Object Oriented Programming

- Everything we know in the Excel environement can be described in terms of VBA **objects**.
- Thinking of VBA structures in terms of objects, is a way of introducing a "superstructure" around which the whole programmation language is organized.
- Objects are the fundamental building blocks of VBA.
- An **object** is a special type of variable that contains both data and codes.
- Objects are often grouped in collections. A **collection** is a group of objects of the same **class**.
- The most used Excel objects in VBA programming are **Workbook**, **Worksheet**, **Sheet**, and **Range**.
- In the past weeks we have been using the object **Range** very often in VBA programs

Objects can have names

syntax: object(“name”)

Expl.: Workbook (“Labsession5.xls”),
Worksheet(“Sums”), Range(“trigdata”),
Range(“A1:A25”), ActiveCell, ActiveSheet,....

objects can be used as object variables

Expl.: Dim WB as object
Set WB = Workbook (“Labsession5.xls”)

similar as the variables we already know, we can use WB instead of Workbook (“Labsession5.xls“)

objects are arranged in a strict hierachy

Excel application → workbook → worksheet → objectX → objectY → ...

- this hierachy has to be respected in the VBA syntax, e.g.

Workbook("book1.xls").Worksheet ("sheet1").Range("A1:A2")

💣 not: Worksheet ("sheet1"). Workbook("book1.xls")

- when referring to an object which is in an active workbook or sheet, you do not need to specify the entire hierachy

Expl.:

Range("A1:A2")

- when it is in a non-active workbook and worksheet, you need to refer to the entire hierachy as above.

the WITH ....END WITH short hand

- this is a useful command which allows to avoid long hierachies

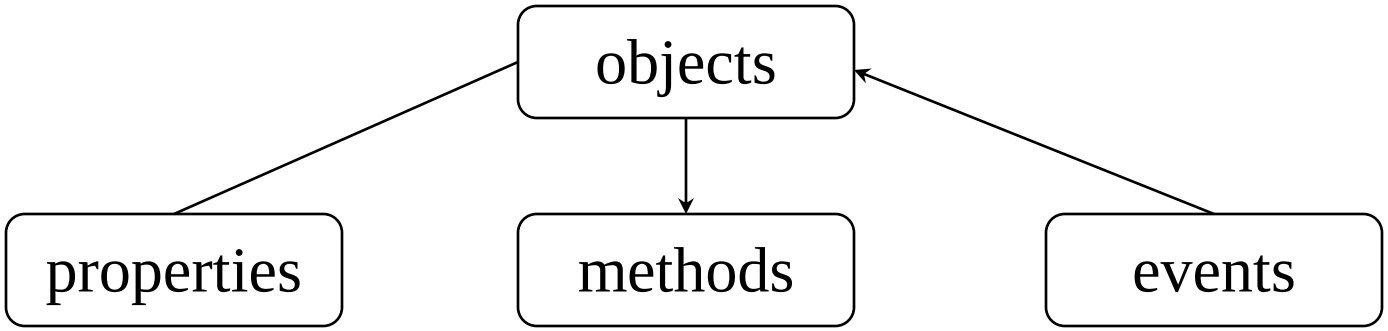
```
syntax: WITH objectX
                .objectY
                .objectZ
        END WITH
```

Expl.:

```
workbook("book1.xls").worksheet ("sheet1").Range("A1")
workbook("book1.xls").worksheet ("sheet1").Range("B25")
workbook("book1.xls").worksheet ("sheet1").Range("data")
```

```
instead: WITH workbook("book1.xls").worksheet ("sheet1")
                .Range("A1")
                .Range("B25")
                .Range("data")
         END WITH
```

objects posses properties, can carry out methods, react to events



the properties of objects are their characteristics

syntax: object.property = property value

Expl.:

Range("A1").**ColumnWidth** = 10

Range("A1"). **Font**.**Name**="Arial"

Name.**Value** = "This is Pi"

Chart("temp").**ChartType** = "xlLine"

Worksheets("Sheet1").**Columns**("A").**ColumnWidth** =10

the methods (functions) are actions the object can initiate

syntax: object.method [parameter := parameter value]

Expl.:

Worksheets("Sheet1").**Copy** After:=Worksheets("Sheet3")
(creates a copy of Sheet1 and places it just after Sheet3)

Range("A1").**Copy** Destination:=Worksheets("Sheet3").Range("B2")
(copies the content of cell A1 on the active worksheet to cell B2 in Sheet3)

Application.WorksheetFunction.**Cosh**(2)

Application.Worksheet Function.**Vlookup**(1,[a1:b2],3,False)

- objects can change their properties as a reaction to an event

syntax: object.event

Expl.:

Worksheet(“Sheet1”).**Calculate**

(the object worksheet named “Sheet1” is re-calculated and changes its properties)

➢Events may also be indicated at the beginning of a subroutine, as we have seen when we studied UserForms.

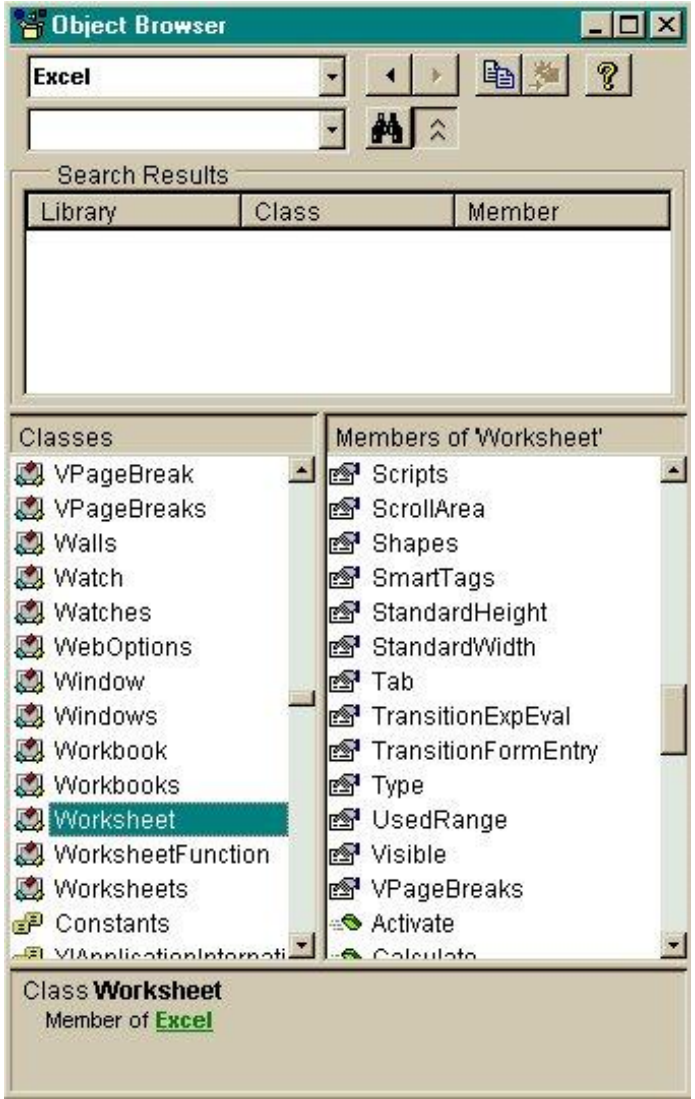
Expl.:

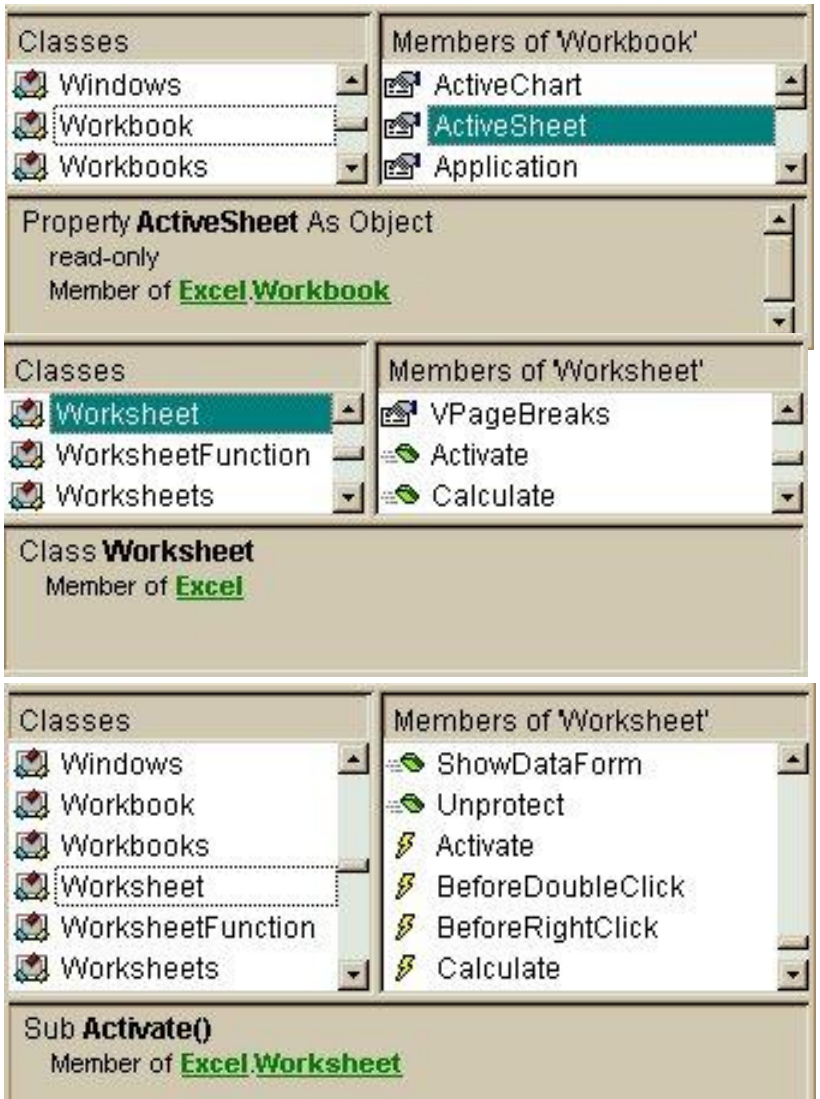
```
Private Sub Worksheet_Calculate()
Columns("A:F").AutoFit
End Sub
```

(This example adjusts the size of columns A through F whenever the worksheet is recalculated)

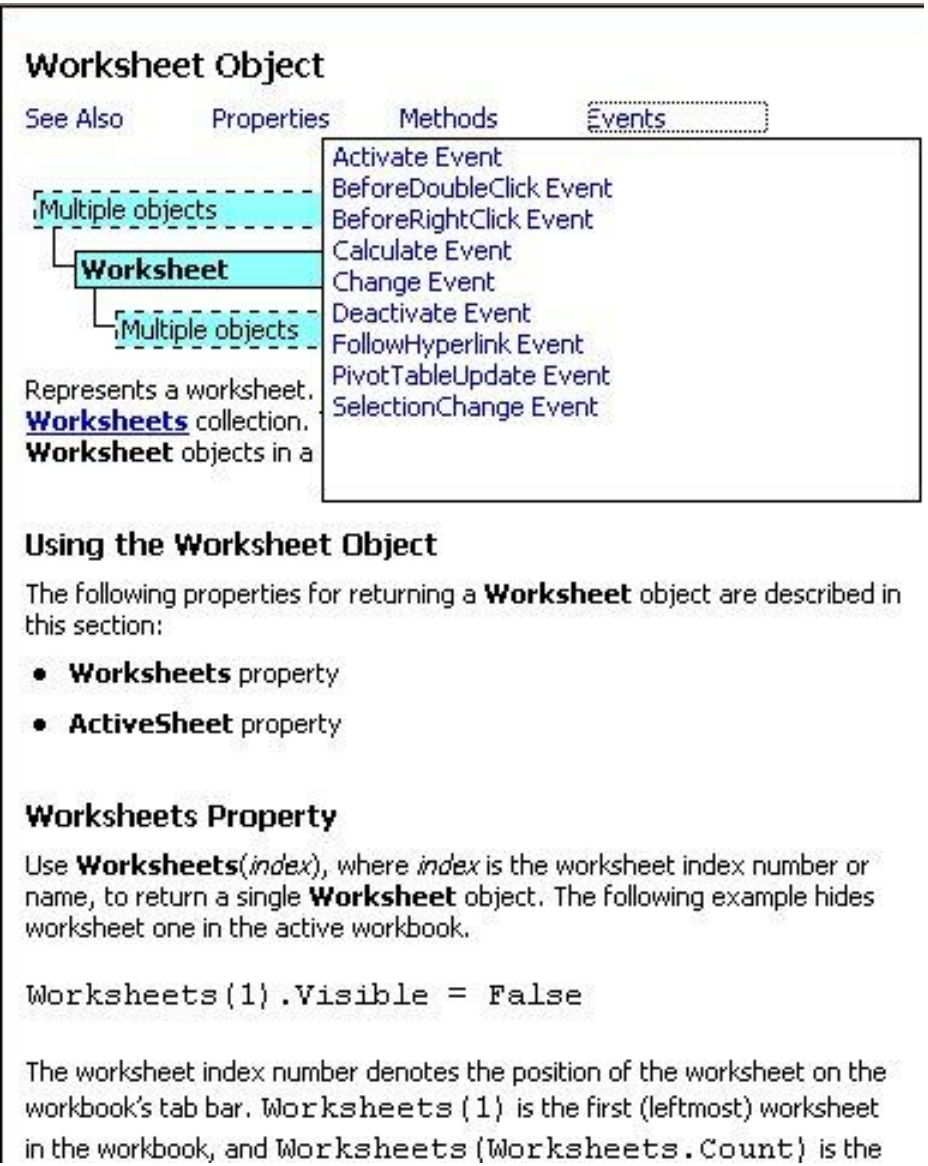
the object browser provides you with the details of the **properties, methods and events** associated to particular objects

- it is activated in the VBA editor
- view → object browser or with the function key F2





- clicking the question mark in the browser you can find out about the properties, methods and events related to an object:



Worksheet Object

See Also Properties Methods Events

Multiple objects
Worksheet
Multiple objects

Activate Event
BeforeDoubleClick Event
BeforeRightClick Event
Calculate Event
Change Event
Deactivate Event
FollowHyperlink Event
PivotTableUpdate Event
SelectionChange Event

Represents a worksheet. **Worksheets** collection. **Worksheet** objects in a

Using the Worksheet Object

The following properties for returning a **Worksheet** object are described in this section:

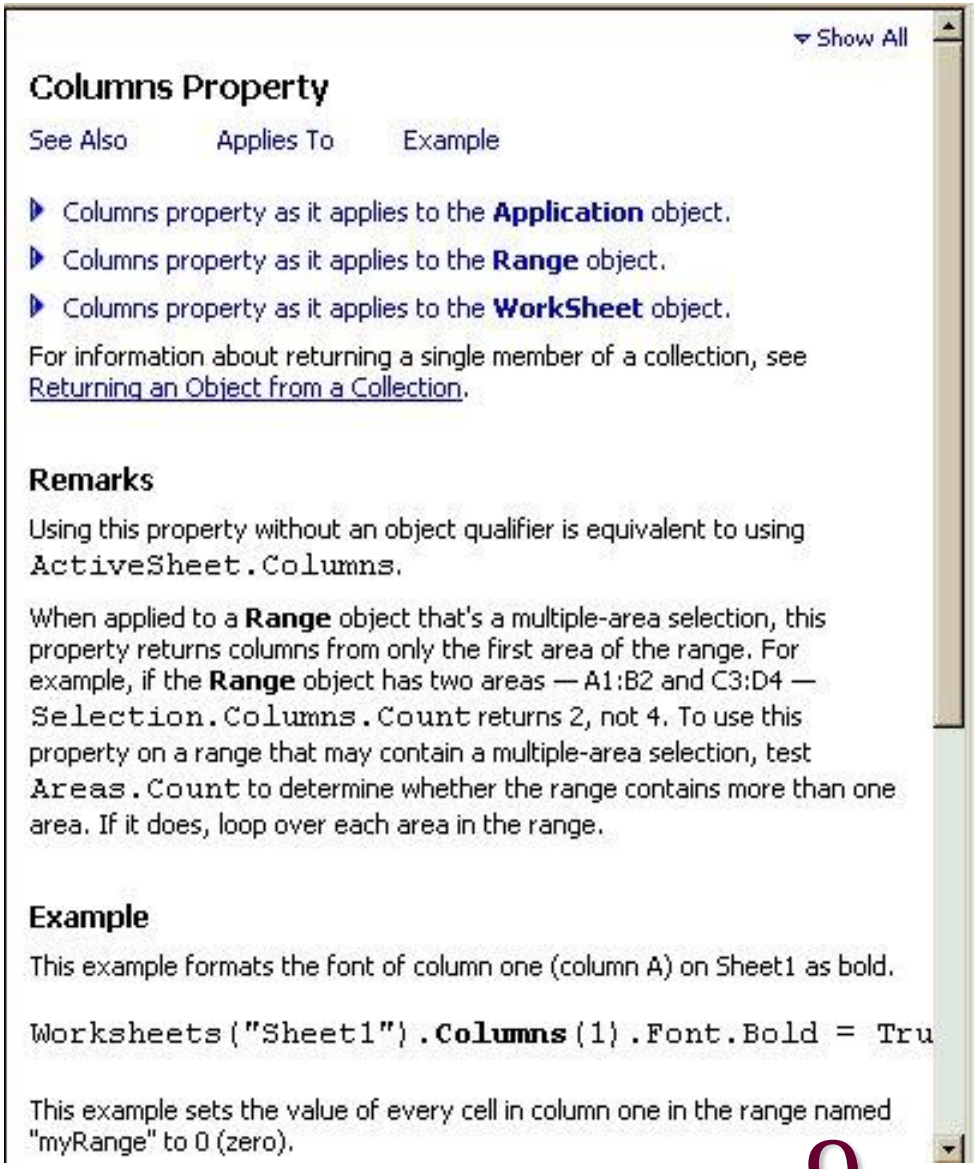
- **Worksheets** property
- **ActiveSheet** property

Worksheets Property

Use **Worksheets**(*index*), where *index* is the worksheet index number or name, to return a single **Worksheet** object. The following example hides worksheet one in the active workbook.

```
Worksheets(1).Visible = False
```

The worksheet index number denotes the position of the worksheet on the workbook's tab bar. `Worksheets(1)` is the first (leftmost) worksheet in the workbook, and `Worksheets(Worksheets.Count)` is the



Show All

Columns Property

See Also Applies To Example

- Columns property as it applies to the **Application** object.
- Columns property as it applies to the **Range** object.
- Columns property as it applies to the **WorkSheet** object.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Remarks

Using this property without an object qualifier is equivalent to using `ActiveSheet.Columns`.

When applied to a **Range** object that's a multiple-area selection, this property returns columns from only the first area of the range. For example, if the **Range** object has two areas — A1:B2 and C3:D4 — `Selection.Columns.Count` returns 2, not 4. To use this property on a range that may contain a multiple-area selection, test `Areas.Count` to determine whether the range contains more than one area. If it does, loop over each area in the range.

Example

This example formats the font of column one (column A) on Sheet1 as bold.

```
Worksheets("Sheet1").Columns(1).Font.Bold = Tru
```

This example sets the value of every cell in column one in the range named "myRange" to 0 (zero).

objects can be organized in collections

- members in same collection are on the same hierachical level
- you refer to a member of a collection just by a number

syntax: collection name(#)

Expl.:

worksheets(5) refers to the 5-th member in the **worksheet** collection

workbooks(3) refers to the 3-rd member in the **workbook** collection

names(6) refers to the 6-th member in the **name** collection

hyperlinks(1) refers to the 1-st member in the **hyperlink** collectio

- note: worksheets ≠ worksheet , names ≠ name, etc
- collections can be created by using the add-method

syntax:

collection name.add [parameter1:= parameter value 1] , [:=]

Examples:

- x = 3.141592653589793

  y = true

  z = “too many names“

  (the variables can be of different type)

  Names.Add Name:="pi", RefersTo:=x

  Names.Add Name:="correct", RefersTo:=y

  Names.Add Name:="message", RefersTo:=z

  - you can refer to a member of the names collection as:
  - Names(2) → true in the VBA code
  - correct → true on the Excel sheet

- WITH worksheets(1)

  .Hyperlinks.Add .Range(“B25”), http://www.city.ac.uk/

  END WITH

  Range(“B25”).Hyperlinks(1).Follow NewWindow:=True

  - inserts a hyperlink into cell B25 and executes it thereafter

Important announcement: Next week's lecture has been moved.

We will have a revision lecture on Monday the 28th between 12:00-13:00 in the Oakden Lecture Theatre.

Labs will go ahead as planned, also next week. The next Lab will be a revision Lab based on last year's test. This will be the last Lab.