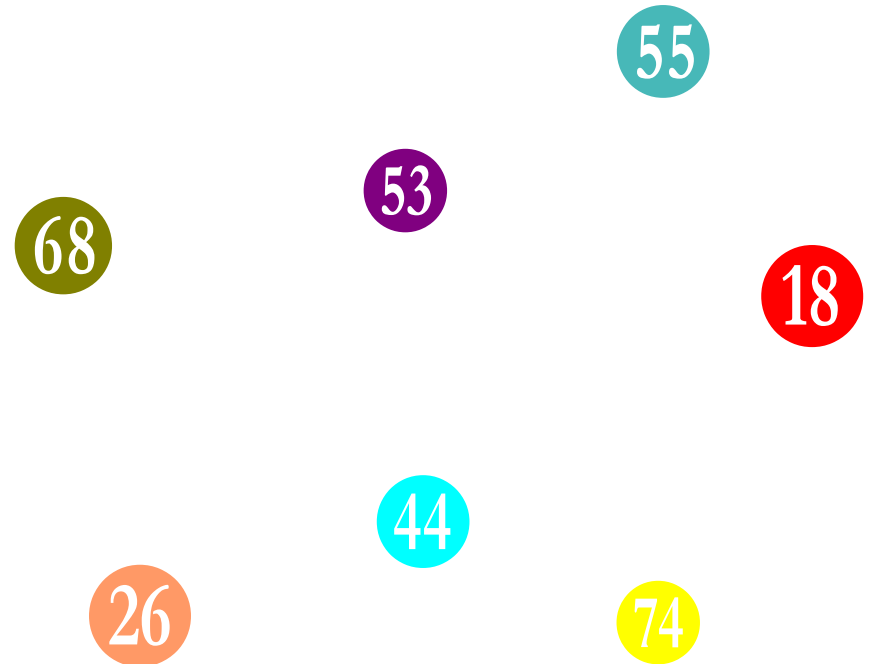


# Talkin' Bout Enumerations

**Alex Kodat**  
**Sirius Software Inc.**



Sirius Software, Inc.

# What Are Enumerations?

- They're attributes of objects or actions with discrete values
  - Married, single, divorced, widowed, etc. for example
  - True or false, for example
  - Yes, no, or maybe, for example
  - Paper or plastic, for example
  - Pending, shipped, or delivered, for example
- Number of values relatively small (fewer than 10 values as a rough guideline) and relatively static
- A special kind of class



# Traditional Enumerations

- Were simply static integer variables:

```
%maritalStatusSingle    is fixed static initial(0)
%maritalStatusMarried   is fixed static initial(1)
%maritalStatusDivorced  is fixed static initial(2)
%maritalStatusWidowed   is fixed static initial(3)
```

- You could do this with UL if you wished
  - Variables would have to be *Common* to be useful
- Names had to be unique
  - So long and clunky
- This is the current VB.Net approach (more or less) and the old Java approach



# But Really, Enumerations are Special Kinds of Classes

- The class is defined by the possible values of the enumeration
- And extra properties of the values
  - For example, an EverMarried property of a MaritalStatus enumeration
  - Or a StatusCode property of MaritalStatus
- And things you can do with with the enumeration
  - ToString is the most common
    - Although maybe this is really a property, too
- You can declare variables of an enumeration class
  - Most commonly used for method parameters



# Example Enumeration: LockStrength

- Used by many file object methods
- Can have values *None*, *Share*, and *Exclusive*



# Use of the LockStrength Enumeration

```
%record      is object record in foobar  
%recordset   is object recordset in foobar  
...
```

\* Going to update record, so get an exclusive lock

```
%record = new(%recnum, exclusive)  
...
```

\* If recordset not locked, need to do extra stuff to avoid “accidents”

```
if %recordset:lockStrength eq none then  
...
```



# So Why Use Enumerations?

- Case independence
  - Yes, methods could be case independent on input parameters, but how do you do case-independent comparisons?
    - \$upcase everything? Yuck
- Compile time error checking
  - If I type `if %maritalStatus eq 'Divorcd'` then I have a bug
  - If I type `if %maritalStatus eq Divorcd` then I have a compile error
- Makes clear that the code is dealing with a special limited set of values



# Using Enumeration Values

- The same value can be used in different enumerations
  - For example, *None* might be used in `LockStrength` and in, say, a `MilitaryRank` class
- Strictly speaking, enumerations should be written in class fully qualified format: `%(militaryRank):none`
- But, in almost all contexts, the compiler can determine the class of a value from context:  
`if %customer:militaryRank eq general then`
- So class-qualification almost never necessary
  - Except in really weird cases
  - Though always allowed



# System Enumerations

- Boolean: True, False
- LockStrength: None, Share, Exclusive
- XmlNodeType: Attribute, Comment, Element, Namespace, PI, Root, Text
- XmlNamespaceSetting: Ignore, None, On
- AttentionKey: Enter, Clear, PF1, PF2, etc.
- FieldColor: Blue, Red, Pink, Green, Turquoise, Yellow, White
- Available from Day 1 of SOAP ULI O-O support



# The Boolean Enumeration

- Perhaps the most heavily used enumeration
- Automatically converted to 0 or 1 in If tests:

```
%needToRecalc  is boolean
```

```
...  
if %needToRecalc then
```

```
...
```

- Note that *Enumeration* keyword not required for boolean declaration
  - It is required for all other declarations:

```
%currentNodeType  is enumeration xmlNodeType
```



# Booleans and Named Parameters: A Match Made in Heaven

- Makes code much clearer:

```
%order:notifyCustomer(backorder=true)
```

is much clearer than

```
%order:notifyCustomer(1)
```

or even

```
%order:notifyCustomer(backorder=1)
```

- Makes method “switches” self-documenting
  - Other enumerations accomplish the same thing though less commonly



# Null Values for Enumeration Variables

- Because enumerations are just a special class, variables of that class can have null values.
- So, if we have  
`%sendWarning` is boolean  
the value of `%sendWarning` is *Null*
- So the test  
`if %sendWarning then`  
would throw an error but  
`if %sendWarning eq true then`  
would not
- Bug or feature?
  - Can catch unset values
  - But slightly weird inconsistent behavior



# Initial Values for Enumeration Variables

- Available in Sirius Mods 7.0.
- Can specify initial value on enumeration declaration:  
`%sendWarning is boolean initial(false)`
- So now you can control which behavior you want
  - No initial value can often catch mistakes where you forgot to set a value
  - Initial value can be used to set natural default value for enumeration variable
- Perhaps even more useful – default values on enumeration method parameters

```
%subroutine notifyCustomer( -  
    %backorder is boolean default(true) -  
    namerequired)
```



# User Language Enumerations

- Finally available in Sirius Mods 7.0
  - Sauce for the goose...
- Declared with an *Enumeration* block
  - Very similar to a class block
  - Has Public, Public Share, Private, Private Shared sections
  - Each section can have methods: Subroutine, Function, and Property
  - But no Variables allowed
  - Public section can (should) have *Values* declared



# Sample User Language Enumeration (idea lifted from java.com web site)

```
enumeration suit
  public
    value hearts
    value diamonds
    value spades
    value clubs
    function isRed    is boolean
    function isBlack is boolean
  end public
function isRed is boolean
  if %this eq hearts or %this eq diamonds then
    return true
  else
    return false
  end if
end function
function isBlack is boolean
  if %this eq spades or %this eq clubs then
    return true
  else
    return false
  end if
end function
```



# Another Way to Skin The IsBlack Cat Using the Ordinal Method

```
enumeration suit
  ...
  function isBlack is boolean
    jump to (red, red, black, black) %this:ordinal
    red:
      return false
    black:
      return true
  end function
end enumeration
```



# The Ordinal Function for Enumerations

- An implicit method that's always available **inside** the enumeration class declaration
- Dependent on order of declaration of Values
  - First value has ordinal 1, second has 2, etc..
  - So be careful in using this
- Can be used for order comparisons:

```
if %myCard:suit:ordinal gt %yourCard:suit:ordinal then
```
- Can be made available **outside** enumeration declaration with *Allow Ordinal* in the Public block
  - Allows computed jumps and ordinal comparisons
  - Make sure you really want to allow this
  - Currently no system enumerations Allow Ordinal



# An Enumeration Where It Might Make Sense to Allow Ordinal

```
enumeration rank
  public
    allow ordinal
    value deuce
    value three
    value four
    value five
    value six
    value seven
    value eight
    value nine
    value ten
    value jack
    value queen
    value king
    value ace
  end public
end enumeration
```



# The ToString Method

- Converts enumeration value to string
  - For debugging
    - Common mistake: `print %maritalStatus`
    - Correct: `print %maritalStatus:toString`
  - For conversion to XML
- Implicit ToString method always provided for UL enumerations
  - Slightly unusual: Uses case used in Value declaration
  - Can be overridden with explicit ToString function



# Summary

- Enumerations can be useful in making code more self-documenting and error free
- Several new useful enumeration capabilities in Sirius Mods 7.0
  - User Language enumerations
  - Enumeration Initial and Default values
- As usual, at leading edge of current technology in making enumerations classes
  - Along with Java, in this case
    - Though Java has **slight** edge with value iterators (rarely used in real life)
  - Ahead of VB.Net and C++
- More enhancements coming



# Let's Play Around with Enumerations a Bit



Sirius Software, Inc.