The European Journal of Education and Applied Psychology 2025, No 4
ISSN 2310-5704

PREMIER Publishing
ppublishing.org

DOI:10.29013/EJEAP-25-4-16-19

EXPLORING OBJECT ORIENTED PROGRAMMING AND MICROSOFT EXCEL'S OBJECT MODEL

Bashirova Goncha [1]

[1] Azerbaijan State Pedagogical University, Baku

Cite: *Bashirova G. (2025). Exploring Object Oriented Programming and Microsoft Excel's Object Model. European Journal of Education and Applied Psychology 2025, No 4.* https://doi.org/10.29013/EJEAP-25-4-16-19

Abstract

Relevance of the article. Currently, programming is widely used in all areas of human activity. This article is dedicated to teaching how to create effective software systems using the full capabilities of Algorithmic Language. Scientific novelty of the article. It explains the principles of creating effective software systems by fully using the capabilities provided by the Algorithmic Language. Practical significance and application of the article. Informatics teachers and people conducting research in this field can use the materials presented in the article.

Keywords: *Encapsulation, Inheritance, Polymorphism, Borders, Border objects*

In this article, we will study the object-type components of Microsoft Excel Application Software System, and gain the necessary knowledge to create a programming interface between VBA and Microsoft Excel. First of all, it should be noted that the Object Model of Microsoft Excel is slightly different from the standard Ob UserForm class ject oriented programming meanings considered in modern programming, and the programmer must take these differences into account when implementing Application Software System projects in the Excel&VBA environment.

First, let's get acquainted with the class and its derived object meanings.

1) A class is a type belonging to a certain structure. Unlike other types (intrinsic and user), the components of data/objects declared or created with it can be primitive type data/properties and function/method. One-to-one reference to these components is impossible, and all references are possible through the identifier of the object(s) derived from the class.

2) The given object declared and initialized through the class is called.

Modern algorithmic languages Object Oriented Programming have differences in the implementation of Object oriented programming. Algorithmic languages like Java, C++, C# are considered Standard Object oriented programming. Standard Object Oriented Programming has 3 main meanings:

Encapsulation
Inheritance
Polymorphism

Encapsulation

This meaning refers to the set of features and functions of the object. Vehicle features

(seat, seat color, front and rear windows, doors and door glass, etc.) and functions (engine start, wheel rotation, windows down/up, etc.) are encapsulated (i.e. bundled together) into a vehicle object. In this case, any object component (be it a property or a module) is referred to as a component of the object rather than a single object.

| | ekt.komponent[(parametr1,...)]

where, a component can be a property or a method. If a component is a method/function, then it can have certain parameters/arguments.

For example, several UserForm objects can be created from the UserForm class, one of VBA's control components. Even if these objects derive from the same class and have properties and methods with the same name, they are distinct and can be referenced/referenced only from the object itself/identifier:

| | User Form 1. Back Color = RGB(255, 0, 0)' UserForm1 change the BackColor property of the object

| | User Form1. Show ' UserForm1 call the object's Show method

where the **User Form 1** object is derived from the **UserForm class**

Inheritance

New car objects (eg Mercedes, Nissan, Volvo, etc.) can be created by adding new properties and functions to an abstract car (4-wheeled, moving, steering, body, etc.) object. Newly created objects derive from the abstract car object and inherit this viewpoint. This meaning is not implemented in VBA. So, VBA does not derive a new class from any class. That is, the components of any class cannot be imported into another class. Therefore, if the presence of any component is necessary, then this component must be physically added to the new class. The following construct implemented in AD such as Java, C++, C#, PERL, PHP cannot be implemented in VBA:

| | class B extends A {...}

where class B derives from A and retains the components of class A

Polymorphism

Objects can perform essentially the same functions. For example, although the function of opening doors in a car is essentially the same, their realizations may differ (right, up, left, sliding, etc.). This is called polymorphism.

A collection is an object that is a collection of objects of the same type. There is too much information about all collections/objects and their components (properties and methods). For this reason, let's explore one of the most commonly used objects, CellFormat.

The CellFormat object belongs to the Application object and holds the following collections and objects as properties:

- **Borders (collection). Border objects belong to this collection.**
- **Font (object)**
- **Interior (object)**

CellFormat.e_CellFormat_Borders_Properties (initial code)

```
| | 01 Sub e_CellFormat_Borders_Properties()
| | 02 Dim oB
| | 03 Set oB = Worksheets(«S1»).Range(«C5: F15»).Borders
| | 04 oB.LineStyle = xlDouble 'const. xlContinuous, xlDash, xlDashDot, xlDashDotDot, xlDot, xlDouble, xlSlantDashDot, xlLineStyleNone
| | 05 oB.Value = xlDashDot
| | 06 oB.Weight = xlMedium 'const. xlHairline, xlThin, xlMedium, xlThick
| | 07 oB.Color = RGB(0, 0, 255)
| | 08 oB.ColorIndex = 35
| | 09 oB.Item(xlEdgeBottom).Color = RGB(255, 0, 0)
| | 10 Debug.Print oB.Application 'res. Microsoft Excel
| | 11 Debug.Print oB.Count 'res. 6
| | 12 Debug.Print oB.Creator 'res. 1480803660
| | 13 oB.Parent.Name = «x» '??????
| | 14 Debug.Print oB.Parent.Name 'res. 'S1'!$C$5:$F$15
| | 15 Debug.Print oB.Parent.Row 'res. 5
| | 16 Debug.Print oB.Parent.Rows.Count 'res. 11
| | 17 Debug.Print oB.Parent.Column 'res. 3
| | 18 Debug.Print oB.Parent.Columns.Count 'res. 4
| | 19 End Sub
```

CellFormat.e_CellFormat_Borders_Properties (comment)

- **Line 02**. We declare a variable called oB;
- **Line 03**. We assign an expression composed of objects and collections to the variable oB. Then oB becomes an object that defines the borders (via the Borders collection) of cells C5: F15 of page S1 and has all the properties of the Borders collection;
- **Line 04**. We assign the xlDouble constant to the LineStyle property of the oB object. This means delimiting the actual displayed cells with a double line. Instead of the xlDouble constant, we can draw the boundaries in a different way, using the other 7 constants;
- **Line 05.** We assign the xlDashDot constant to the Value property, which is an alternative to the LineStyle property of the oB object. This means that the actual displayed cells are bordered by a dotted line;
- **Line 06.** We assign the xlMedium constant to the Weight property of the oB object. This means determining the thickness of the boundary lines. It is necessary to use one of the 4 constants that determine the thickness of the line;
- **Lines 07–09.** We define the color of the oB object using one of the alternative properties Color, ColorIndex, Item(xxx).Color;
- **Line 10**. The Application object is used as a property to indicate which system the oB object belongs to. This feature is especially important when working with variables derived from objects belonging to different systems. In line 10, we specify that the object oB belongs to the Microsoft Excel system;
- **Line 11**. It's hard to understand what the Count property on the oB object does. The information in the help system is not enough to clarify this. Also, the Border object, which is the only object in the Borders collection, does not have this property. The Count property on lines 16 and 18 specifies the number of rows and columns, respectively. This naturally makes sense. But the fact that the result of the 11th line has 6 numbers remains unknown. In my opinion, this indicates the possible number of lines used to delimit the cell: 4 sides and 2 diagonal lines. In all cases, this should be considered an error by the programmers who created VBA. Information about what this feature serves in a specific case (ie, Borders collection) should be reflected in the appropriate Help system article;
- **Line 12**. The Creator property of the oB object returns a specific number (1480803660). This property is intended for use on the Macintosh operating system and specifies that the oB object belongs to the Microsoft Excel system;
- **Line 13**. The Name of the Parent property of the oB object is initialized. Without it, lines 14–18 are not executed. This feature contains a bug (BUG). So if oB.Parent.Name = «x» is defined on line 13, then on line 14 the oB.Parent.Name property returns ='S1'!$C$5:$F$15!!!
- **Line 14**. The Name of the Parent property of an oB object returns the name of the object;
- **Line 15.** The Row of the Parent property of the oB object returns the row number (5) of the first row of the object;
- **Line 16**. Rows.Count of the Parent property of the oB object returns the number of rows (11) of the object;
- **Line 17.** The Column of the Parent property of the oB object returns the sequence number (3) of the object's first column;
- **Line 18.** The Columns.Count property of the oB object's Parent returns the object's number of columns (4).

References:

Partyka T. L., Popov I. I., Golitsyna O. L. Programming Languages. – Moscow: Forum Infra, 2008. – 400 p.
Sommerville I. Software Engineering. DialecticaWilliams Publishing Group, 2002. – 624 p.
Graham I. Object-Oriented Methods: Principles and Practice. 3rd ed. Moscow: Williams, 2004. – 880 p.
Pyshkin E. V. Basic Concepts and Mechanisms of Object-Oriented Programming. BHV-Saint Petersburg, 2005. – 640 p.
Ivanova G. S., Nichushkina T. N., Pugachev E. K. Object-Oriented Programming. Bauman Moscow State Technical University, Moscow N. E. Bauman, 2007. – 316 p.
Ken Getz, Mike Gilbert. Programming in Microsoft Office: The Complete Guide to VBA.

submitted 06.11.2025;
accepted for publication 20.11.2025;
published 30.12.2025

Contact: qoncabashirova@yahoo.com