Totalising Tables and Streaming Databases – Subclassing ooRexx

- Jon Wolfers – Rexx developer since 1980s
- Came to ooRexx in 2001
- Following tutorial on RexxLa Wiki (Jan/Feb 2008) most questions were about subclassing
ooRexx subclasses

- Subclasses inherit methods of parents
- Parents can be your classes or built-in
- Eventually all classes have built-in parent
- Some tasks can be achieved quickly and easily by subclassing the ooRexx built-in classes
Two case studies

- Building a totaliser class from the built-in table class
- Building a class to handle Dbase data from the stream class
Two case studies

- Concepts I want to introduce
  - the subclass keyword on the class directive
  - the forward keyword
  - the unknown method
The Totaliser

Often when reporting one cursors through data
# The Totaliser

<table>
<thead>
<tr>
<th>Garment</th>
<th>Colour</th>
<th>Qty</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKS</td>
<td>RED</td>
<td>4</td>
<td>8.50</td>
</tr>
<tr>
<td>SHIRTS</td>
<td>BLUE</td>
<td>2</td>
<td>12.98</td>
</tr>
<tr>
<td>SOCKS</td>
<td>BLUE</td>
<td>2</td>
<td>2.80</td>
</tr>
<tr>
<td>SHIRTS</td>
<td>GREEN</td>
<td>3</td>
<td>22.00</td>
</tr>
<tr>
<td>SHIRTS</td>
<td>RED</td>
<td>1</td>
<td>23.99</td>
</tr>
</tbody>
</table>
Initialise counters
sort data by section
last_section = ''
Do line over data
    parse data ...
    if section \= last_section
    then do
        if last_section \= ''
        then output section totals
        initialise section counters
        output section section header
    end
    increment counters
    output row
End
Output last section total
Output grand totals
The Totaliser

It would help to have an object that:
- initialised the counters automatically
- allowed one to add quantities
- allowed one to retrieve totals
- as a bonus provided other statistics (grand totals, percentages, mins and maxes ...)
The Totaliser

The built in table class

- Can contain various totals **BUT**
  - Uninitialised indexes have .nil value
  - There is no method to add a quantity to a held value
The Totaliser

- Lets subclass the Table Class

```cpp
::class totaliser subclass table public
```
The Totaliser

Let's subclass the Table Class

::class totaliser subclass table public

This is a directive telling ooRexx that we are defining a class.

:::requires directives must appear before all other directives
The Totaliser

- Lets subclass the Table Class
- ::class totaliser subclass table public
  - This is the name we are giving our new class
The Totaliser

Let's subclass the Table Class

`::class totaliser subclass table public`

These two short words give all the methods of the table class to our class
The Totaliser

Let's subclass the Table Class

::class totaliser subclass table public

Public means that this class can be accessed from other scripts using the ::Requires directive.
The Totaliser

::attribute grandTotal

The ::attribute directive creates get and set methods for a variable with object scope.

If there is a method of the same name in the superclass this will override it.

As it is this adds the grandtotal and grandtotal= methods to our class.
The Totaliser

:::method init

- When a new instance of a class is created, the init message is sent to it.
- If no such method exists in our class then the chain of superclasses is searched till one is found.
- We want our init method to run, but first we need the table class's init method to run.
The Totaliser

::method init
forward class (super) continue

The forward keyword sends on the message that triggered this method
The Totaliser

::method init
forward class (super) continue

- The forward keyword sends on the message that triggered this method
- class (super) starts the search for a target method at the immediate superclass
The Totaliser

:::method init
forward class (super) continue

- The forward keyword sends on the message that triggered this method
- class (super) starts the search for a target method at the immediate superclass
- continue specifies that after the action initiated by the message forwarding is complete, continue with this method
The Totaliser

::method init
forward class (super) continue
self~grandtotal = 0

- Now we can initialise our attribute
- unInitialised variables in rexx take the value of their name in uppercase – which causes an error when you add a number to them!
The Totaliser

- We need method to accumulate values
  - Table class has put method
  - We will create a new method called add
The Totaliser

::method add
expose grandTotal

Grand total is an attribute

- We could access it via messages or expose it
The Totaliser

::method add
expose grandTotal
use strict arg amt,index

use instead of parse passes objects not strings

strict means we will get error if args missing
The Totaliser

```
::method add
expose grandTotal
use strict arg amt,index
  if \datatype(amt,'n')
  then raise syntax 26.900 array ,
     ('Amount to add must be a number, found' amt)
```

⚠️ We should check that what we are totalising is numeric
The Totaliser

::method add
expose grandTotal
use strict arg amt,index
  if \self-hasIndex(index) then self-put(0,index)

- If we haven't seen this index before we initialise a new totaliser
- As the totaliser class does not have these methods, the messages will be forwarded to the superclass (table)
The Totaliser

::method add
expose grandTotal
use strict arg amt,index
    if \self~hasIndex(index) then self~put(0,index)
    self~put(self~at(index) + amt,index)

We do the totalising
The Totaliser

```rexx
::method add
expose grandTotal
use strict arg amt,index
  if \self~hasIndex(index) then self~put(0,index)
  self~put(self~at(index) + amt,index)
grandTotal += amt
```

We look after the grandtotal

This could have been `self~grandTotal += amt`
The Totaliser

We now have an add method

- We still need a method to retrieve our totals
- The table class has a method called at
- The at method returns .nil for uninitialised indexes – we would prefer to return 0
The Totaliser

[] is a standard ooRexxx method name for retrieving a value from an index

- The table class already has a [] method so we are over-riding it
- The message operator is implied when using [], so: table[index] is equivalent to table~"[]"(index)
The Totaliser

::method '[]'
use strict arg index
  total = self~'[]':super(index)

- Here we see the '[]' method is followed by ::super

  ::super is just a variable which the interpreter points to our immediate superclass

  ::super tells the interpreter to only start looking for this method at the superclass level
  - Is this necessary here?
The Totaliser

::method '[]'
use strict arg index
    total = self~at:super(index)
    if total = .nil
        then return 0
    else return total

- If we have totalised this index we return the total
- If not we return 0
The Totaliser

- Our Totaliser is ready to use
- Let's just add a bit of extra value

We can retrieve a total as a percentage of the grand-total
The Totaliser

::method percent
use strict arg index
  dividend = self[index]
  divisor = self~grandTotal

  if divisor = 0
     ,
     | \dividend~datatype('n'),
     | \divisor~datatype('n')
     then pc = 0
  else pc = (dividend / divisor) * 100

return pc~format(,1)|'|'%
The Totaliser

How would we use the totaliser class?

- totaliser demo
- class library
The Totaliser

We need one instance of the totaliser class for each metric

It would be a nice touch if totalisers initialised themselves the way indexes do

Let's build a totalisers class

- This will also give me a chance to demonstrate the unknown method
The Totaliser

```ruby
::class totalisers public
```

There is no subclass keyword, so totalisers is a subclass of the object class directly
This is familiar to us now. We create a table attribute to hold all our totalisers.
The Totaliser

::class totalisers public
::attribute totalisers
::method init
   self~totalisers = .table~new
::method unknown

We use the ooReXX special method called unknown
Any message sent to our class that does not correspond to a method in the subclass chain is sent to the unknown method.

- If there is no unknown method an error is raised.
- The interpreter passes the unknown method two arguments:
  - The name of the method not found
  - An array containing the arguments accompanying the message
The Totaliser

::method unknown
use arg msg, args

totaliser = args[1]

We decree that the first argument is the name of the totaliser the message is for
The Totaliser

::method unknown
use arg msg, args
   totaliser = args[1]
   if \self~totalisers~hasIndex(args[1])
      then self~totalisers[args[1]] = .totaliser~new

If this is a new totaliser then create it
The Totaliser

::method unknown

use arg msg, args

    totaliser = args[1]

    if \self~totalisers~hasIndex(args[1])
    then self~totalisers[args[1]] = .totaliser~new

forward message (msg),
    arguments (args~section(2)),
    to (self~totalisers[totaliser])

Now forward the message to it
The Totaliser

How would we use the totalisers class?

- totalisers demo
- class library