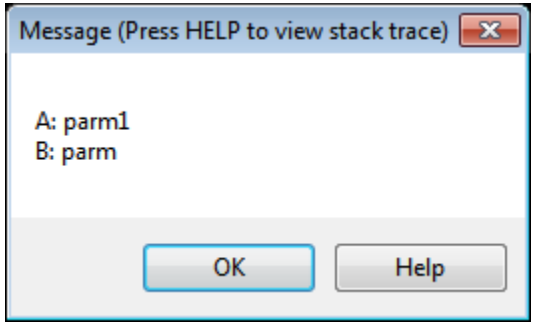
Public

OO Language Concepts for 4GL Developers
OO Language Elements – Access Modifiers

```
/* presentation/classes/AccessClass.cls  */
CLASS presentation.classes.AccessClass:
DEFINE VARIABLE cur-char AS CHARACTER NO -UNDO.
DEFINE VARIABLE cur-int  AS INTEGER  NO-UNDO.

METHOD PUBLIC CHARACTER GetChar():
RETURN(cur-char).
END METHOD.

CONSTRUCTOR PUBLIC AccessClass():
THIS-OBJECT("parm1", 1).
END CONSTRUCTOR.
CONSTRUCTOR PUBLIC AccessClass
                (parm AS CHARACTER):
THIS-OBJECT(parm, 1).
END CONSTRUCTOR.

CONSTRUCTOR PRIVATE AccessClass(
          parm1 AS CHARACTER,
          parm2 AS INTEGER).

ASSIGN
   cur-char = parm1
   cur-int  = parm2.

END CONSTRUCTOR.
END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements – Access Modifiers

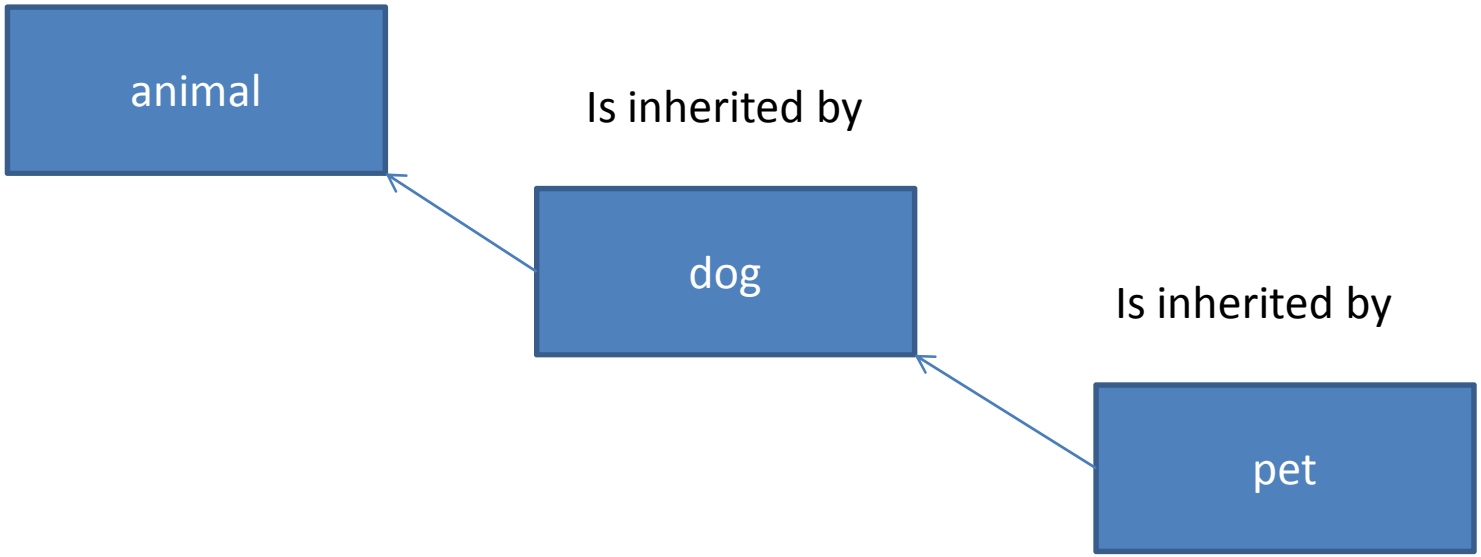
```
/* presentation/examples/example08-access.p */
USING presentation.classes.*.

DEFINE VARIABLE oClassA AS AccessClass NO-UNDO.
DEFINE VARIABLE oClassB AS AccessClass NO-UNDO.

oClassA = NEW AccessClass().
oClassB = NEW AccessClass("parm").

MESSAGE "A:" oClassA:Getchar()   SKIP
        "B:" oClassB:GetChar()
    VIEW-AS ALERT-BOX.
```



OO Language Concepts for 4GL Developers
OO Language Elements – Inheritance

What is Inheritance?

Inheritance is a way to compartmentalize and extend code. This is done by creating collections of attributes and behaviors in sub-classes that takes super-class functionality and extends it to do new things.



OO Language Concepts for 4GL Developers
OO Language Elements – Inheritance

Pet	SetName GetName
Dog	**Bark** **Fetch**
Animal	**SetFeedingTime** **GetFeedingTime**

OO Language Concepts for 4GL Developers
OO Language Elements – Static Members

Static Class Members:

- are scoped to the class, not a given single instance,
- have their own constructor,
- are able to reference other static members or object instances that it starts or are passed to it,
- are referenced using ClassName:MethodName() format.
- can be a method, property, variable, buffer, or any other class member type,

OO Language Concepts for 4GL Developers
OO Language Elements – Static Members

Static Class Members:

- are instantiated whenever any static member in the class is referenced, or a dynamic object instance is created,
- cannot call a “super” method.

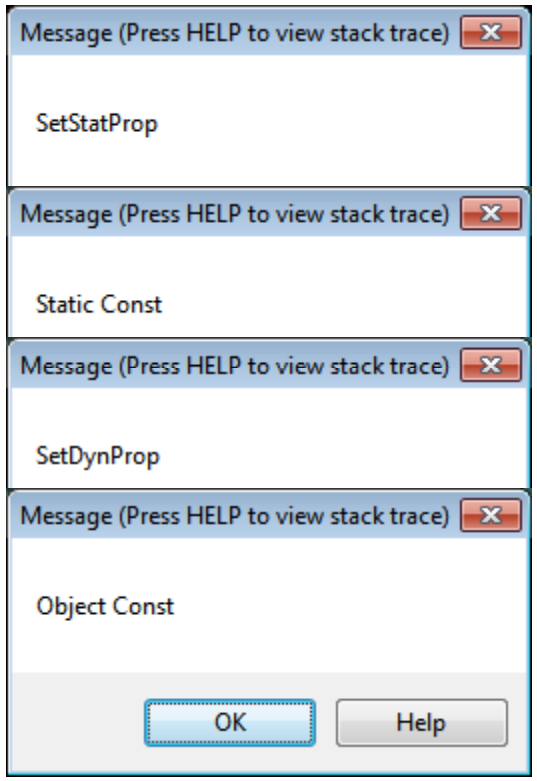
OO Language Concepts for 4GL Developers
OO Language Elements – Static Members

```
/* presentation/classes/StaticMembers.cls   */
CLASS presentation.classes.StaticMembers:
DEFINE PUBLIC STATIC PROPERTY stat-value AS CHARACTER NO-UNDO GET. SET.
DEFINE PUBLIC PROPERTY cls-value AS CHARACTER NO-UNDO GET. PRIVATE SET.

METHOD PUBLIC STATIC CHARACTER GetStatValue():
RETURN(stat-value).
END METHOD.
METHOD PUBLIC CHARACTER GetClsValue():
RETURN(cls-value).
END METHOD.
```

```
CONSTRUCTOR STATIC StaticMembers():
MESSAGE "Static Const" VIEW-AS ALERT-BOX.
END CONSTRUCTOR.

CONSTRUCTOR PUBLIC StaticMembers():
MESSAGE "Object Const" VIEW-AS ALERT-BOX.
ASSIGN cls-value = "cls".
END CONSTRUCTOR.

END CLASS.
```

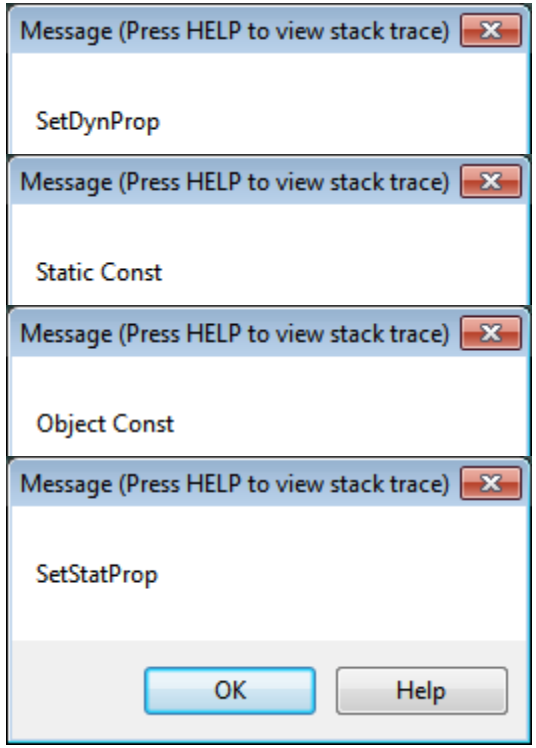
OO Language Concepts for 4GL Developers
OO Language Elements – Static Members

```
/* presentation/examples/example09a-static.p */
USING presentation.classes.*.

DEFINE VARIABLE oStat AS StaticMembers NO-UNDO.

MESSAGE "SetStatProp" VIEW-AS ALERT-BOX.
StaticMembers:stat-value = "value".

MESSAGE "SetDynProp" VIEW-AS ALERT-BOX.
oStat = NEW StaticMembers().
```



OO Language Concepts for 4GL Developers
OO Language Elements – Static Members

```
/* presentation/examples/example09b-static.p */

USING presentation.classes.*.

DEFINE VARIABLE oStat AS StaticMembers NO-UNDO.

MESSAGE "SetDynProp" VIEW-AS ALERT-BOX.
oStat = NEW StaticMembers().

MESSAGE "SetStatProp" VIEW-AS ALERT-BOX.
StaticMembers:stat-value = "value".
```



OO Language Concepts for 4GL Developers
OO Language Elements – Static Members

```
/* presentation/examples/example09c-static.p */

USING presentation.classes.*.

MESSAGE "set stat prop" VIEW-AS ALERT-BOX.
StaticMembers:stat-value = "value".

MESSAGE "get Stat Prop:" StaticMembers:GetStatValue()
   VIEW-AS ALERT-BOX.
```