OO Language Concepts for Procedural 4GL Developers

Timothy D. Kuehn

Senior OpenEdge Consultant

TDK Consulting Services Inc

timk@tdkcs.ca tim.kuehn@gmail.com

Ph 519-576-8100 Cell: 519-781-0081

Note to Reader

This document was part of a workshop given at PUG Challenge Americas 2013. The associated code samples, exercises, quizzes, and answers and are part of an "Introduction to OO" course offered by TDK Consulting Services Inc and are available to attendees of this course.

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

TDK Consulting Services Inc

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

- Define Various OO Language Elements
- Examine code which shows how they behave
- Use OEA/PDS to step through live examples

OO Language Concepts for 4GL Developers
And we’re off!

TDK Consulting Services Inc

And we’re off!

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.

```
/* presentation/classes/ClassWrapperClass.cls       */
CLASS presentation.classes.ClassWrapperClass:
   /* data definitions         */
  /* method definitions     */
  /* constructor, destructor*/
END CLASS.
```

Note: Class names must be distinct from db table field names!

What is an Object?

An object is the actualization of a class, it contains the class's default data and any actions that can be performed by its methods.

class = definition of how to do something.
object = an instance of that definition

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.

How does one make an object?

```
/* presentation/examples/Class.p      */
DEFINE VARIABLE oClass AS presentation.classes.ClassWrapperClass  NO-UNDO.
oClass = NEW presentation.classes.ClassWrapperClass().
```

What is a Method?

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

```
/* presentation/classes/MethodWrapper.cls       */
CLASS presentation.classes.MethodWrapperClass:

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

END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements: Methods

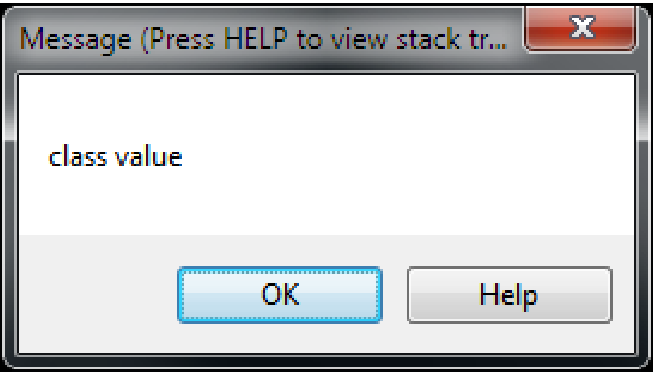
TDK Consulting Services Inc

```
/* presentation/classes/MethodClass.cls */

CLASS presentation.classes.MethodClass:
DEFINE VARIABLE ch-MethodVar AS CHARACTER NO-UNDO.

METHOD VOID SetVariable(ip-methodvar AS CHARACTER):
ch-MethodVar = ip-methodvar.
END METHOD.

METHOD CHARACTER GetVariable():
RETURN(ch-MethodVar).
END METHOD.
END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements: Methods

TDK Consulting Services Inc

```
/* presentation/examples/Method.p  */
DEFINE VARIABLE oMethodClass AS presentation.classes.MethodClass NO-UNDO.

oMethodClass = NEW presentation.classes.MethodClass().

oMethodClass:SetVariable("class value").
MESSAGE oMethodClass:GetVariable() VIEW-AS ALERT-BOX.
```



class value

What is a Property?

A property is a variable encased in a class that can

- be directly accessed by objects and programs,
- contain logic.

```
/* presentation/classes/property/Property.cls      */
CLASS presentation.classes.property.PropertyClass:


DEFINE PUBLIC PROPERTY character-property AS CHARACTER NO-UNDO
   GET.
   SET.


END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements: Properties

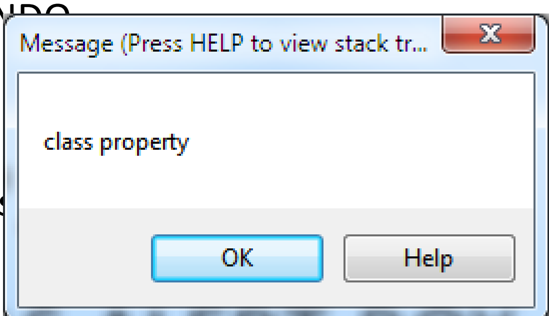
TDK Consulting Services Inc

```
/* presentation/classes/property/PropertySetterClass.cls    */
CLASS presentation.classes.property.PropertySetterClass:

DEFINE PROPERTY ExampleProperty AS CHARACTER NO-UNDO
   GET:
   RETURN(ExampleProperty).
   END GET.

   SET(ip-char AS CHARACTER):
   ASSIGN ExampleProperty = ip-char.
   END SET.

END CLASS.
```

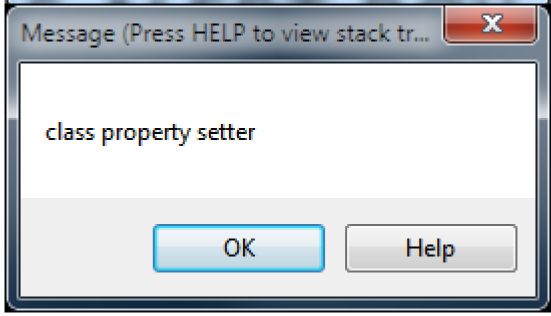
OO Language Concepts for 4GL Developers
OO Language Elements: Properties

TDK Consulting Services Inc

```
/* presentation/classes/Property.p */
DEFINE VARIABLE oProperty
         AS presentation.classes. property.PropertyClass            NO-UNDO.
DEFINE VARIABLE oPropertySetter
         AS presentation.classes. property.PropertySetterClass      NO-UN

oProperty        = NEW presentation.classes.property.PropertyClass().
oPropertySetter = NEW presentation.classes. property.PropertySetterCla

oProperty:ExampleProperty         = "class property".
oPropertySetter:ExampleProperty = "class property setter".

MESSAGE oProperty:ExampleProperty VIEW-AS ALERT-BOX.
MESSAGE oPropertySetter:ExampleProperty VIEW-AS ALERT-BOX.
```



class property



class property setter

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

TDK Consulting Services Inc

```
/* presentation/examples/Using.p     */

DEFINE VARIABLE oMethodClass AS presentation.classes.MethodClass NO-UNDO.
oMethodClass = NEW presentation.classes.MethodClass().
```

Newer, Faster Way:

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

oMethodClass:SetCharacter("class value").
MESSAGE oMethodClass:GetCharacter() VIEW-AS ALERT-BOX.
```

OO Language Concepts for 4GL Developers
OO Language Elements: Exercise Set 1

Quiz #1 - What is all this stuff?

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

Polymorphism provide a way to describe interface functionality that supports calls from different contexts. (In programmer speak, the same method name can be called using different parameter signatures.)

```
InventoryClass:GetParentSKU(1234).
InventoryClass:GetParentSKU("SKU Name").
InventoryClass:GetParentSKU(oSKU).


ProductionOrder:CreateUsing(sales-order-number).
ProductionOrder:CreateUsing(oSalesOrder).
```

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

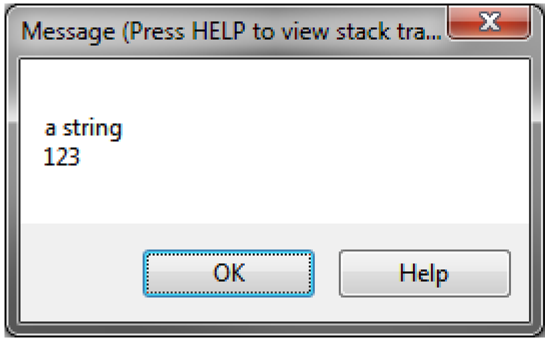
TDK Consulting Services Inc

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

METHOD VOID SetVariable(ip-numeric-value AS INTEGER):
SetVariable(STRING(ip-numeric-value)).
END METHOD.

METHOD VOID SetVariable(ip-charvalue AS CHARACTER):
ch-ExampleVar = ip-charvalue.
END METHOD.

METHOD CHARACTER GetVariable():
RETURN(ch-ExampleVar).
END METHOD.
END CLASS.
```

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

TDK Consulting Services Inc

```
/* presentation/examples/Polymorphism.p */
USING presentation.classes.*.
DEFINE VARIABLE oPolyClass1 AS PolymorphismClass NO-UNDO.
DEFINE VARIABLE oPolyClass2 AS PolymorphismClass NO-UNDO.

oPolyClass1 = NEW PolymorphismClass().
oPolyClass2 = NEW PolymorphismClass().

oPolyClass1:SetVariable("a string").
oPolyClass2:SetVariable(123).

MESSAGE oPolyClass1:GetVariable()   SKIP
                oPolyClass2:GetVariable()
   VIEW-AS ALERT-BOX.
```



```
a string
123
```

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

TDK Consulting Services Inc

Caution: Polymorphism only applies to *parameter data type signatures*, not to *data return types*.

Legal – two methods in the class with
- the same method name,
- the same data return type,
- different parameter signatures:

```
METHOD PUBLIC VOID SetVariable(ch-var as CHARACTER):
METHOD PUBLIC VOID SetVariable(de-var as DECIMAL):
```

Not Legal – two methods in the class with
- the same method name,
- the same parameter signature,
- differing data return types:

```
METHOD PUBLIC CHARACTER GetVariable():
METHOD PUBLIC DECIMAL GetVariable():
```

Attempts to do this will result in a compiler error.

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

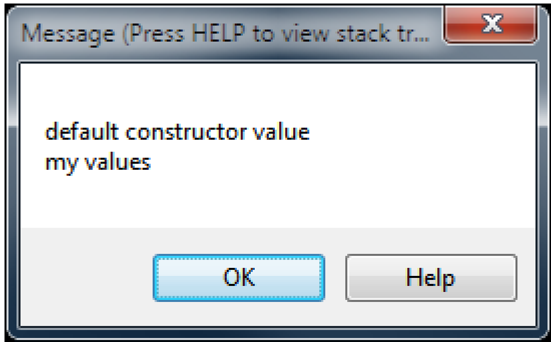
TDK Consulting Services Inc

```
/* presentation/classes/ConstructorClass.cls */
CLASS presentation.classes.ConstructorClass:
DEFINE VARIABLE ch-value AS CHARACTER.

CONSTRUCTOR ConstructorClass():
THIS-OBJECT("default constructor value").          /* Calls a different constructor in this class */
END CONSTRUCTOR.

CONSTRUCTOR ConstructorClass(ipc-value AS CHARACTER):
ASSIGN ch-value = ipc-value.
END CONSTRUCTOR.

METHOD CHARACTER GetVariable():
RETURN(ch-value).
END METHOD.
END CLASS.
```

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

TDK Consulting Services Inc

```
/* presentation/examples/Constructor.p */
USING presentation.classes.*.

DEFINE VARIABLE aConstructor AS ConstructorClass NO-UNDO.
DEFINE VARIABLE bConstructor AS ConstructorClass NO-UNDO.

aConstructor = NEW ConstructorClass().
bConstructor = NEW ConstructorClass("my values").

MESSAGE aConstructor:GetVariable()   SKIP
         bConstructor:GetVariable()
  VIEW-AS ALERT-BOX.
```



```
default constructor value
my values
```

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

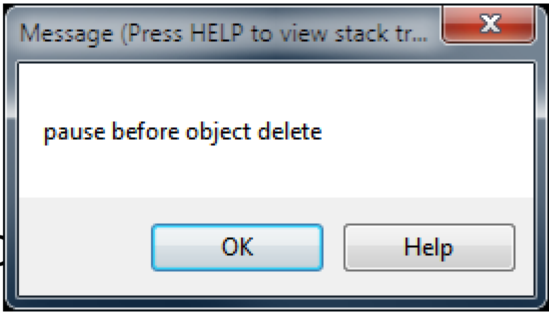
TDK Consulting Services Inc

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

DESTRUCTOR DestructorClass():
MESSAGE "in destructor for object" SKIP ch-name VIEW-AS ALERT-BOX.
END DESTRUCTOR.

CONSTRUCTOR DestructorClass():
ASSIGN ch-name = "created using default constructor".
END CONSTRUCTOR.
CONSTRUCTOR DestructorClass(ipc-parm AS CHARACTER):
ASSIGN ch-name = ipc-parm.
END CONSTRUCTOR.
END CLASS.
```

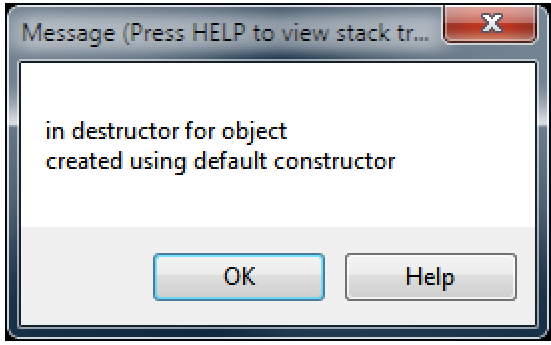
OO Language Concepts for 4GL Developers
OO Language Elements: Destructors

TDK Consulting Services Inc

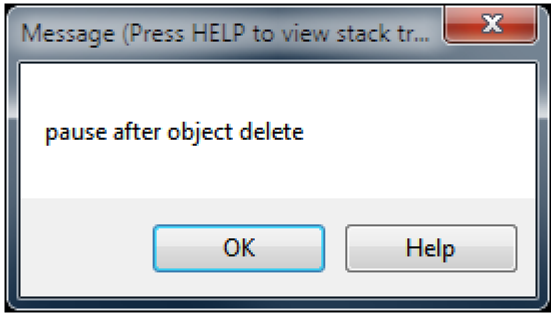
```
/* presentations/examples/Destructor.p    */
USING presentation.classes.*.
DEFINE VARIABLE oDestructorClass AS DestructorClass NO-UNDO

oDestructorClass = NEW DestructorClass().

MESSAGE "pause before object delete" VIEW-AS ALERT-BOX.

DELETE OBJECT oDestructorClass.

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


Message (Press HELP to view stack tr...

pause before object delete


Message (Press HELP to view stack tr...

in destructor for object
created using default constructor


Message (Press HELP to view stack tr...

pause after object delete

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

TDK Consulting Services Inc

Sample Code:

```
USING presentation.classes.*.
DEFINE VARIABLE oDestructorClass AS DestructorClass NO-UNDO.
oDestructorClass = 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”.

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

TDK Consulting Services Inc

```
/* presentations/examples/Garbage-collection-1.p    */
USING presentation.classes.*.
DEFINE VARIABLE oDestructorClass AS DestructorClass NO-UNDO.

oDestructorClass = NEW DestructorClass().

MESSAGE "run child program" VIEW-AS ALERT-BOX.
RUN presentation/examples/Garbage-collection-2.p.
MESSAGE "pause main program after child was run" VIEW-AS ALERT-BOX.

DELETE OBJECT oDestructorClass.

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

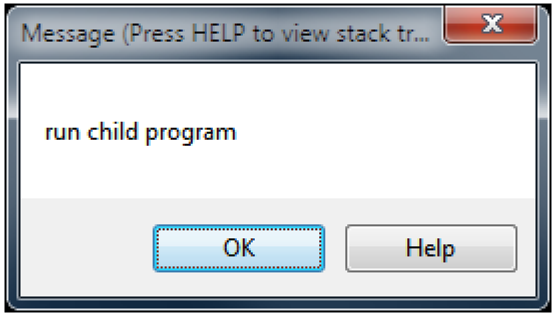
OO Language Concepts for 4GL Developers
OO Language Elements: Garbage Collection

TDK Consulting Services Inc

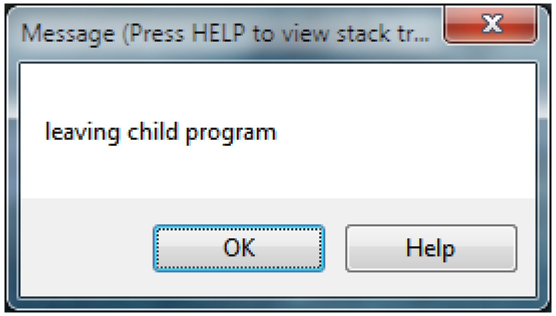
```
/* presentations/examples/Garbage-collection-2.p */

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

oDestructClass = NEW DestructorClass("created by child procedure").

MESSAGE " leaving child program"
   VIEW-AS ALERT-BOX.
```

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

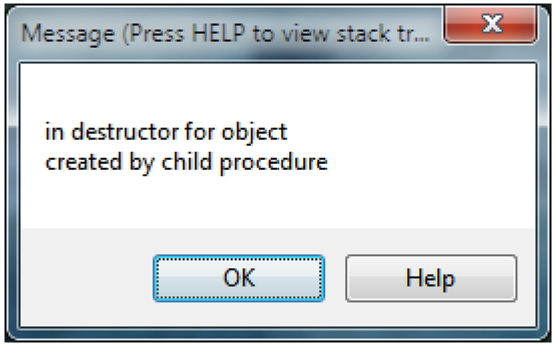
TDK Consulting Services Inc


Message (Press HELP to view stack tr...

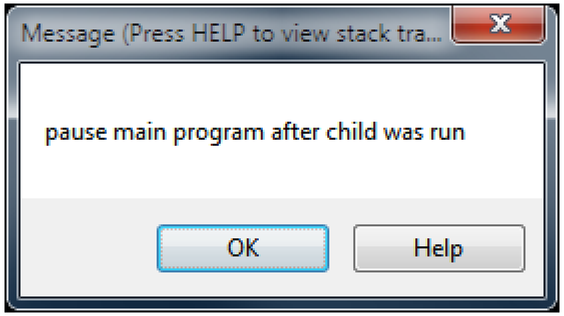
run child program


Message (Press HELP to view stack tr...

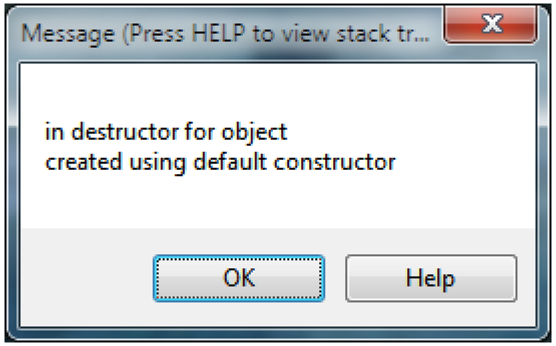
leaving child program


Message (Press HELP to view stack tr...

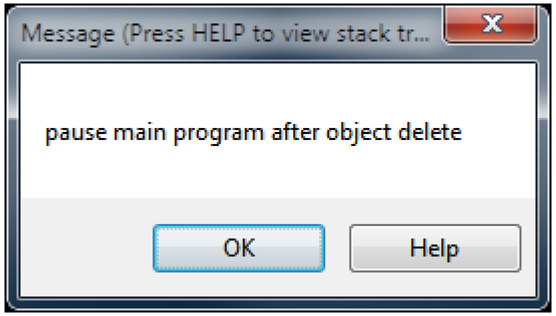
in destructor for object
created by child procedure


Message (Press HELP to view stack tra...

pause main program after child was run


Message (Press HELP to view stack tr...

in destructor for object
created using default constructor


Message (Press HELP to view stack tr...

pause main program after object delete

OO Language Concepts for 4GL Developers
OO Language Elements: Quiz #2

TDK Consulting Services Inc

Quiz #2 - It's a ShapeShifting Class's Life

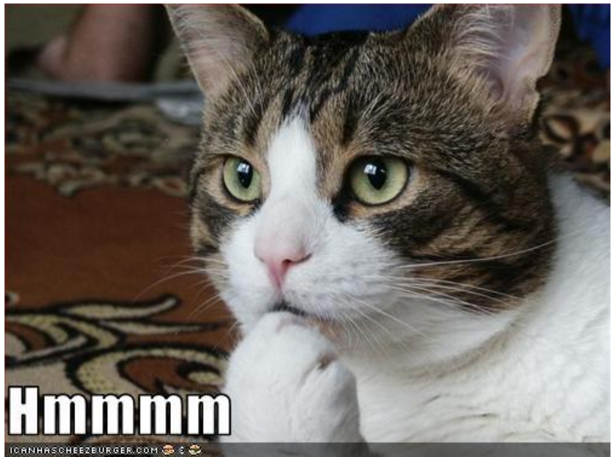
OO Language Concepts for 4GL Developers
OO Language Elements: Inheritance

What is Inheritance?

Inheritance is a way to compartmentalize and extend code. This is done by creating collections of attributes and behaviors in a class, and then creating new classes which take the functionality of that class and extend the functionality to do new things.

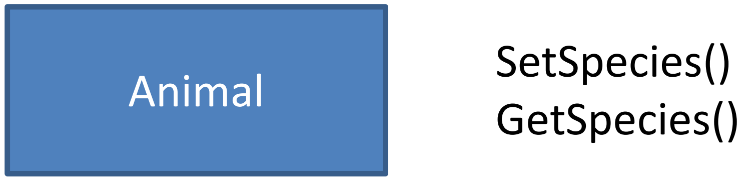
Inheritance also allows a subclass to be used anywhere a superclass can be used.

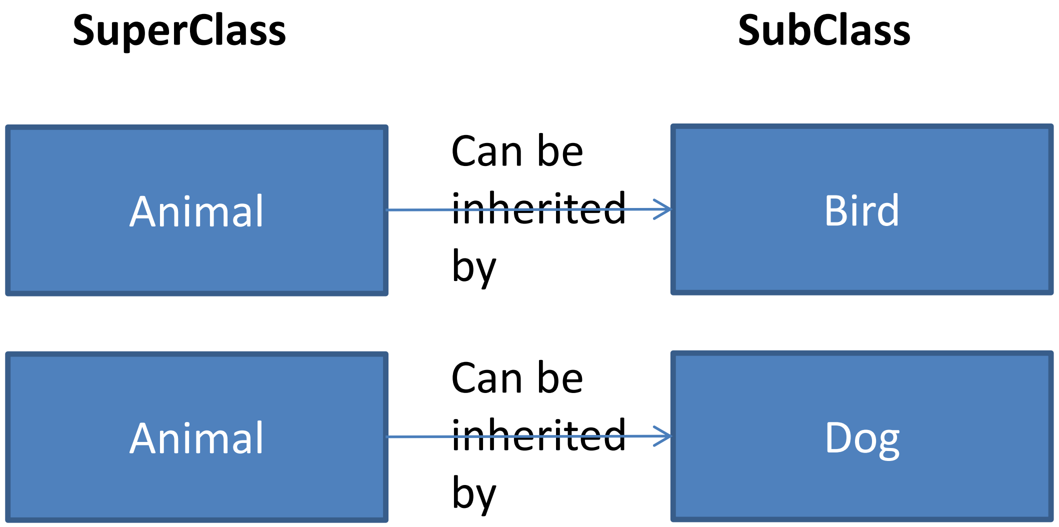
This is also where things start to get interesting



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

TDK Consulting Services Inc





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

TDK Consulting Services Inc

```
/* presentation/classes/inheritance/Animal.cls  */
CLASS presentation.classes.inheritance.Animal:
DEFINE VARIABLE ch-species-name AS CHARACTER NO-UNDO.

METHOD VOID SetSpecies(ip-species-name AS CHARACTER):
ASSIGN ch-species-name = ip-species-name.
END METHOD.

METHOD CHARACTER GetSpecies():
RETURN(ch-species-name).
END METHOD.

METHOD CHARACTER GetInformation():
RETURN(ch-species-name).
END METHOD.

CONSTRUCTOR Animal():
END CONSTRUCTOR.

CONSTRUCTOR Animal(ip-species AS CHARACTER):
SetSpecies(ip-species).
END CONSTRUCTOR.

END CLASS.
```

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

TDK Consulting Services Inc

```
/* presentation/classes/inheritance/Bird.cls     */
CLASS presentation.classes.inheritance.Bird
              INHERITS presentation.classes.inheritance.Animal:
DEFINE VARIABLE i-wing-span AS INTEGER NO-UNDO.

METHOD VOID SetWingSpan(ip-wing-span AS INTEGER):
ASSIGN i-wing-span = ip-wing-span.
END METHOD.

METHOD INTEGER GetWingSpan():
RETURN(i-wing-span).
END METHOD.

CONSTRUCTOR Bird(ip-species AS CHARACTER, ip-wing-span AS INTEGER):
SetSpecies(ip-species).
SetWingSpan(ip-wing-span).
END CONSTRUCTOR.
END CLASS.
```

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

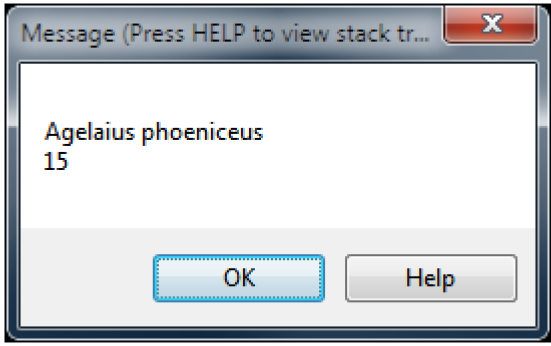
TDK Consulting Services Inc

```
/* presentation/classes/inheritance/Dog.cls     */
CLASS presentation.classes.inheritance.Dog
   INHERITS presentation.classes.inheritance.Animal:
DEFINE VARIABLE ch-breed-name       AS CHARACTER NO-UNDO.

METHOD VOID SetBreed(ip-breed-name AS CHARACTER):
ASSIGN ch-breed-name = ip-breed-name.
END.

METHOD CHARACTER GetBreedName():
RETURN(ch-breed-name).
END METHOD.

CONSTRUCTOR Dog(ip-species AS CHARACTER,  ip-breed   AS CHARACTER):
SetSpecies(ip-species).
SetBreed(ip-breed).
END CONSTRUCTOR.

END CLASS.
```

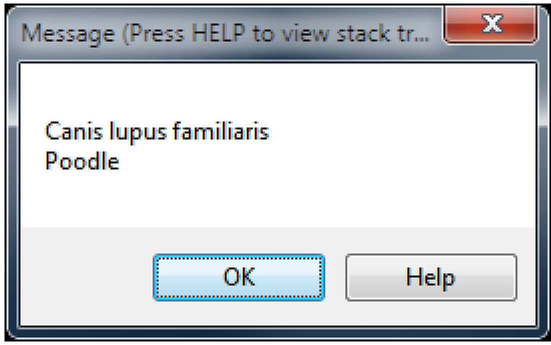
OO Language Concepts for 4GL Developers
OO Language Elements: Inheritance

TDK Consulting Services Inc

```
/* presentation/examples/Inheritance.p              */

USING presentation.classes.inheritance.*.
DEFINE VARIABLE oBird   AS Bird   NO-UNDO.
DEFINE VARIABLE oDog    AS Dog    NO-UNDO.

oBird = NEW Bird("Agapornis canus",         5).
oDog = NEW Dog("Canis lupus familiaris", "Poodle").

MESSAGE oBird:GetSpecies()  SKIP
            oBird:GetWingSpan()
    VIEW-AS ALERT-BOX.

MESSAGE oDog:GetSpecies()  SKIP
            oDog:GetBreedName()
    VIEW-AS ALERT-BOX.
```



```
Agelaius phoeniceus
15
```



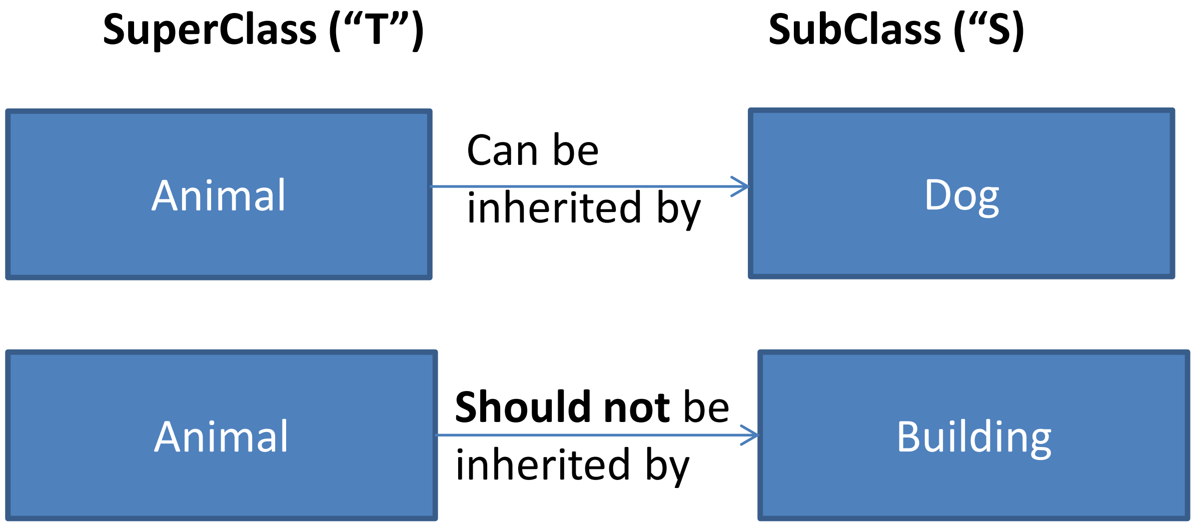
```
Canis lupus familiaris
Poodle
```

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

TDK Consulting Services Inc

Liskov Substitution Principle
If S is a subtype of T, then objects of type T in a program may be replaced with objects of type S without altering any of the desirable properties of that program

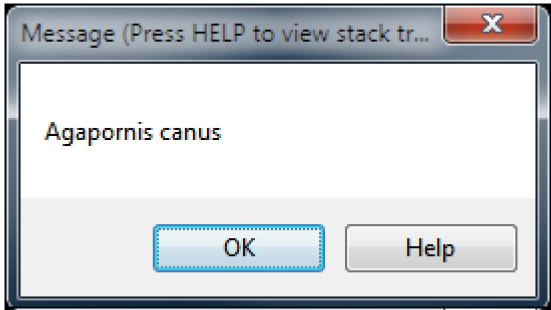
In other words, if the subclass can always be substituted for it's base/superclass, then
- the design is good,
- the subclass can be used anywhere the base/superclass can.



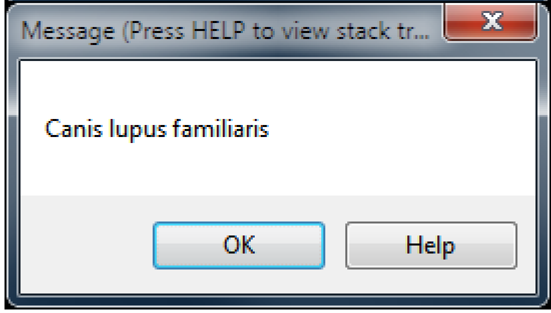
OO Language Concepts for 4GL Developers
OO Language Elements: Inheritance/Substitution

TDK Consulting Services Inc

```
/* presentation/examples/InheritanceAndSubstitution.p     */
USING presentation.classes.inheritance.*.
DEFINE VARIABLE oBird   AS Bird   NO-UNDO.
DEFINE VARIABLE oDog    AS Dog    NO-UNDO.

oBird = NEW Bird("Agapornis canus",        5).
oDog = NEW Dog("Canis lupus familiaris", "Poodle").

/* Note passing subclass as a superclass parameter */
RUN show-species(oBird).
RUN show-species(oDog).

PROCEDURE show-species:
DEFINE INPUT PARAMETER oAnimal AS Animal NO-UNDO.
MESSAGE oAnimal:GetSpecies()   SKIP
            VIEW-AS ALERT-BOX.
END PROCEDURE.
```


Message (Press HELP to view stack tr...

Agapornis canus


Message (Press HELP to view stack tr...

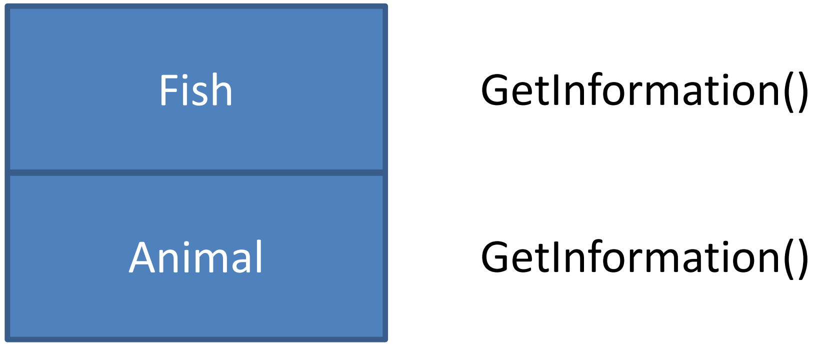
Canis lupus familiaris

OO Language Concepts for 4GL Developers
OO Language Elements: Inheritance/Override

Polymorphism allows for using the same API name with different parameters within a class.

Question:

What about when
- a subclass needs to extend a method with the same parameter signature as a superclass?
- It still needs to access the superclass method's functionality?



Answer: SUPER: Call up the inheritance chain for a matching method definition

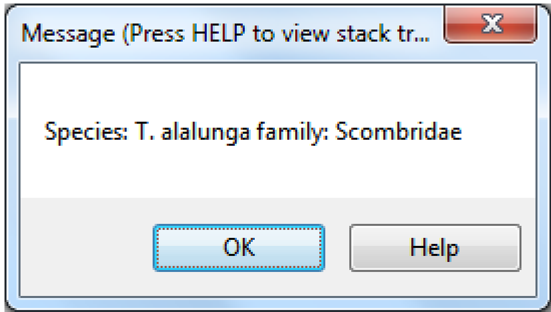
OO Language Concepts for 4GL Developers
OO Language Elements: Inheritance/Override

```
/* presentation/classes/override/Fish.cls      */

METHOD OVERRIDE CHARACTER GetInformation(): /* Overrides "Animal" Method    */
RETURN(SUPER:GetInformation() + " family: " + ch-family).
END METHOD.

CONSTRUCTOR Fish(ip-family AS CHARACTER):
THIS-OBJECT("", ip-family).
END CONSTRUCTOR.

CONSTRUCTOR Fish(ip-species AS CHARACTER, ip-family  AS CHARACTER):
SUPER(ip-species).  /* Call the super-class constructor */
SetFamily(ip-family).
END CONSTRUCTOR.

METHOD VOID SetFamily(newvalue AS CHARACTER):
ch-family = newvalue.
END METHOD.
END CLASS.
```

OO Language Concepts for 4GL Developers
OO Language Elements: Inheritance/Override

TDK Consulting Services Inc

```
/* presentation/examples/InheritanceAndOverride.p   */
USING presentation.classes.override.*.
DEFINE VARIABLE oFish AS Fish NO-UNDO.

                              /* Albacore Tuna    */
oFish = NEW Fish("T. alalunga",     /* Species */
                             "Scombridae"   /* Family   */
                             ).

MESSAGE oFish:GetInformation()
   VIEW-AS ALERT-BOX.
```



Species: T. alalunga family: Scombridae

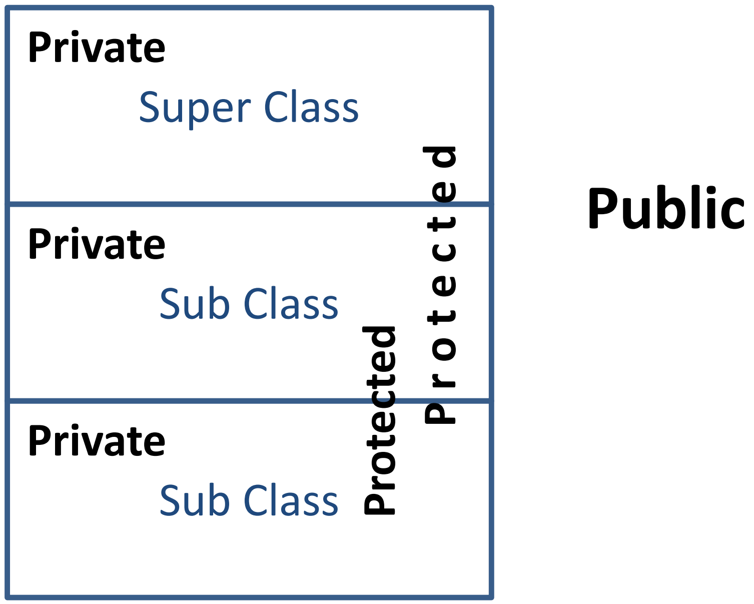
What does Private, Protected, Public mean?

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.

Modifier	OO Access Limitations	Procedural version
PRIVATE	Current class instance	PRIVATE
PROTECTED	Current class instance and all subclasses	THIS-PROCEDURE super procedure
PUBLIC	Any code which has access to the object	Session super procedure, Procedure handle access

Note: OE 11.3 will change these access rules from “instance” scope to “class” scope

What does Private, Protected, and Public look like?



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

TDK Consulting Services Inc

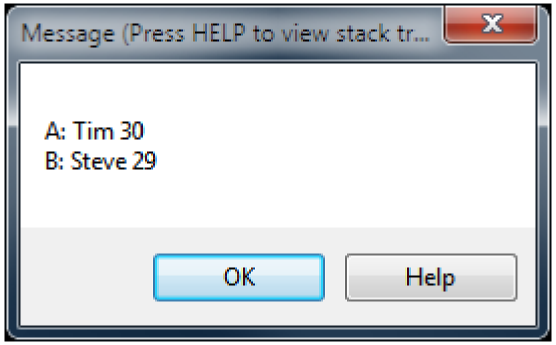
presentation/classes/access:
Access Classes, their methods, and access modifiers

Class	Method	Access
AccessSuperClass	SetMyName	PUBLIC
	GetMyName	PUBLIC
	SetMyAge	PROTECTED
	GetMyAge	PUBLIC
	AccessSuperClass (CR)	PUBLIC
	AccessSuperClass (CR)	PRIVATE
AccessSubClass	SetMyBirthDay	*PUBLIC (default)*

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

TDK Consulting Services Inc

```
/* presentation/examples/Access.p     */
USING presentation.classes.access.*.
DEFINE VARIABLE oStudentA AS AccessSubClass NO-UNDO.
DEFINE VARIABLE oStudentB AS AccessSubClass NO-UNDO.
oStudentA = NEW AccessSubClass().
oStudentB = NEW AccessSubClass().
oStudentA:SetMyBirthDay(ADD-INTERVAL(TODAY, -30, "years")).
oStudentB:SetMyName("Steve").
oStudentB:SetMyBirthDay(ADD-INTERVAL(TODAY, -29, "years")).
MESSAGE "A:" oStudentA:GetMyName() oStudentA:GetMyAge()   SKIP
            "B:" oStudentB:GetMyName() oStudentB:GetMyAge()
  VIEW-AS ALERT-BOX.
```



```
A: Tim 30
B: Steve 29
```

OO Language Concepts for 4GL Developers
OO Language Elements: Quiz #3

TDK Consulting Services Inc

Quiz #3 – Inheritance and Access Control

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

TDK Consulting Services Inc

Static Class Members:

- can be a method, property, variable, buffer, or any other class member type,
- are scoped to the class, not an instance,
- have their own constructor,
- are instantiated whenever any static member in the class is referenced, or a dynamic object instance is created,
- are able to reference other static members or object instances that it starts or are passed to it,
- cannot call a “super” method,
- are referenced using ClassName:MethodName() format

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

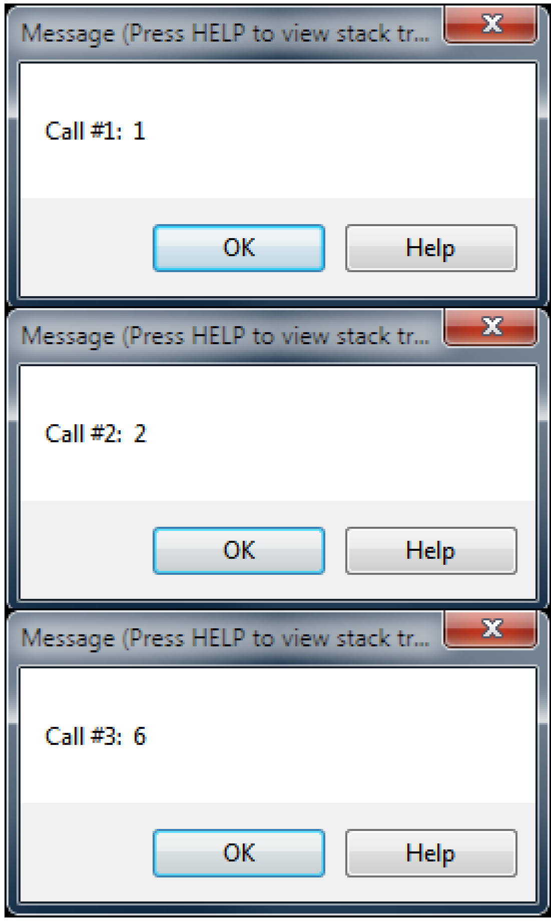
TDK Consulting Services Inc

```
/* presentation/classes/StaticMemberClass.cls   */
CLASS presentation.classes.static.StaticMemberClass:
DEFINE PUBLIC STATIC PROPERTY il-key-value AS INT64 NO-UNDO GET. SET.

METHOD PUBLIC STATIC INT64 GetNextKey():
il-key-value = il-key-value + 1.
RETURN(il-key-value).
END METHOD.

METHOD PUBLIC STATIC VOID SetKey(ip-keyvalue AS INT64):
il-key-value = ip-keyvalue.
END METHOD.

CONSTRUCTOR STATIC StaticMemberClass():
il-key-value = 0.
END CONSTRUCTOR.
END CLASS.
```

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

TDK Consulting Services Inc

```
/* presentation/examples/Static-only.p     */
USING presentation.classes.static.*.

MESSAGE StaticMemberClass:GetNextKey()
    VIEW-AS ALERT-BOX.

MESSAGE StaticMemberClass:GetNextKey()
    VIEW-AS ALERT-BOX.

StaticMemberClass:SetKey(5).

MESSAGE StaticMemberClass:GetNextKey()
    VIEW-AS ALERT-BOX.
```


Message (Press HELP to view stack tr...

Call #1: 1

Message (Press HELP to view stack tr...

Call #2: 2

Message (Press HELP to view stack tr...

Call #3: 6

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

TDK Consulting Services Inc

```
CLASS presentation.classes.static.StaticAndDynamicClass:

DEFINE PUBLIC STATIC PROPERTY il-stat-key-value AS INT64 NO-UNDO
        GET():
        MESSAGE "In Get Static Property" VIEW-AS ALERT-BOX.
        RETURN(il-stat-key-value).
        END GET.

        SET(il-parm AS INT64):
        MESSAGE "In Set Static Property" VIEW-AS ALERT-BOX.
        ASSIGN il-stat-key-value = il-parm.
        END SET.
```

```
DEFINE PUBLIC PROPERTY il-dyn-key-value AS INT64 NO-UNDO
        GET():
        MESSAGE "In Get Dynamic Property" VIEW-AS ALERT-BOX.
        RETURN(il-dyn-key-value).
        END GET.


       SET(il-parm AS INT64):
        MESSAGE "In Set Dynamic Property" VIEW-AS ALERT-BOX.
        ASSIGN il-dyn-key-value = il-parm.
        END SET.
```

```
CONSTRUCTOR StaticAndDynamicClass():
MESSAGE "In Dynamic Constructor" VIEW-AS ALERT-BOX.
END CONSTRUCTOR.

CONSTRUCTOR STATIC StaticAndDynamicClass():
MESSAGE "In Static Constructor" VIEW-AS ALERT-BOX.
END CONSTRUCTOR.
```

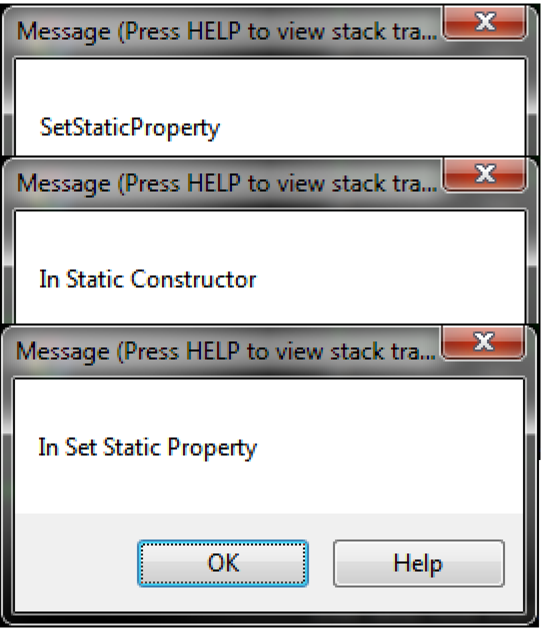
OO Language Concepts for 4GL Developers
OO Language Elements: Static Members

TDK Consulting Services Inc

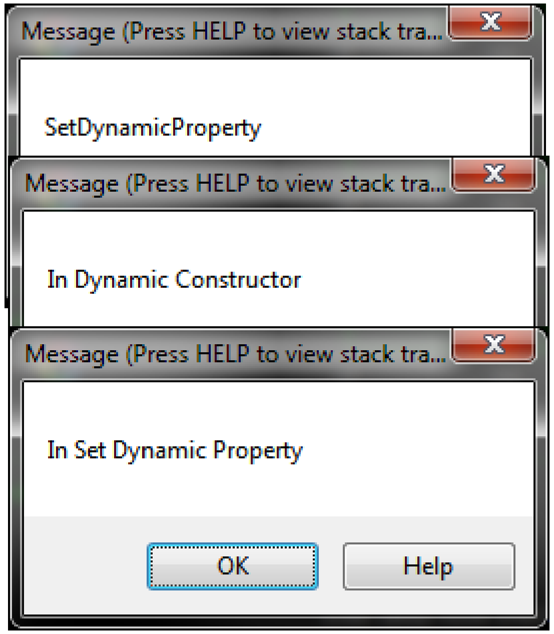
```
/* presentation/examples/StaticThenDynamic.p       */
USING presentation.classes.static.*.

DEFINE VARIABLE oStaticAndDynamicClass  StaticAndDynamicClass NO-UNDO.

MESSAGE "SetStaticProperty" VIEW-AS ALERT-BOX.
StaticAndDynamicClass:il-stat-key-value = 1.

MESSAGE "SetDynamicProperty" VIEW-AS ALERT-BOX.
oStaticAndDynamicClass = NEW StaticAndDynamicClass().
oStaticAndDynamicClass:il-dyn-key-value = 1.

 MESSAGE StaticAndDynamicClass:il-stat-key-value   SKIP
             oStaticAndDynamicClass:il-dyn-key-value
      VIEW-AS ALERT-BOX.
```

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

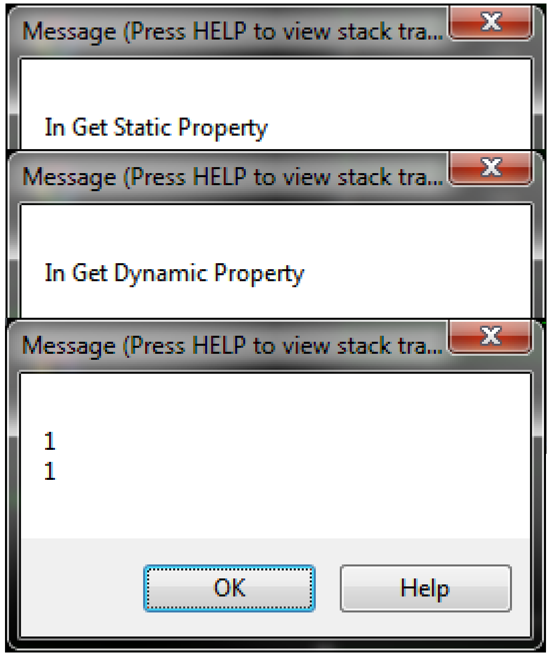
TDK Consulting Services Inc



```
Message (Press HELP to view stack tra...
SetStaticProperty

Message (Press HELP to view stack tra...
In Static Constructor

Message (Press HELP to view stack tra...
In Set Static Property
```



```
Message (Press HELP to view stack tra...
SetDynamicProperty

Message (Press HELP to view stack tra...
In Dynamic Constructor

Message (Press HELP to view stack tra...
In Set Dynamic Property
```



```
Message (Press HELP to view stack tra...
In Get Static Property

Message (Press HELP to view stack tra...
In Get Dynamic Property

Message (Press HELP to view stack tra...
1
1
```

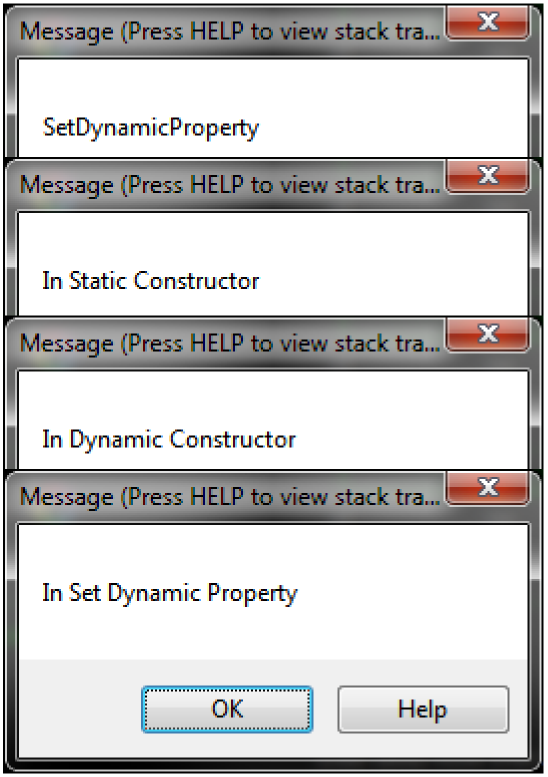
OO Language Concepts for 4GL Developers
OO Language Elements: Static Members

TDK Consulting Services Inc

```
/* presentation/examples/DynamicThenStatic.p          */
USING presentation.classes.static.*.

DEFINE VARIABLE oStaticAndDynamicClass AS StaticAndDynamicClass NO-UNDO.

MESSAGE "SetDynamicProperty" VIEW-AS ALERT-BOX.
oStaticAndDynamicClass = NEW StaticAndDynamicClass().
oStaticAndDynamicClass:il-dyn-key-value = 1.

MESSAGE "SetStaticProperty" VIEW-AS ALERT-BOX.
StaticAndDynamicClass:il-stat-key-value = 1.

 MESSAGE StaticAndDynamicClass:il-stat-key-value   SKIP
         oStaticAndDynamicClass:il-dyn-key-value
        VIEW-AS ALERT-BOX.
```

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

TDK Consulting Services Inc


Message (Press HELP to view stack tra...

SetDynamicProperty

Message (Press HELP to view stack tra...

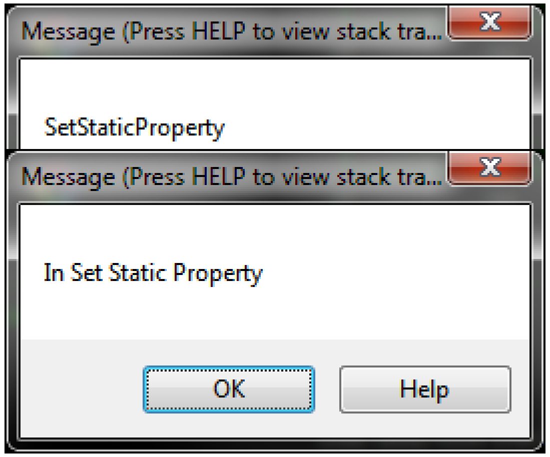
In Static Constructor

Message (Press HELP to view stack tra...

In Dynamic Constructor

Message (Press HELP to view stack tra...

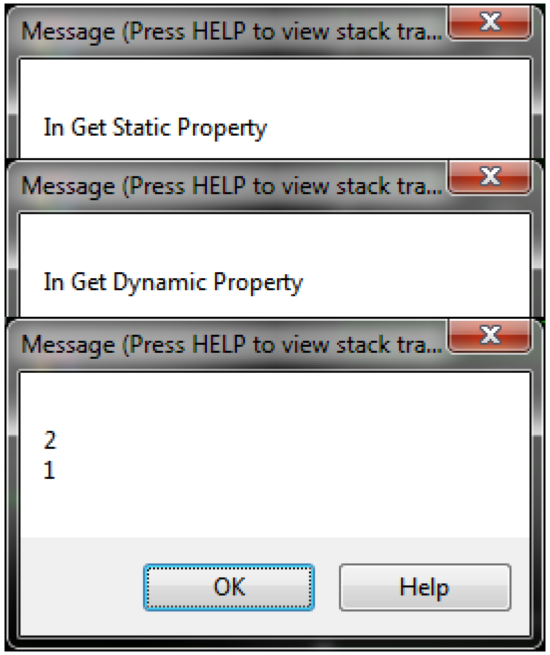
In Set Dynamic Property


Message (Press HELP to view stack tra...

SetStaticProperty

Message (Press HELP to view stack tra...

In Set Static Property


Message (Press HELP to view stack tra...

In Get Static Property

Message (Press HELP to view stack tra...

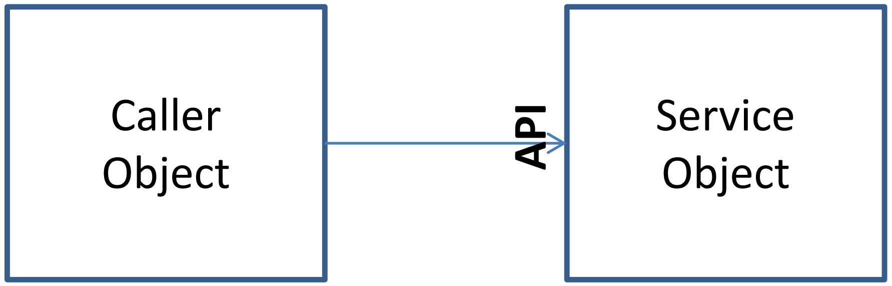
In Get Dynamic Property

Message (Press HELP to view stack tra...

```
2
1
```

OO Language Concepts for 4GL Developers
OO Language Elements: Interfaces

TDK Consulting Services Inc



Interfaces provide a way to specify what an object's service API will look like without actually implementing it.

This API specification (aka "interface") can then be used anywhere an object reference can

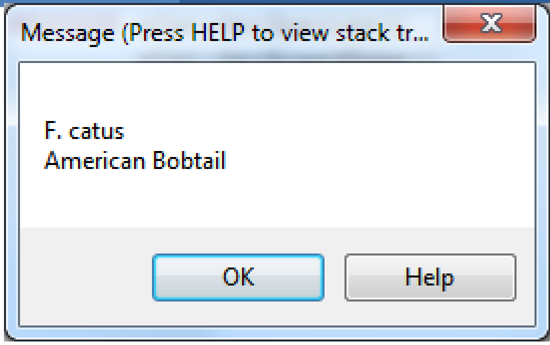
OO Language Concepts for 4GL Developers
OO Language Elements: Interfaces

TDK Consulting Services Inc

```
/* presentation/classes/interface/iAnimal.cls      */
INTERFACE presentation.classes.interface.iAnimal:
METHOD VOID           SetSpecies(ip-species-name AS CHARACTER):
METHOD CHARACTER GetSpecies():
END INTERFACE.


/* presentation/classes/interface/iCat.cls         */
INTERFACE presentation.classes.interface.iCat
   INHERITS presentation.classes.interface.iAnimal:

METHOD VOID           SetBreed(ip-breed-name AS CHARACTER):
METHOD CHARACTER GetBreedName():

END INTERFACE.
```

OO Language Concepts for 4GL Developers
OO Language Elements: Interfaces

TDK Consulting Services Inc


Message (Press HELP to view stack tr...

```
F. catus
American Bobtail
```

```
/* presentation/classes/examples/Interface.p    */

USING presentation.classes.interface.*.
DEFINE VARIABLE oCat AS iCat NO-UNDO.

oCat = NEW Cat( "F. catus",                    /* Domestic Cat */
                "American Bobtail").

MESSAGE oCat:GetSpecies()  SKIP
            oCat:GetBreedName()
   VIEW-AS ALERT-BOX.
```

OO Language Concepts for 4GL Developers
OO Language Elements: Quiz #4

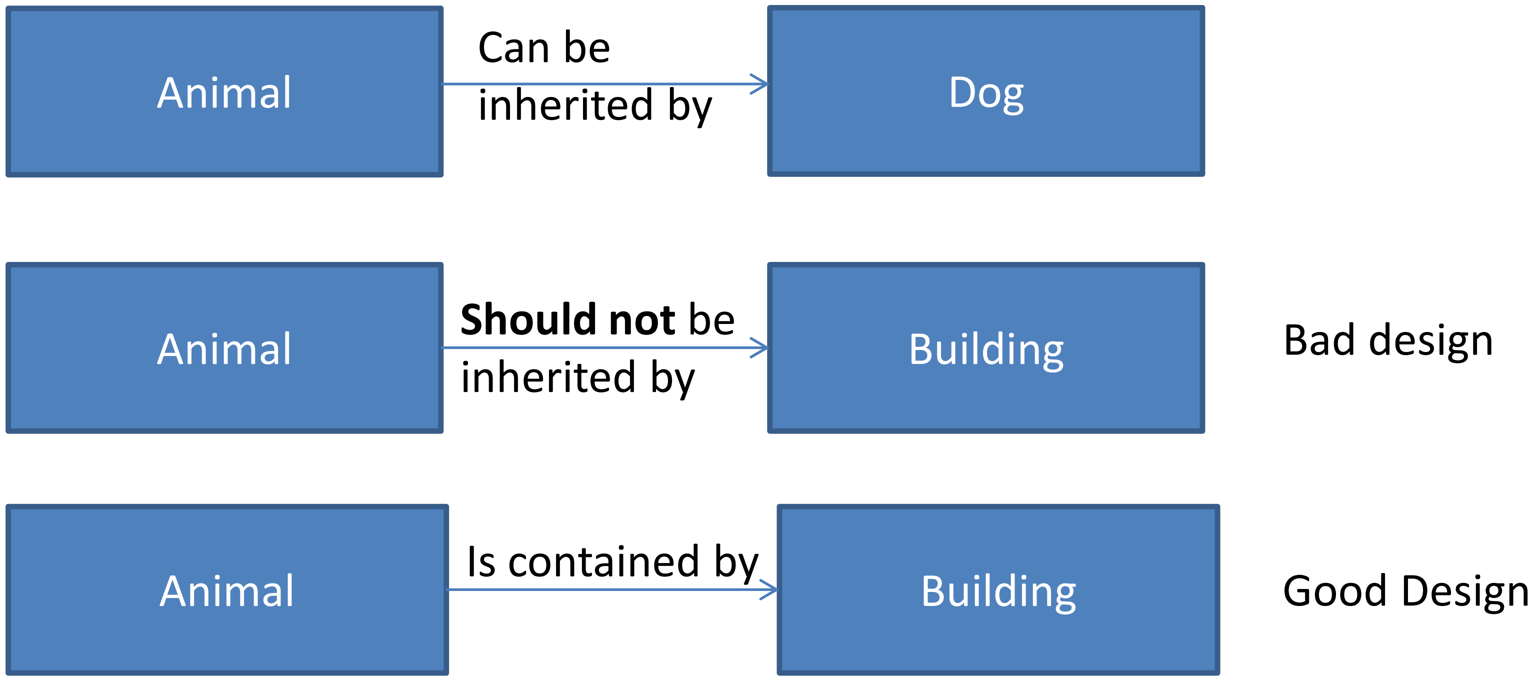
TDK Consulting Services Inc

Quiz #4: Static Members and Interfaces

OO Language Concepts for 4GL Developers
OO Language Elements: Is A & Has A

TDK Consulting Services Inc

- Is-A: The current object is always identical to the inherited object
- Has-A: The current object contains another object



OO Language Concepts for 4GL Developers
OO Language Elements: Is A & Has A

TDK Consulting Services Inc

- Is-A: The current object is always identical to the inherited object

```
CLASS presentation.classes.inheritance.Dog
   INHERITS presentation.classes.inheritance.Animal
```

- Has-A: The current object contains another object

```
CLASS presentation.classes.Building:
DEFINE VARIABLE oDog AS presentation.classes.inheritance.Dog NO-UNDO.
```

OO Language Concepts for 4GL Developers
Some Parting Comments: Now, About those Tools

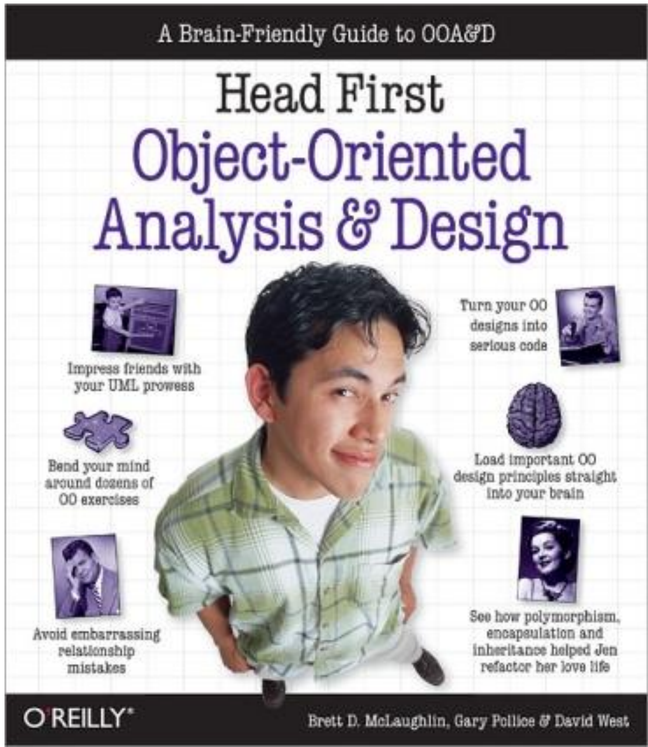
This presentation has shown some OO tools, how they work, and a few ideas of how to use them.

And like these tools, it'll take a lot of work to learn how to master them.

OO Language Concepts for 4GL Developers
Some Parting Comments: Where From Here

TDK Consulting Services Inc

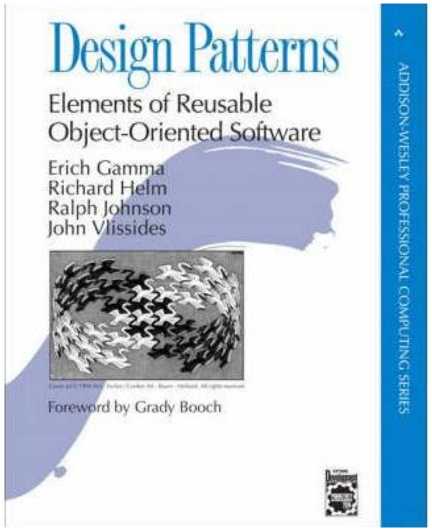
- Find a local mentor, or bring someone on-site for a 1-2 years
- Youtube for Google Code talks on OO
- Good Books on OO structure
- Books written for languages like java are also good
- Find online forums where OO people hang out – like stackexchange.com
- Lots of material on the Internet and Amazon.com

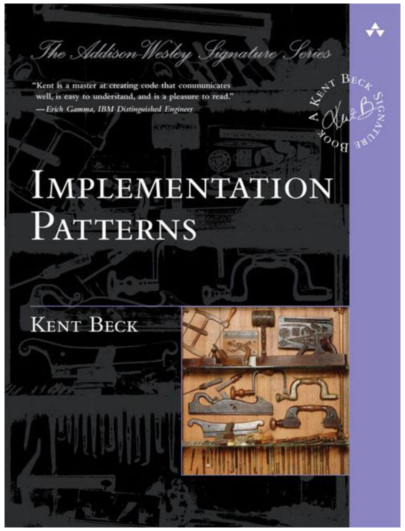

A Brain-Friendly Guide to OOA&D
Head First
Object-Oriented Analysis & Design
Impress friends with your UML prowess
Turn your OO designs into serious code
Bend your mind around dozens of OO exercises
Load important OO design principles straight into your brain
Avoid embarrassing relationship mistakes
See how polymorphism, encapsulation and inheritance helped Jen refactor her love life
O'REILLY
Brett D. McLaughlin, Gary Pollice & David West

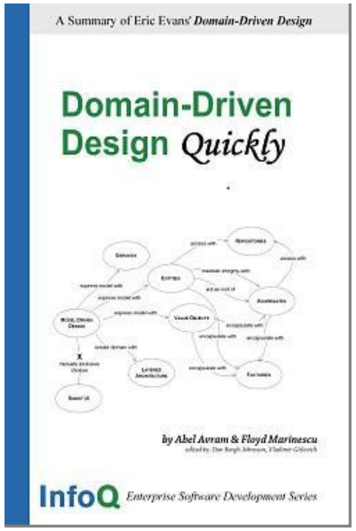
OO Language Concepts for 4GL Developers
Some Parting Comments: Where From Here

TDK Consulting Services Inc

Lots of "Pattern" books can save you from hours of inventing and debugging wheels others perfected a long time ago


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


The Addison Wesley Signature Series
"Kent is a master at creating code that communicates well, is easy to understand, and is a pleasure to read."
—Erich Gamma, IBM Distinguished Engineer
A Kent Beck Signature Book
IMPLEMENTATION PATTERNS
KENT BECK

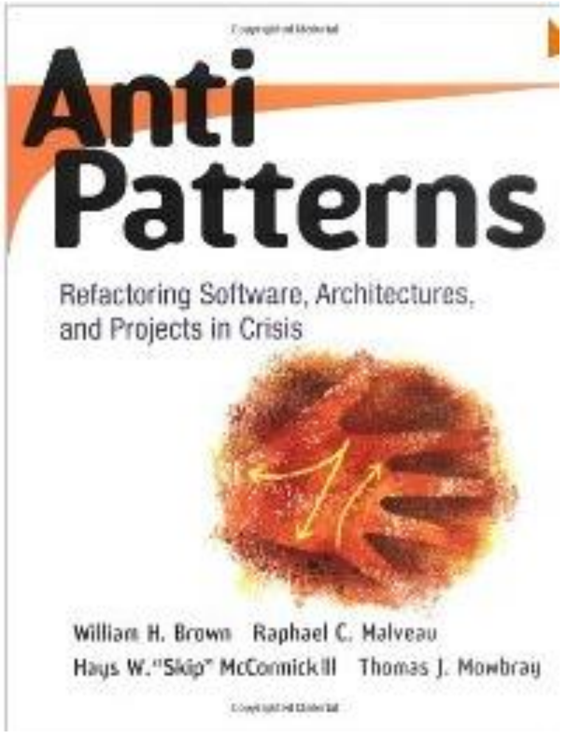

A Summary of Eric Evans' *Domain-Driven Design*
Domain-Driven Design *Quickly*
by Abel Avram & Floyd Marinescu
InfoQ *Enterprise Software Development Series*

http://www.infoq.com/minibooks/domain-driven-design-quickly

OO Language Concepts for 4GL Developers
Some Parting Comments: Danger Will Robinson!

TDK Consulting Services Inc

It’s just as important to know what *not* to do,
as it is to know what *to* do.



Copyright ed Material

Anti Patterns

Refactoring Software, Architectures, and Projects in Crisis

William H. Brown Raphael C. Malveau
Hays W. "Skip" McCormick III Thomas J. Mowbray

When starting out, you will make a big mess in OO, so keep your early efforts confined to places where it can be fixed easily, or doesn’t matter.

Thanks!

Thank you for your time and attention!

Tim Kuehn
TDK Consulting Services Inc.
tim.kuehn@gmail.com