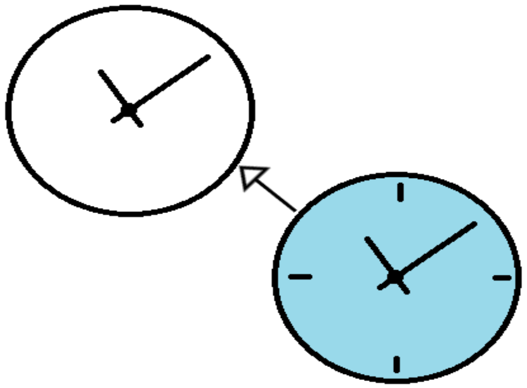
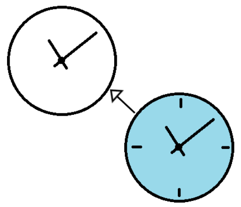


Subclassing the ooRexx dateTime class

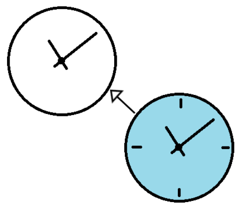
Jon Wolfers

History



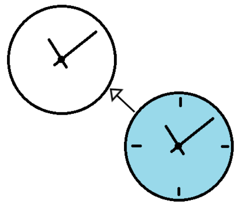
- 2001 – Using ooRexx with APS Paypoint 6 till
- Dates in the format dd/mm/yyyy
  - Wrote a function to convert them to standard date, probably:
  - `return changeStr('/',translate('1234567890',arg(1),'9086751234'),'-')`
- Later: dateTime class was added to ooRexx

dateTime class

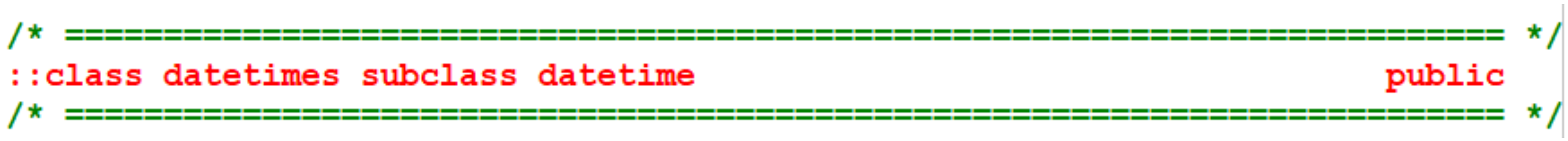


- Represents a point in time
- Can be created from the standard rexx date and time string representations
- Can present itself as the standard rexx date and time string representations
- Is immutable
- Can take part in date/time arithmetic

Subclassing



- Primarily thought of as way to specialise
- Allows use of all methods of superclass
- Allows over-riding for specialisation
- Allows adding new methods to extend

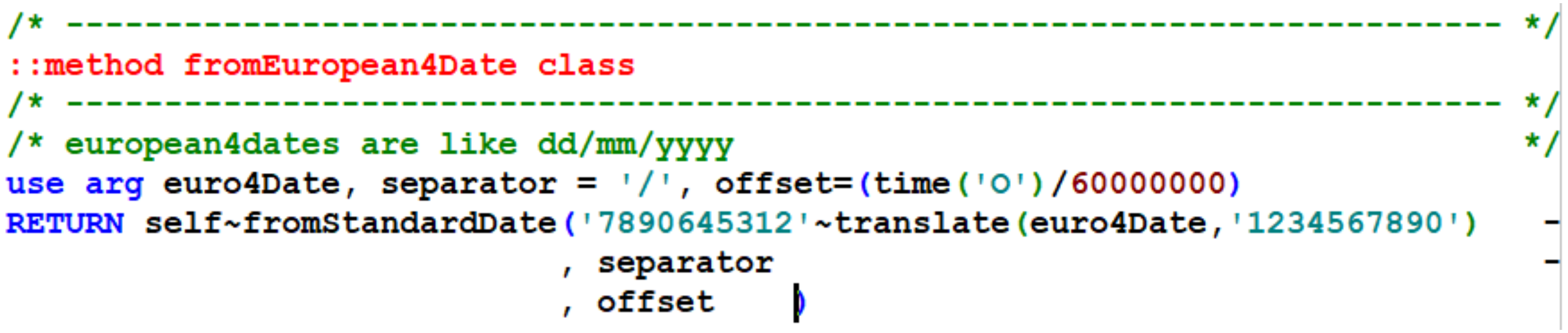


```
/* ========================================================================= */
::class datetimes subclass datetime                                   public
/* ========================================================================= */
```

- datetimes is a sub-class of datetime

Adding a constructor for european4date

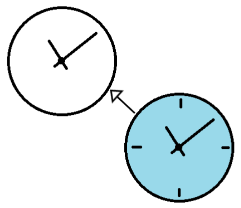
- Constructor is a class method for creating an instance



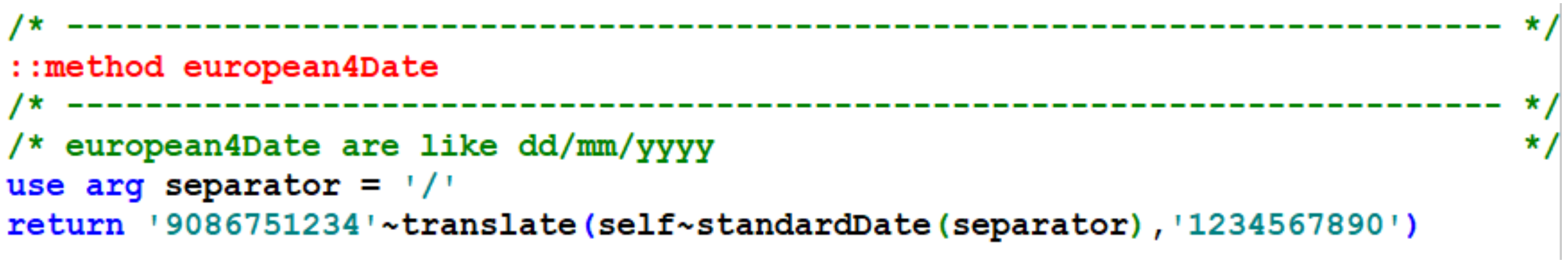
```
/* ---------------------------------------------------------------------- */
::method fromEuropean4Date class
/* ---------------------------------------------------------------------- */
/* european4dates are like dd/mm/yyyy                                     */
use arg euro4Date, separator = '/', offset=(time('O')/60000000)
RETURN self~fromStandardDate('7890645312'~translate(euro4Date,'1234567890')   -
                            , separator                                       -
                            , offset    )
```

- `myDatetimes = .datetimes~fromEuropean4date('22/09/2019')`

Providing a toString method for european4date



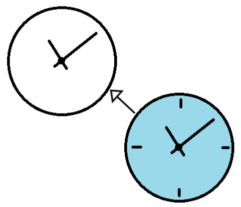
- This is an instance method for creating a string representation



```
/* ------------------------------------------------------------------ */
::method european4Date
/* ------------------------------------------------------------------ */
/* european4Date are like dd/mm/yyyy                                  */
use arg separator = '/'
return '9086751234'~translate(self~standardDate(separator),'1234567890')
```

- That's it – 3 directives & 4 lines of code and datetime is extended to handle a new format.

Immutability and the subclass



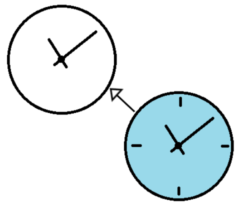
```
  a = .datetime~new
  b = .datetimes~new

  c = a~addDays(1)
  d = b~addDays(1)                  -- addDays is a method of the superclass

  say 'c is an instance of' c~class -- c is an instance of The DateTime class
  say 'd is an instance of' d~class -- d is an instance of The DATETIMES class

::requires 'datetimes.cls'
```

Converting a datetime to a datetimes instance

- Datetimes has access to all methods of datetime, but not vica versa
- fromDateTime allows one to create a datetimes from a datetime

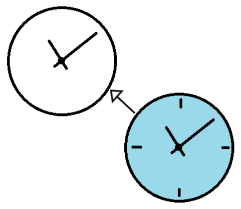
```
   a = .datetime~fromstandardDate('2019-01-01','-')
   b = .datetimes~fromDatetime(a)
   say b~class b~european4Date      -- returns The DATETIMES class 01/01/2019

::requires 'datetimes.cls'
```

Now we see how easy it is…



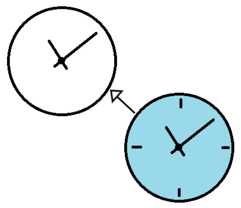
- datetime is specific about how an ISO datetime should be formatted, so is MySql
    - yyyy-mm-ddThh:mm:ss.uuuuuu
    - yyyy-mm-dd hh:mm:ss
- USA date and Ordered date

Now let’s generalise it



- Imagine a universal constructor method ‘**from**’ that you told what format to accept
- And a universal toString method ‘**to**’ that would build the output to your format

Format strings

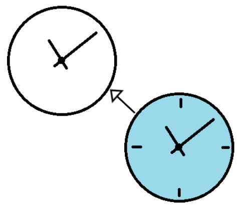


- ooDialog already has a class that uses date/time format strings as defined by Microsoft
- For our purposes we need some new ones…

Format strings not included in ooDialog documentation

D	The day followed by suffix - ie 9th
q	The quarter number (1 – 4)
s	The one- or two-digit seconds *
ss	The two-digit seconds *
f	Fraction of a second (symbol chosen by Rick McGuire)
*	Not documented in ooDialog but used in underlying control
	Full list of format strings in help document

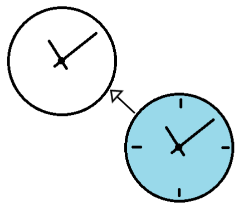
Escape characters



- Literals (for example separators) may be included in the format string
- Where a literal is used that is defined as a format string – it must be escaped
- The escape character is chosen by programmer to be unique within the format string

Some Examples

Period start and finish

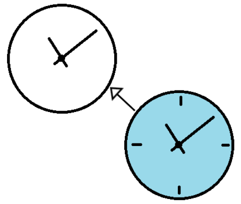


- Reporting often requires date/time calculations to find start and end of periods
- Add '**startOf**' and '**endOf**' methods
- Period may be offset from current period

Minute	Hour
Day	Week
Month	Quarter
Year	Decade
Century	Millennium

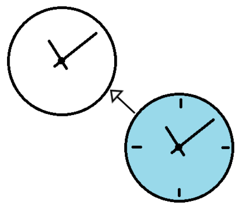
Some Examples

Coordinated Universal Time



- UTC (Coordinated Universal Time) recognises time zones
- Times include an offset of hours and minutes from GMT
- FromUTCdate constructor accepts an offset yyyy-mm-ddThh:mm:ss.uuuuuu**+hhmm**
- All other datetime (and datetimes) constructors can accept an offset in minutes

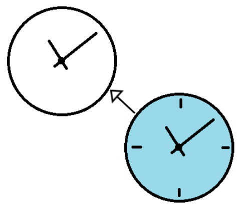
Summertime Blues



- dateTime provides a method toLocalTime which returns an instance representing the local time
- BUT it doesn't work across daylight savings boundaries
- The EU provide an algorithm for DS boundary dates (US does too, but it is different)
- toLocalTimeEU works across DS boundarys

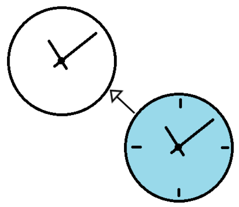
An example

A Bonus

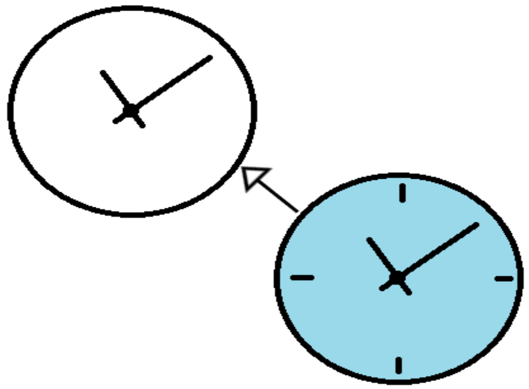


- Working out daylight savings boundaries required finding the last Sunday in the month
- I provided a general method nthWeekdayOfMonth to work out 1st, 2nd , last… Occurrence of a particular day in the month referred to be a datetimes instance

Where can I get this?



- Datetimes.cls is in my sandbox on the ooRexx site
- …BUT…
- Since I started work on this Rick has built much of the new functionality shown here into the language. Coding is done, but not yet documentation.



Subclassing the ooRexx dateTime class

Jon Wolfers

The End