How To Design Classes for Object Orientated VBA Programming

Object Orientated Excel VBA

Object orientated programming in Excel is primarily through the existing object libraries that are available in the VBA development environment.

However, in order to exercise a greater degree of control over functions by grouping them, and in order to improve re-usability of code, we can re-program them in an object orientated fashion. We create classes that contain data and functions, these can be abstract or semantic data models as described in a previous document.

The benefits of object orientated programming are significant productivity gains for developers and much easier maintenance, as code can be re-used easily, is easier to read and understand, and does not require much code to apply in new projects.

The key concepts for object orientated programming in the Excel VBA environment are:

Classes and Objects
Encapsulation
Inheritance
Polymorphism

Classes and Objects

A class is a collection of data and functions in one re-usable unit. Classes are usually functionally coherent - they correspond to a semantic object model, or effectively carry out a well defined purpose in modelling a data or other abstract entity. For example, if we look at the Excel library, we have the Range class which refers to the cells in the worksheets of a spreadsheet. We could also create a Monte Carlo simulation class which contains the properties and methods required to calculate prices and risk metrics.

Objects are initialised instances of classes: if classes are the blueprint, classes are the 'tangible' version. Classes are initialised into objects using a statement such as

Dim ObjectName as new ClassName

This creates a new instance or object, using the blueprint of the class.

An object has properties - variables that are accessible to other functions or classes that use the class - and methods, which are functions that are sometimes accessible to other classes that use the object.

In Excel VBA there is no need to declare a class as such: we simply create a class module. The name we assign to the class module is the name of the class. It is good practice however, to provide key class details in comment form at the top of the class module.

'Name: Monte Carlo Simulation Class
'Description: Carries out Monte Carlo simulations for different classes of derivatives and provides risk and pricing measures as required
'Inherits From: Analytics Class
'Inherited Classes: Equity_Derivatives, FX_Derivatives, Interest_Rate_Derivatives

Encapsulation

Encapsulation is the ability to limit access to an object's data and functions to specific objects or functions. This enables objects to be self-contained, and increases programmer productivity by hiding how a particular class or method or property is coded. In order to interface with the object, the programmer need know only what it does, not the specifics of how it is coded.

Encapsulation is accomplished in VBA by:
a) Declaring class properties and their accessors. The interfaces with the class can make use of it without knowing the specifics of the data in the properties and how the properties are coded. They simply need to know what the property does.

The accessors are Get and Set properties that retrieve and amend the values of the property variables. They are usually public and are sometimes called interfaces as they enable the class to interact with other data and properties. They set the value of an intermediate variable that is private to the class in the Set property, and retrieve the value of the same variable in the Get property.

Public Property Get Interest_Rate() as Double
Public Property Set Interest_Rate(Interest as double) as Double

b) Declaring methods and variables as private. Private indicates that the method is accessible only to other methods within the same class. This would be useful for internal variables and methods that are used to perform calculations or actions within the class itself, as an intermediate step to the end result shown by the interface.

The types of methods that can be used within the class include functions and subroutines.

Private Function FunctionName() as Object
Private Sub SubName()

Inheritance

Inheritance is the ability to create generic (base) classes and more specific (inherited) classes that inherit the generic properties and methods. Inherited classes inherit the methods and properties of the base class, but can also have unique or particular features. For example, we could define a car class as the base class, and have specific types of cars as inherited classes, such as a Vauxhall Corsa.

Inheritance is implemented in a simple way in Excel VBA. In the inherited class; the following text is placed at the top of the class module:

Implements BaseClass

Next Steps

In order to make practical use of the object orientated features, we need a reliable methodology for mapping concepts such as those in the Monte Carlo simulation, to the Class Modules in Excel VBA. The next document outlines a methodology for this.

How VBA Developer.net Can Save You Time and Money

You can get complete Excel apps from VBA Developer.net containing the code in this document, customisation, VBA development of any Excel, Access and Outlook apps, as well as C# and C++ add-ins and technical documentation.

Visit VBA Developer.net

Examples of VBA documents from VBA Developer.net

How to build a Black Scholes VBA Option Pricer

How to build a Black Scholes C# Option Pricer

How to build a Black Scholes VBA Option Pricer for FX Options

How to build a Black Scholes VBA Option Pricer for Equity Options

How to build a Black Scholes VBA Option Pricer using Monte Carlo Simulation

How to build a Black Scholes VBA Option Pricer for Binary Options

How to build a Black Scholes VBA Option Pricer for Equity Barrier Options

How to build a Black Scholes VBA Option Pricer for Exotic Asian Options

How to build a Black Scholes VBA Option Pricer for Exotic Lookback Options

How to build an Equity Option Pricer using the Binomial Tree in Excel VBA

How to code a Choleskey Decomposition in VBA (Numerical Methods for Excel)

3 ways to sort in VBA

How to Code a Multivariate Value at Risk (VaR) VBA Monte Carlo Simulation

How To Code the Newton-Raphson Method in Excel VBA

How to Model Volatility Smiles, Volatility Term Structure and the Volatility Surface in Excel VBA

How To Write Use Cases for a Portfolio Reporting VBA Tool

How To Write a User Interface Model For a Portfolio Reporting VBA Tool

How To Create a Semantic Object Model For a Portfolio Reporting VBA Tool

How To Normalise a Database For VBA Apps

How To Create a Database using SQL Scripts for a Portfolio Reporting VBA App

How to Write Stored Procedures in SQL/Access/VBA for a Portfolio Reporting VBA App

How to Use Cursors in SQL for a Portfolio Reporting VBA Tool

How to Move Data from Access to Excel with SQL for a Portfolio Reporting VBA App

Portfolio Reporting VBA Tool: Inserting Data into SQL/Access Databases from Excel

Portfolio Reporting VBA Tool: Connecting Excel with SQL & Access Databases

How To Design Classes For an Option Pricer in VBA: UML Concepts

How To Design Classes for Object Orientated VBA Programming