Object Oriented Programming Review

PxPlus 2017 (v14) & PxPlus 2018 (v15)

Agenda

OOP in Review

- What is **O**bject **O**riented **P**rogramming
- PxPlus OOP extensions
- Using OOPs in fun ways
  - Global Data storage
  - Automated Wrap up routines
  - Embedded File IO
  - XML Parsing

Object Oriented Programming

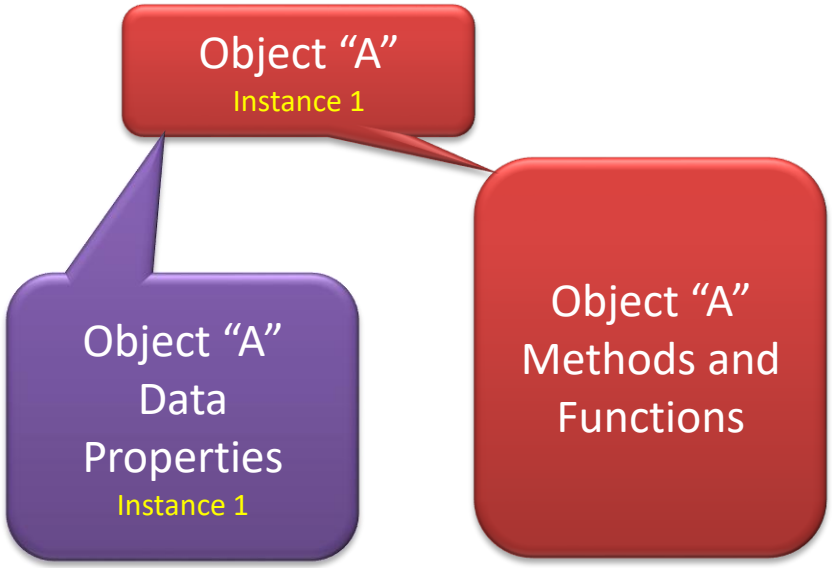
- OOP programming is
  - More structured and modular than standard 3GL
  - Allows more code sharing between modules
  - Simplifies the code by hiding much of the internal workings
  - Provides better integration between code and data structures
  - Available in MOST programming languages
  - When done well can simplify code, sadly, quite often it results in just the opposite

Object Oriented Programming

- What is an object?
  - Think of an object as a code library with data
    - The system will load a single copy of the code
      - Code consists of functions often called **"Methods"**
    - Each **"Instance"** of an object will have its own data
      - Instance data in an object is called a **"Property"**
      - Properties can be exposed or kept private to instance
    - When executing a method…
      - Property values are preserved between calls
      - All other data cleared as per standard CALL logic
    - Objects can build upon one another
      - This is called **"Inheritance""**
      - One object 'inherits' attributes of another

Object Oriented Programming

- What is an object?
  - The Visual…

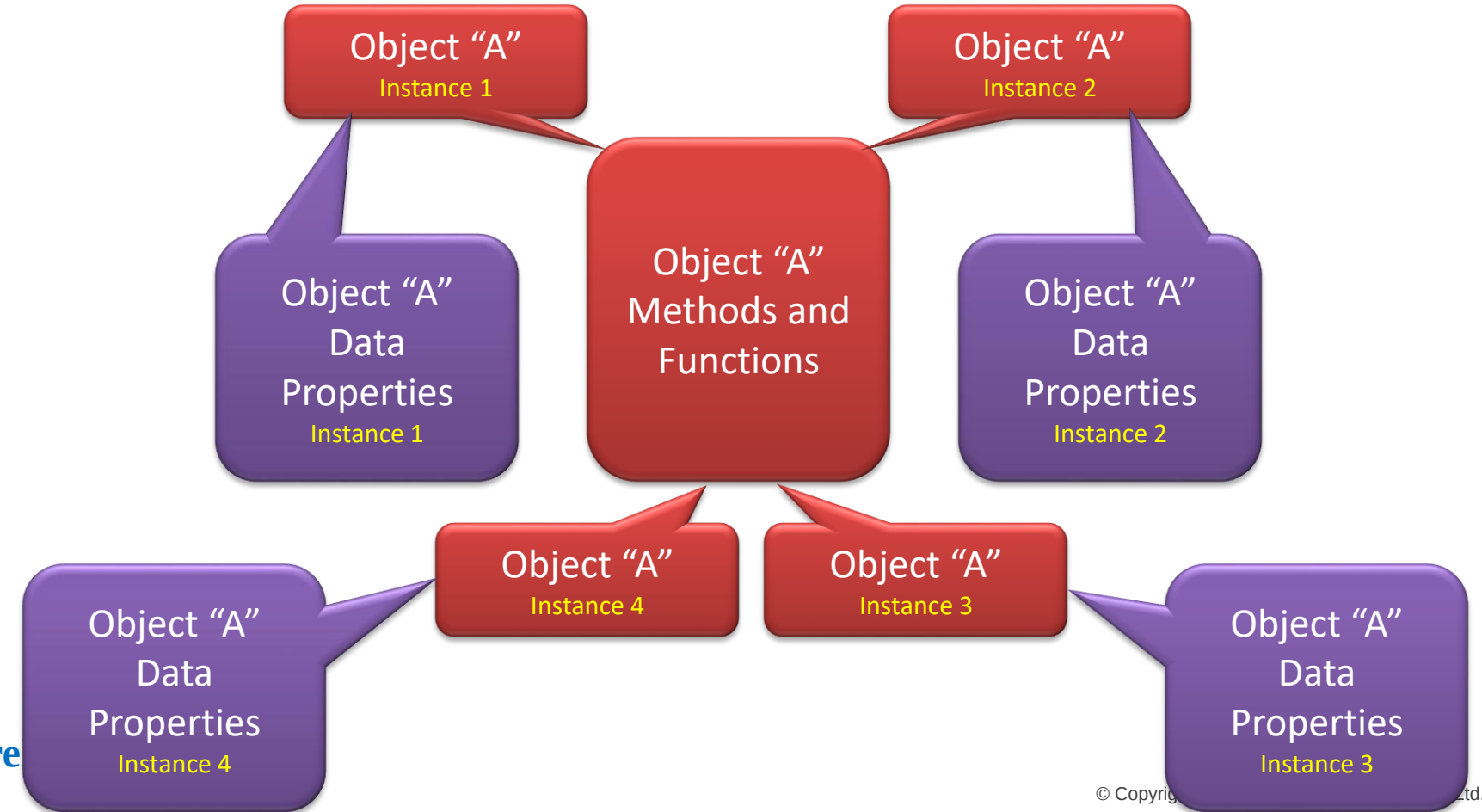


  - The Code…

```
Def Class "a"
Function ReadByKey(k$) DO_ReadByKey
!
Property ClientID$
Property Name$, Address1$, Address2$
!
Local fileNo
End Def
!
On_Create:
  open object (hfn,iol=*)"Client"
  fileNo=lfo
  return
!
DO_ReadByKey:
  enter _key$
  read (fileNo,key=_key$)
  return 1
```

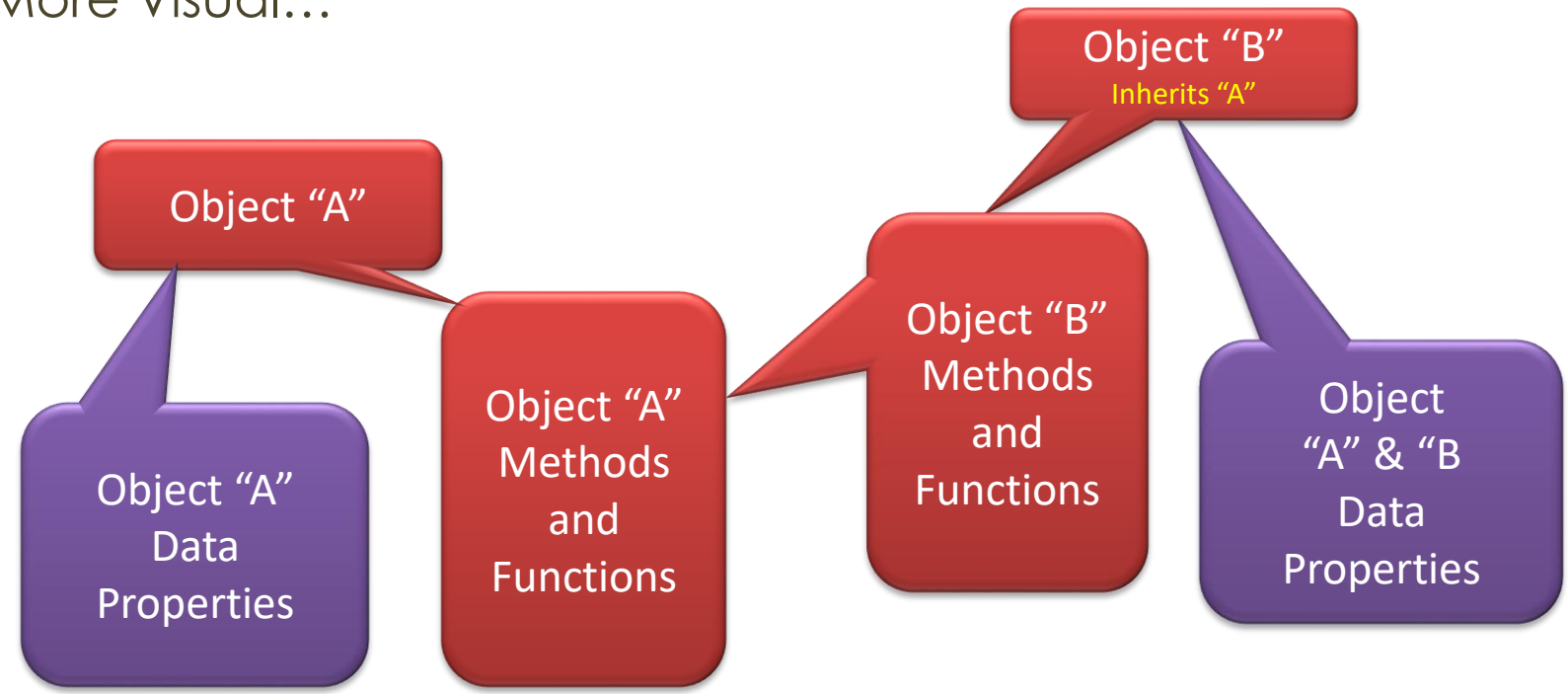
Object Oriented Programming

- What is an object?
  - The Visual…



Object Oriented Programming

- What is an object?
  - More Visual…



Object Oriented Programming

Clear as Mud right??

- Its not that hard
  once you get used to it
- And once you get used to it
  things get easier

PxPlus Extensions

- Automatic cleanup
- Dynamic Properties
- Accept Properties
- Merge Objects
- Embedded IO
- Attach Objects to Controls
- Object Defined Controls
- Object Caching

PxPlus Extensions

- Automatic cleanup
- Dynamic Properties
- Accept Properties
- Merge Objects
- Embedded IO
- Attach Objects to Controls
- Object Defined Controls
- Object Caching

- So let's have a little FUN with objects

Automatic Housekeeping

- To create an object

```
objHandle = NEW( "objectname " )
```

  - This creates an instance of the named object

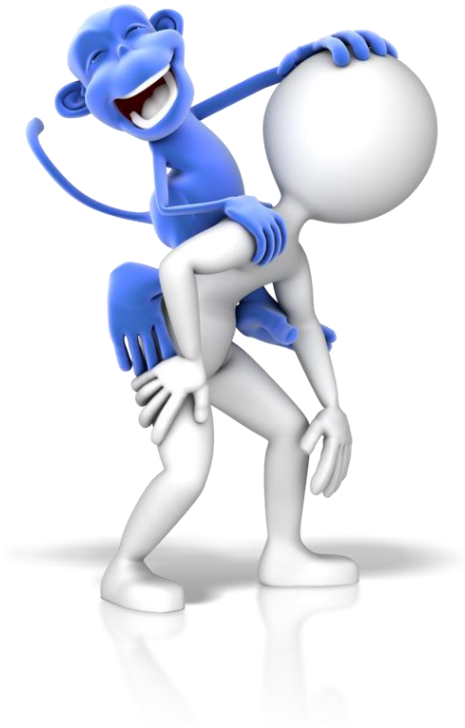
- To delete a reference to an object

```
DROP OBJECT objHandle
```

  - Frees object data
  - Drops reference to object methods

Automatic Housekeeping

- What's the problem?
  - You have to remember to DROP objects when you are done with them
- How does PxPlus get this monkey off your back?
  - Ties object to other components
    - Current Program
    - Files
    - Controls
    - Windows
    - Other Objects



Automatic Housekeeping

Add the relationship to the NEW function

objHandle = NEW("objectname " FOR PROGRAM)

- Drops object when program ends/exits

objHandle = NEW("objectname " FOR FILE *nnn*)

- Drops object when file is closed

objHandle = NEW("objectname " FOR WINDOW)

- Drops object when window is closed

objHandle = NEW("objectname " FOR CONTROL *nnn*)

- Drops object when control deleted

objHandle = NEW("objectname " FOR OBJECT *nnn*)

- Drops object when other object is dropped

Automatic Housekeeping

- Advantages
  - Reduces risk of forgetting to free objects
  - Less coding required
  - Provides method to tie logic to closing file, window or any of the FOR xxx components.

Automatic Housekeeping

- Enabled the new "*plus/obj/wrapup"
  - Object allows program call or EXECUTE to occur when component deleted

```
o = NEW( "*plus/obj/wrapup", "program", "param"  FOR xxx)
```

- CALLs specified program passing it "param" when *xxx* is closed/deleted.

```
o = NEW( "*plus/obj/wrapup", "", "directives"  FOR xxx)
```

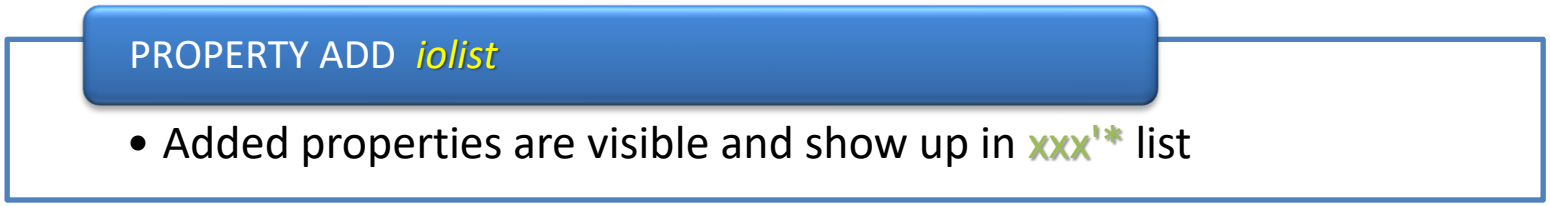
- EXECUTEs specified directives when *xxx* is closed/deleted.

- Typical uses
  - Reset parameters/precision, close files, write log

Dynamic Properties

- Externally visible properties normally fixed
  - Defined by the object definition
    - Declared by PROPERTY directive in class definition
  - Same properties for all instances of the object

- PxPlus provides following directive to add properties



Dynamic Properties

- Simplifies generic "File" handler objects

```
        def class "FileIO"
        local fileno
!
        function Open(pathname$)
        enter p$
        open object (hfn,iol=*)p$
        fileno=lfo
        add property iol=iol(lfo)
        return 1
!
        function Read()
        read (fileno)
        return 1
!
        function ReadByKey(TheKey$)
        enter k$
        read (fileno,key=k$)
        return 1
!
        end def
```

Accept Properties

- Options on the DEF CLASS

```
DEF CLASS "className" ACCEPT PROPERTIES
```

  - Allows external programs to dynamically create properties
    - May be string or numeric properties
    - Creation of property done on first assignment
    - Reports error if never defined

Accept Properties

- Great use – Create a Data object

  - Instead of passing multiple data values you simply pass the object handle
  - Make it global for system wide values

  - The %nomads system object accepts properties

Merge Objects

- Including/inheriting object normally done using LIKE directive
  - Inherits properties and methods from another object
  - Controlled by object class definition
- Merge object provides comparable function
  - Done at run time
  - Merges instances of object as opposed to class

Merge Objects

- For example, change messages based on some runtime criteria such as user

```
    def class "client"
    property name$, zipcode$, state$
!
    function Validate()
    if nul(name$) \
     then print "Name ",_obj'msgRequired$;
          return 0
!
    state$=ucs(state$)
    if pos(state$="NYNJFLOHCATX",2)=0 \
     then print "State ",_obj'msgUnknown$;
          return 0
!
    if num(zipcode$,*)=0 \
     then print "Zipcode ",_obj'msgNotNumeric$;
          return 0
!
    if zipcode$="" or state$="" \
     then print _obj'msgError$;
          return 0
!
    return 1
    end def
```

```
    def class "english"
    property msgUnknown$="Is not a known value"
    property msgNotNumeric$="Is not numeric"
    property msgRequired$="Is a mandatory field"
    property msgError$="**** System error ***"
    end def
```

```
    def class "spanish"
    like "english"
    property msgUnknown$="No se conoce un valor"
    property msgNotNumeric$="No es numérico"
    property msgRequired$="Es un campo obligatorio"
    end def
```

Embedded IO

- Based on standard embedded IO interface
  - Perform/Call of entry points replaced with Method calls of same name
  - Invoked by specifying **obj=xxxxxx** as embedded IO program in **Data Dictionary Maintenance**

Obj=MyEmbededIO

  - Object dynamically created and dropped on OPEN and CLOSE
    - Allows logic to maintain data attached to file
    - Special method "Open_File" called on open
      - Passed file pathname
  - Helps preserve integrity as logic NOT **PERFORM**ed

Attach Objects to Controls

- Allows methods and properties to be attached to control

```
MyGrid.CTL'ObjectID = NEW("objectName")
```

  - Reference to control goes to Object first then physical control
    - Property with same name yields Object Property
    - Object properties and methods don't show in control property list
  - Typical uses
    - Validation logic
    - Original value for control
    - Control description for error message
    - Security information

Object Defined Controls

- Allows Objects to take place of controls
  - Buttons, Radio Buttons, List boxes, Grids, etc.
  - Server object used to create the objects in response to Control creation directives
  - All control properties come from object
  - All control directives converted to method calls

- This concept is the basis for ***i*Nomads**
  - Replaces controls with Objects that create HTML
  - Replaces directives and property details with JavaScript functionality

Object Caching

- Objects often get created and destroyed
  - Example: Invoice lines
    - During load of Invoice system creates multiples
    - Advance to next invoice deletes ALL lines
      - Object definitions with no references are dropped
  - Class definition doesn't often change
    - PxPlus Caches unreferenced definitions
    - Avoids reloading/processing the Class definition
    - Speeds object creation
    - Default is 10 object definitions
      - Controlled by '+J' parameter

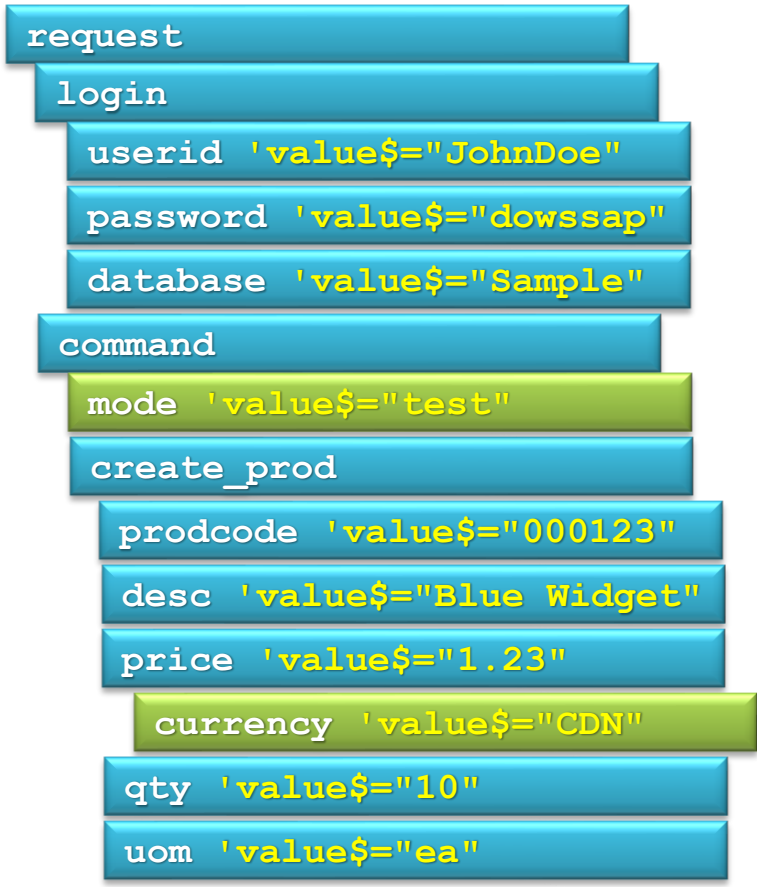
XML Parser *obj/xml

- Simple OOP based utility to parse, edit, and create XML
  - XML is broken down into a tree structure of objects (Nodes)
  - Simple methods to:
    - Add a new "Node"
    - Delete a "Node"
    - Find a "Node" based on its "name"
      - Also Find next and by ordinal number
    - Parse and rebuild XML
  - Other capabilities
    - Read/change a nodes value
    - Add/Edit/Change node attributes

XML Parser *obj/xml

- Visual of a Node tree

```
<request>
 <login>
  <userid>JohnDoe</userid>
  <password>dowssap</password>
  <database>Sample</database>
 </login>
 <command mode="test">
  <create_prod>
   <prodcode>000123</prodcode>
   <desc>Blue widget</desc>
   <price currency="CDN">1.23</price>
   <qty>10</qty>
   <uom>ea</uom>
  </create_prod>
 </command>
</request>
```



Additional Resources

The help link(s) below refer to the current on-line help pages. The functionality may have been further updated since the PxPlus 2018 (version 15) release

- Object-Oriented PxPlus
- PxPlus OOP Interface
- OOP Syntax Elements
- Putting It All Together

- XML Parser (*obj/xml)