The ooRexx decimalFormat Class

Date: November 20, 2007
Author: Lee Peedin
Purpose: To provide a simple means to format a decimal number.
Requires: decimalFormat.cls
Version: Beta .4.1

Methods

new	formatter = .decimalFormat~new(optional pattern) If pattern is not specified, the default pattern "#,###.##' will be used.
format	a_result = formatter~format(number) Returns a result with "number" formatted according to the set pattern.
getVersion	a_result = formatter~getVersion Returns the current version of the decimalFormat class

Attributes

groupingSize	get	a_result = formatter~groupingSize (Default is 3)
	set	formatter~groupingSize = numeric whole number
decimalSeparator	get	a_result = formatter~decimalSeparator (Default is '.')
	set	formatter~decimalSeparator = single length character that is not a duplicate of the groupingSeparator.
groupingSeparator	get	a_result = formatter~groupingSeparator (Default is ',')
	set	formatter~groupingSeparator = single length character that is not a duplicate of the decimalSeparator.
pattern	get	a_result = formatter~pattern (Default is '#,###.##')
	set	formatter~pattern = a valid pattern as described below
pPrefix	get	a_result = formatter~pPrefix Returns the current prefix for a positive number
	set	formatter~pPrefix = '$' Sets the prefix for a positive number
pMask	get	a_result = formatter~pMask Return the *mask* portion of the positive pattern.
	set	formatter~pMask = '#,##0.00' Sets only the *mask* portion of the positive pattern.

pSuffix	get	a_result = formatter~pSuffix Returns the current suffix for a positive number
	set	formatter~pSuffix = ‘ DB’ Sets the suffix for a positive number
pGrouping	get	a_result = formatter~pGrouping Return either .true or .false
	set	formatter~pGrouping = .true/.false Sets the positive grouping to the logical value indicated
nPrefix	get	a_result = formatter~nPrefix Returns the current prefix for a negative number
	set	formatter~pPrefix = ‘$’ Sets the prefix for a negative number
nMask	get	a_result = formatter~nMask Return the *mask* portion of the negative pattern.
	set	formatter~nMask = ‘#,##0.00’ Sets only the *mask* portion of the negative pattern.
nSuffix	get	a_result = formatter~nSuffix Returns the current suffix for a negative number
	set	formatter~pSuffix = ‘ DB’ Sets the suffix for a negative number
nGrouping	get	a_result = formatter~nGrouping Return either .true or .false
	set	formatter~nGrouping = .true/.false Sets the negative grouping to the logical value indicated
zPattern	get	a_result = formatter~zPattern Returns the current pattern for a zero number
	set	formatter~zPattern = ‘[0]’ Sets the pattern for a zero number

Patterns

A pattern can be from 1 to 3 sub-patterns with the sub-patterns separated by a semi-colon ';'.

The first sub-pattern will be applied to positive values.
The second sub-pattern will be applied to negative values.
The third sub-pattern will be applied to a 0 value.

If only one sub-pattern is specified, the same pattern will be applied to positive, negative, and 0 values. A negative value will be preceded with the default minus sign '-'. A 0 value is treated as a positive value.

The positive and negative sub-patterns can have from 1 to 3 parts:
Part 1 – A string prefix enclosed in quotes
Part 2 – A pattern "mask"
Part 3 – A string suffix enclosed in quotes

Note: A negative sub-pattern does not have to repeat the mask, if the same mask is to be used – only the prefix & suffix need be supplied.

The zero sub-pattern has 1 part:
Part 1 – A string value enclosed in quotes

Pattern Masks

Each pattern mask has 4 reserved symbols

\# A pound/hash symbol represents an "expendable" place holder. If the formatted result has a corresponding value, the # will be replaced with the corresponding value.

0 A zero represents a "non-expendable" place holder. If the formatted result has a corresponding value the 0 will be replaced with the corresponding value. If there is not a corresponding value, the 0 will remain as the place holder.

, A comma signifies that each grouping is to be applied

. A period signifies the integer and decimal separator. Results are rounded, according to ooRexx format rules, based on the number of place holders in the decimal portion of the number argument.

Note: If a filler string is required, it should be part of either the prefix or suffix.

Pattern Examples and Corresponding Results

```
/* demo_decimalFormat.rex */
    call SysCls

-- Grouping applied
-- 2 decimal places, if needed (input value is rounded)
    f = .decimalFormat~new(',.##')
    say f~format(1234.567)      --> 1,234.57
    say f~format(1234.5)        --> 1,234.5
    say
    say f~format(-1234.567)     --> -1,234.57
    say f~format(-1234.5)       --> -1,234.5
    say

-- Grouping applied
-- 2 decimal places, even if not needed
    f = .decimalFormat~new(',.00')
    say f~format(1234.56)       --> 1,234.56
    say f~format(1234.5)        --> 1,234.50
    say
    say f~format(-1234.56)      --> -1,234.56
    say f~format(-1234.5)       --> -1,234.50
    say

-- No Grouping
-- leading zero if needed
-- 2 decimal places, even if not needed
-- 1 leading zero, if needed
    f = .decimalFormat~new('0.00')
    say f~format(1234.56)       --> 1234.56
    say f~format(1234.5)        --> 1234.50
    say f~format(3/6)           --> 0.50
    say
    say f~format(-1234.56)      --> -1234.56
    say f~format(-1234.5)       --> -1234.50
    say f~format(0 - (3/6))     --> -0.50
    say

-- Grouping
-- 2 trailing zeros, if needed
-- 1 leading zero, if needed
-- US currency
    f = .decimalFormat~new('"$",0.00')
    say f~format(1234.56)       --> 1,234.56
    say f~format(1234.5)        --> 1,234.50
    say f~format(3/6)           --> 0.50
    say
    say f~format(-1234.56)      --> -1,234.56
    say f~format(-1234.5)       --> -1,234.50
    say f~format(0 - (3/6))     --> -0.50
    say
```

```
-- 2 trailing zeros, if needed
-- 1 leading zero, if needed
-- Accounting
    f = .decimalFormat~new(',0.00" DB";,0.00" CR"')
    say f~format(1234.56)       --> 1,234.56 DB
    say f~format(1234.5)        --> 1,234.50 DB
    say f~format(3/6)           --> 0.50 DB
    say
    say f~format(-1234.56)      --> -1,234.56 CR
    say f~format(-1234.5)       --> -1,234.50 CR
    say f~format(0 - (3/6))     --> -0.50 CR
    say

-- 2 trailing zeros, if needed
-- 1 leading zero, if needed
-- Euro currency
    f = .decimalFormat~new('"?",0.00;"-?"')
    f~groupingSeparator = '.'
    f~decimalSeparator = ','
    say f~format(1234.56)       --> ?1.234,56
    say f~format(1234.5)        --> ?1.234,50
    say f~format(3/6)           --> ?0,50
    say
    say f~format(-1234.56)      --> -?1.234,56
    say f~format(-1234.5)       --> -?1.234,50
    say f~format(0 - (3/6))     --> -?0,50
    say

-- Group positive numbers
-- No grouping for negative numbers
-- Return [0] for zero values
    f = .decimalFormat~new(',0.00;"-"0.00;"[0]"')
    say f~format(1234.56)       --> 1,234.56
    say f~format(-1234)         --> -1234.00
    say f~format(0)             --> [0]
    say

-- Group all numbers
-- Place parens around negative numbers
    f = .decimalFormat~new(',0.00;"("")"')
    say f~format(1234.56)       --> 1,234.56
    say f~format(-1234)         --> (1,234.00)
    say f~format(0)             --> 0.00

::requires 'decimalFormat.cls'
```