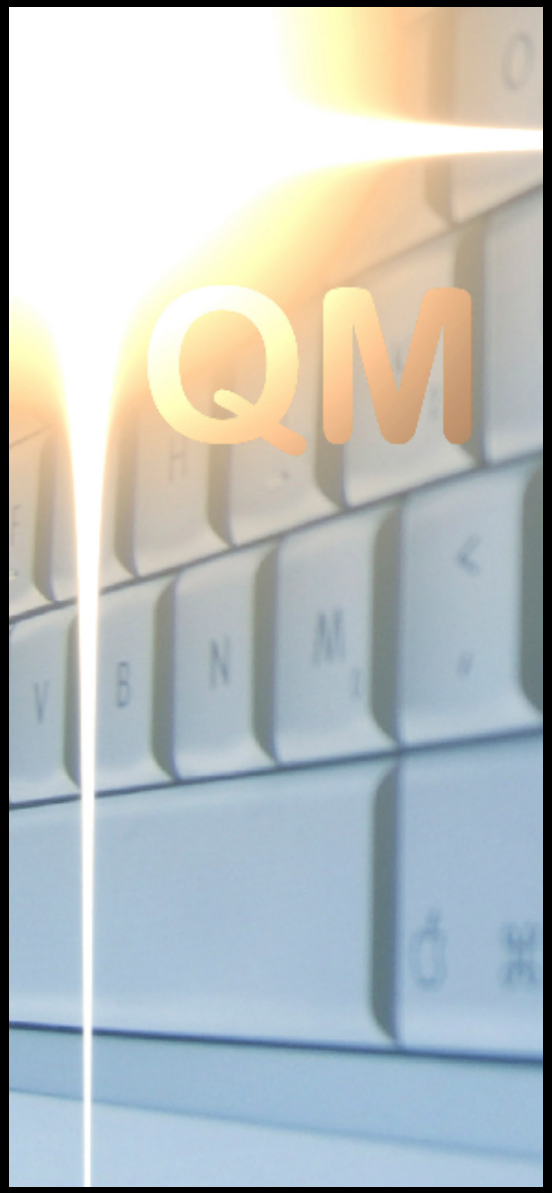


OpenQM

Object Oriented Programming

Martin Phillips
Ladybridge Systems Ltd

OpenQM

What's it all about?

What is an object?

Persistent data

Public subroutines and functions

Inheritance

Destructors and undefined name handlers

OpenQM

What's It All About?

- OO programming does not replace "conventional" methods.
- A new addition to the developer's toolbox.
- An integral part of the QMBasic language.

OpenQM

What Is An Object?

Subroutine:

Program operations that work on supplied data.

Object:

Data that has associated program operations.

OpenQM

What Is An Object?

Defined by a CLASS module.

The CLASS module is a container for...

- The persistent data definitions.
- The program operations that run against this data.

OpenQM

What Is An Object?

An object is a run time instance of the class.

var = OBJECT(“myobj.cls”)

There may be many concurrent instances of the same class each with their own data.

OpenQM

The Objref Operator (->)

References an element of the object.

var->name

var->(*expr*)

OpenQM

Persistent Data

Class modules may use common blocks but these are shared across all instances of the class.

Class modules also have persistent data that is separate for each instance.

OpenQM

Persistent Data

Private data...

- Not visible outside the class module.
- Hides internal operation of the object.

PRIVATE A, B, C(2,3)

OpenQM

Persistent Data

Public data...

- May be visible to programs using the object.

PUBLIC P, Q, R(2,3)

PUBLIC X READONLY

OpenQM

Persistent Data

Referenced from calling program:

result = var->item

var->item = 12

var->item(3) = 12

OpenQM

Public Subroutines and Functions

- Program operations contained within the class module.
- May access or update persistent data.
- Public subroutines store values or perform tasks.
- Public functions return a result value.

OpenQM

Public Subroutines and Functions

```
PUBLIC SUBROUTINE name
   ... Program operations ...
END
```

```
var->name
```

OpenQM

Public Subroutines and Functions

```
PUBLIC SUBROUTINE name(a,b)
   ... Program operations ...
END
```

```
var->name(x,y)

var->name(x) = y
```

OpenQM

Public Subroutines and Functions

```
PUBLIC FUNCTION name(a,b)
   ... Program operations ...
   RETURN value
END
```

```
p = var->name(q, r)
```

OpenQM

Public Subroutines and Functions

Variable length named argument lists...

```
PUBLIC FUNCTION name(a,b) VAR.ARGS
   ... Program operations ...
   RETURN value
END
```

OpenQM

Public Subroutines and Functions

Variable length unnamed argument lists...

```
PUBLIC FUNCTION name(a, ...)
   ... Program operations ...
   RETURN value
END
```

OpenQM

Public Subroutines and Functions

Access arguments by position...

ARG.COUNT()

ARG(n)

SET.ARG n, value

OpenQM

Dual Identity

A name may refer to a public data item when reading and program operations when writing...

...Or vice versa

Allows easy data validation or event triggers.

OpenQM

Inheritance

One class may want to use the data and public routines of another.

The inherited class remains a “black box” where the outer class cannot see how it works.

OpenQM

Inheritance

Static Inheritance...

CLASS name INHERITS other.class

OpenQM

Inheritance

Dynamic Inheritance...

```
obj = object(“otherclass”)
INHERIT obj
```

OpenQM

Inheritance

Dis-inheritance...

DISINHERIT obj

OpenQM

“Automatic” Handlers

- CREATE.OBJECT
- DESTROY.OBJECT
- UNDEFINED (Subroutine / Function)

OpenQM

“Automatic” Handlers

CREATE.OBJECT

Run when the object is instantiated.

Arguments to OBJECT() are passed to this subroutine.

OpenQM

“Automatic” Handlers

- DESTROY.OBJECT
- Run when the last variable referencing the object is released.
- Guaranteed execution, even at program abort.

OpenQM

“Automatic” Handlers

- UNDEFINED
- Run for references to undefined names.
- Both FUNCTION and SUBROUTINE can exist.
- Caller’s arguments passed, plus name.

OpenQM

Example Class Module

This example will walk through a directory, returning the name and type of each item.

OpenQM

Step 1 – Data Definitions

```
CLASS DIR.CLS
$CATALOGUE
    PRIVATE ENTRIES   ;* From DIR()
    PUBLIC ISDIR READONLY
END
```

OpenQM

Step 2 – CREATE.OBJECT

```
PUBLIC SUBROUTINE CREATE.OBJECT(P)
    ENTRIES = DIR(P)
END
```

OpenQM

Step 3 – Retrieve the next item name

```
PUBLIC FUNCTION NEXT
    NAME = ENTRIES<1,1>
    ISDIR = (ENTRIES<1,2> = 'D')
    DEL ENTRIES<1>
    RETURN NAME
END
```

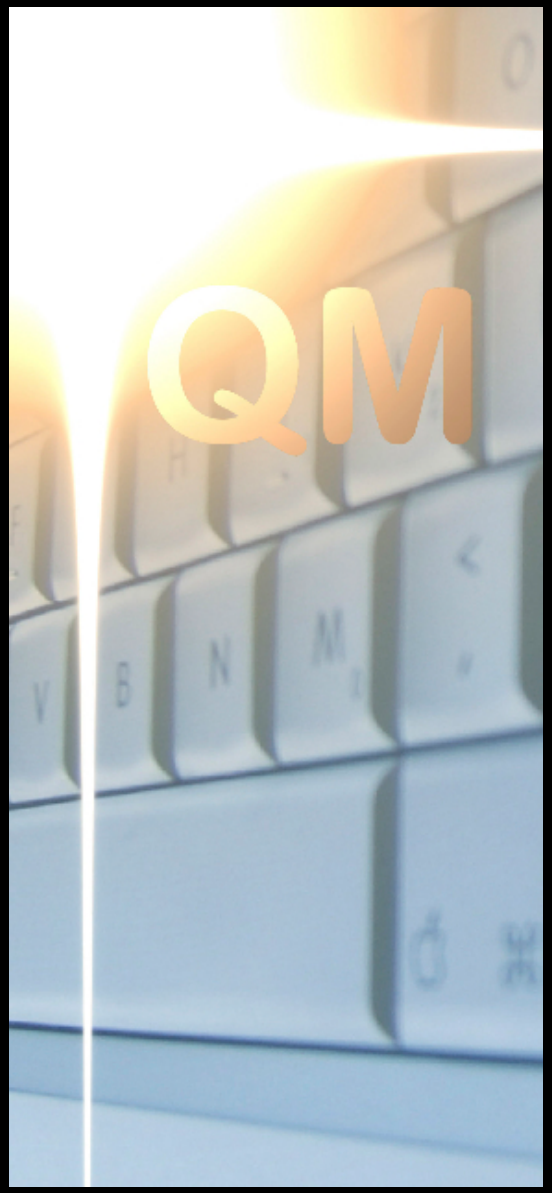
```
CLASS DIR.CLS
$CATALOGUE
    PRIVATE ENTRIES         ;* From DIR()
    PUBLIC ISDIR READONLY ;* Is a directory?

    PUBLIC SUBROUTINE CREATE.OBJECT(PATH)
        ENTRIES = DIR(PATH)
    END

    PUBLIC FUNCTION NEXT
        NAME = ENTRIES<1,1>
        ISDIR = (ENTRIES<1,2> = 'D')
        DEL ENTRIES<1>
        RETURN NAME
    END
END
```

OpenQM

Using the Class

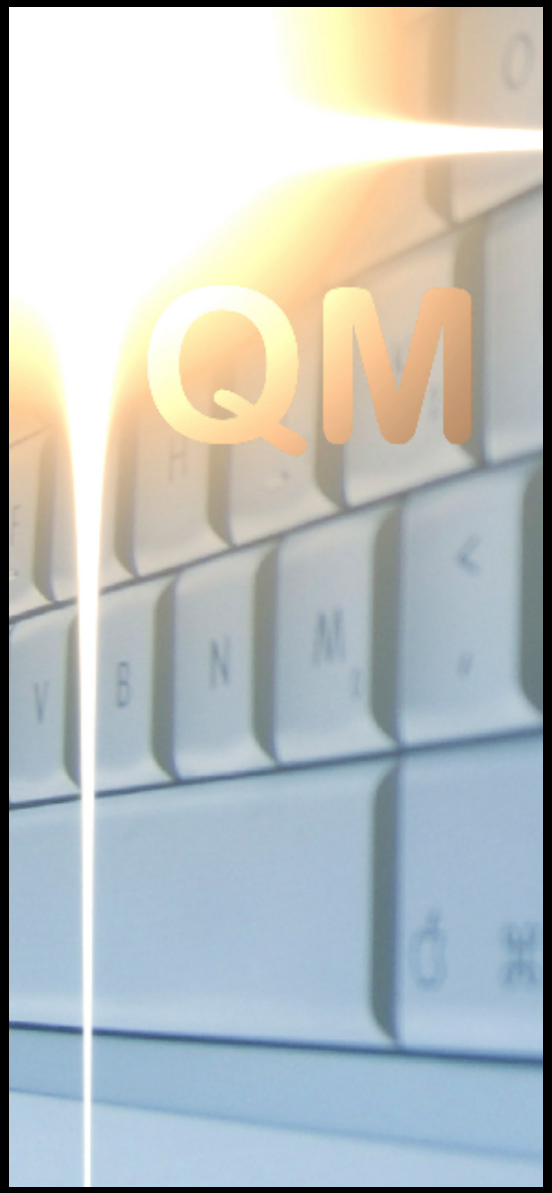
```
DIR = OBJECT('DIR.CLS', '\TEMP')
LOOP
    NAME = DIR->NEXT
UNTIL NAME = ''
    DISPLAY NAME
REPEAT
```

```
PROGRAM OBJECT.DEMO
   GOSUB SCAN(0, '\QMSYS')
   STOP

   LOCAL SUBROUTINE SCAN(INDENT, PATH)
      PRIVATE DIR

      DIR = OBJECT('DIR.CLS', PATH)
      LOOP
         NAME = DIR->NEXT
      UNTIL NAME = ''
         CRT SPACE(INDENT) : NAME
         IF DIR->ISDIR THEN
            GOSUB SCAN(INDENT + 3, PATH:'\'NAME)
         END
      REPEAT

      RETURN
   END
END
```



OpenQM

QUESTIONS?



OpenQM