COMP1022Q
Introduction to Computing with Excel VBA

Programming with Objects

Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Describe the basic concept of object-oriented programming
  2. Write simple object-oriented code in VBA

Overview

- We had a brief introduction to objects in the 'Using Excel Objects' notes
- Now we will look at them in more detail
- *Object-Oriented Programming* is an advanced topic in computer programming
- In this presentation, we will look at these:
  - Introduction to Object-Oriented Programming
  - What is a Class?
  - An Example Class – a Dog Class

Object-Oriented Programming

- We are dealing with ‘objects’ every day
- It would be great if we can ask a program to ‘think’ using objects too
- This way of programming, thinking using objects, is called *object-oriented programming*
- To do that we first design the objects
- Then we can make the objects interact with each other, if we want them to do that

Introduction to Objects

- There are many ‘objects’ around us in the real world, e.g. a dog and a car are both objects
- We can say that each object has two kinds of characteristics:
  - *attributes* and
  - *behaviours* - which we call *methods*
- For example, a dog has:
  - *attributes* such as name, colour and weight
  - *methods* such as eating, barking and running

Some Attributes of the Range Object 1/2

- We have used *Range* many times on the course
- Here are some examples of Range attributes we have met before:
  - `Value` – the value stored in a cell, e.g.

    ```
    Range("A1").Value = 1999
    ```

  - `Row` – the row number of a cell, e.g.

    ```
    MsgBox Range("A1").Row
    ```

Some attributes are read-only, such as this

Some Attributes of the Range Object 2/2

- Here are some examples of Range attributes we have not met before:
  - `Text` – the cell content as it is displayed in the cell, e.g.

    ```
    MsgBox Range("A1").Text
    ```

  - `HasFormula` – whether or not the cell has a formula, e.g.

    ```
    MsgBox Range("A1").HasFormula
    ```

It returns True or False

Some Methods of the Range Object

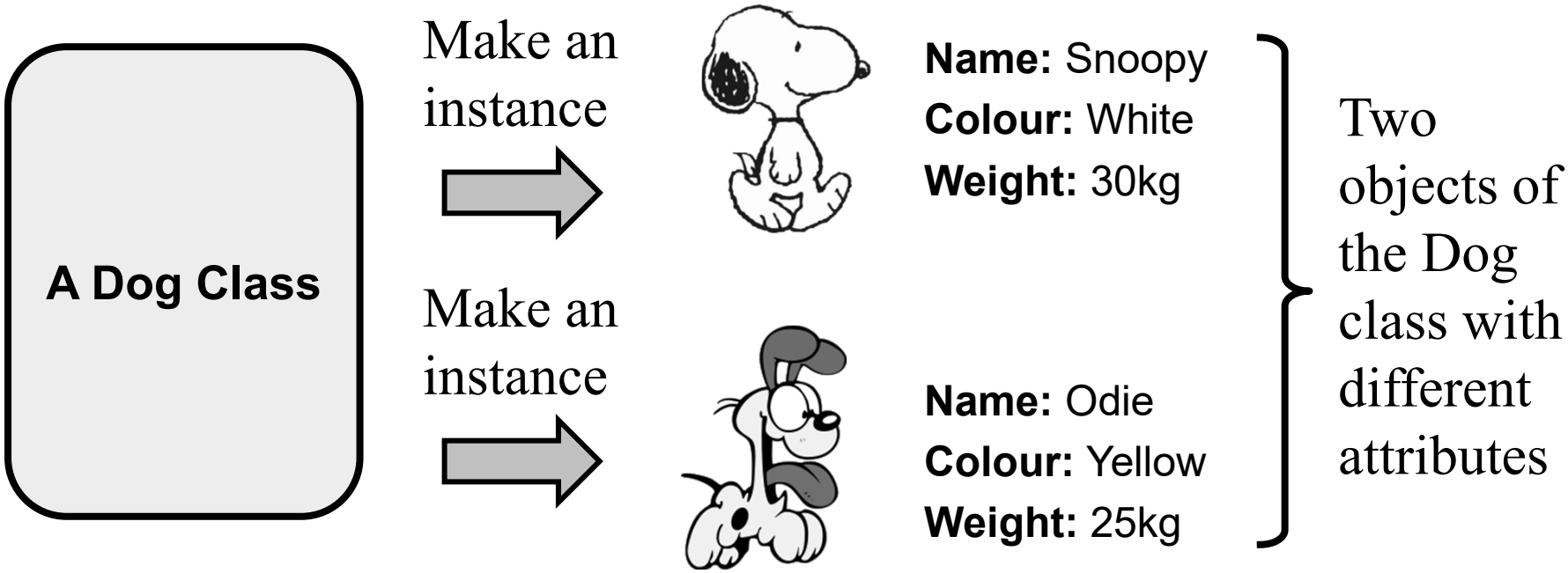
- Examples of Range methods we have met before:
  - `Clear` – clear the cells
  - `Select` – select the cells
- Examples of Range methods we haven't met before:
  - `AdvancedFilter` – filter the cells
  - `Find` – find something in the cells
  - `Sort` – sort the cells
  - `Speak` – speak the cell content
  - `Table` – convert the cells to an Excel table

What is a Class?

- In Computer Science we usually call the definition of an object a *Class*
- A class is only a definition, i.e. a class is the blueprint of an object you want to create
- When you want to create an object you need to make an *instance* of the class
- In a program you can create as many instances of the class as you want

An Example of Using a Class 1/2

- Let's say we have defined a Dog class
- In order to make Snoopy and Odie we need to create a Dog object, i.e. an instance of the Dog class, for each of them, like this:



An Example of Using a Class 2/2

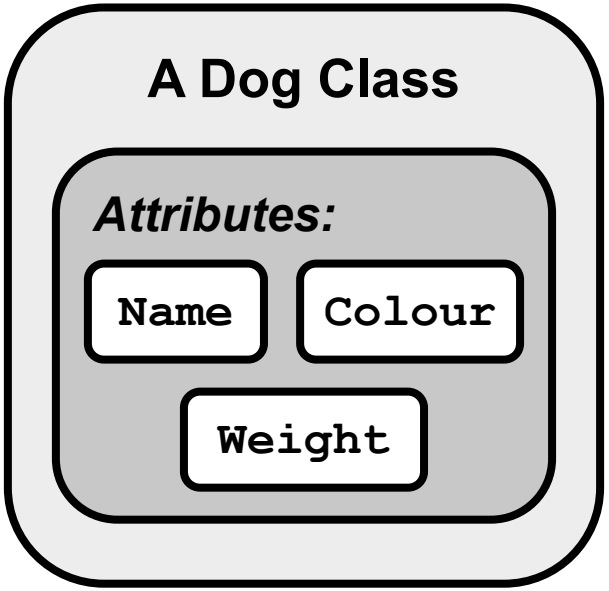
- Both the Snoopy object and the Odie object are created using the same class, the Dog class
- They are different to each other because they have different attribute values, such as their name, colour and weight

Name: Snoopy
Colour: White
Weight: 30kg

Name: Odie
Colour: Yellow
Weight: 25kg

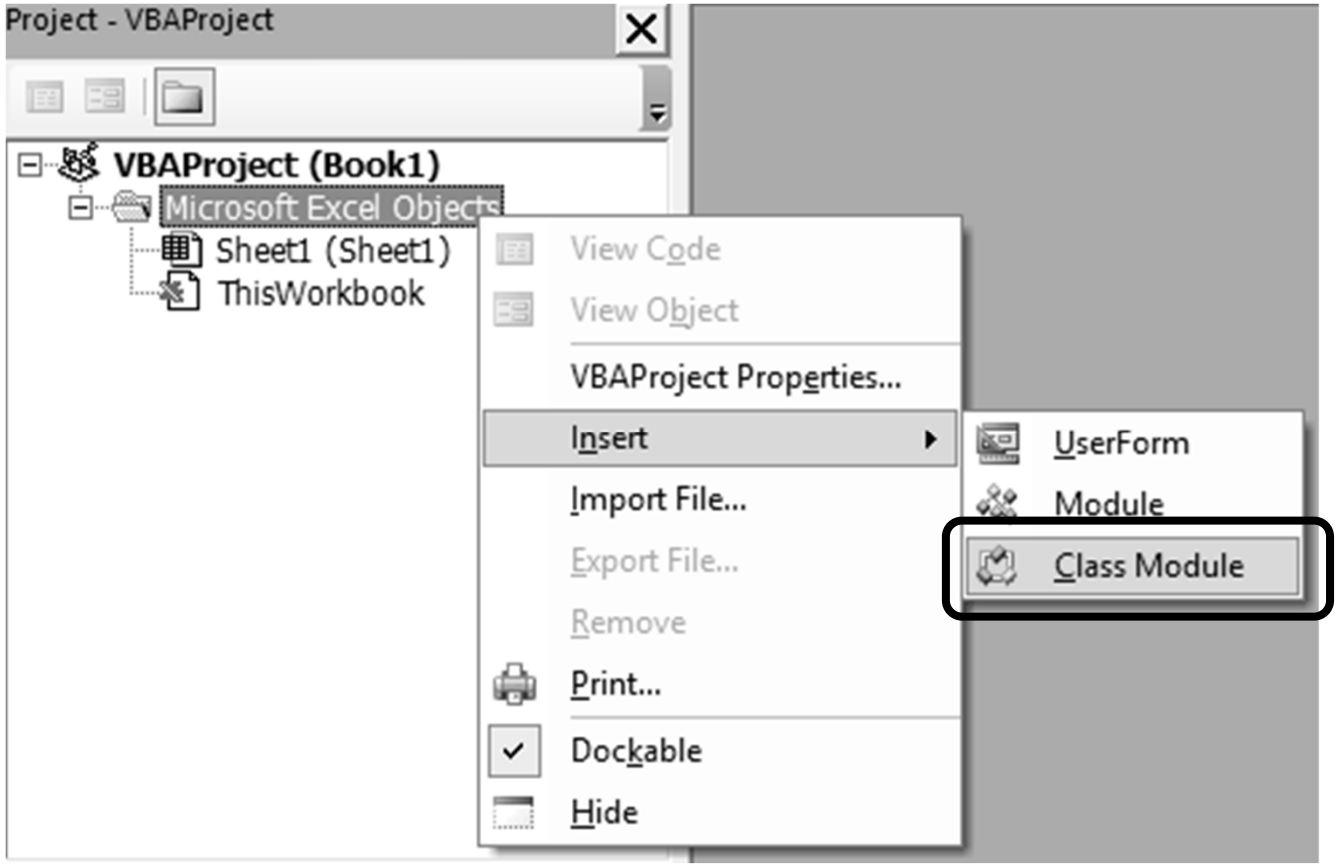
Defining the Dog Class in VBA

- Let's define the Dog class in VBA
- The Dog class has the following attributes:
  - `Name`
  - `Colour`
  - `Weight`



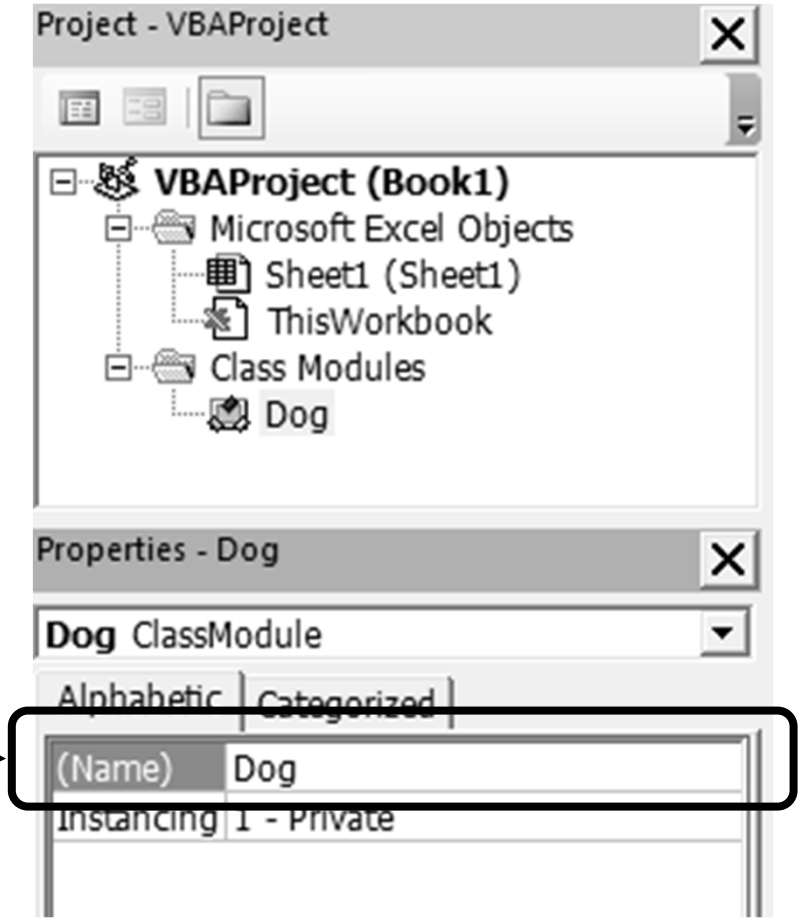
Creating a Class Module 1/2

- To create a class in VBA we need to first create a *Class Module* in the VBAProject, like this:



Creating a Class Module 2/2

- After we have the class module we need to change its name to the class name we want to use
- For example, here we change the name of the class module to 'Dog' because we are making a class called Dog



Making the Class Attributes

- To make class attributes, you put the attributes at the **top** of the class module like this:

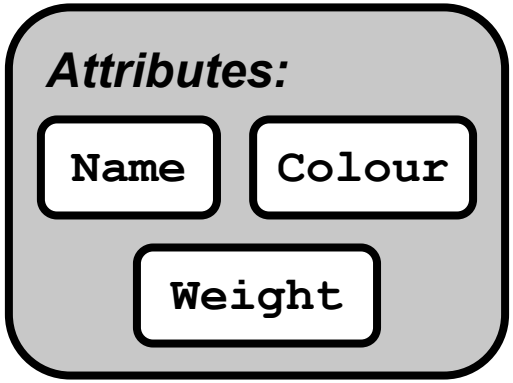
```
Public Name As String
```

Attribute name Data type

- Similar to a variable, a class attribute has an attribute name and a data type
- The word `Public` means any code in the same Excel file can read the content of the attributes

Attributes in the Dog Class

- The attributes in the Dog class can then be created using these lines of code at the top of the class module:




```
Public Name As String
Public Colour As Long
Public Weight As Double
```

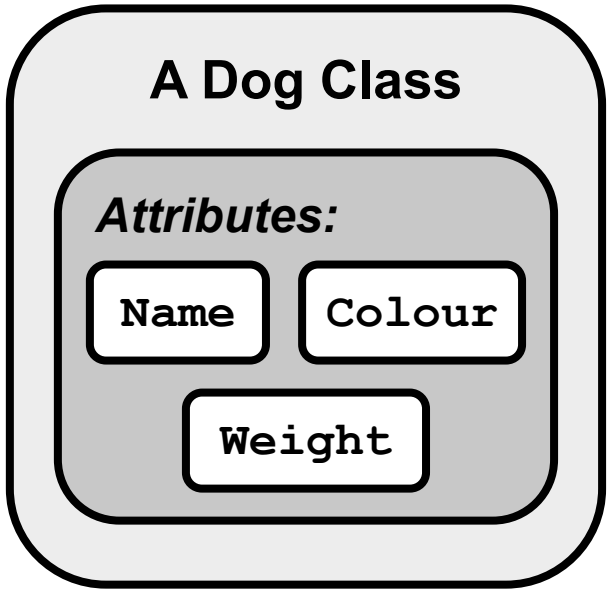
- In programming, it is very common to use a number to store a colour value
- In VBA, we use a `Long` to store a colour

The Dog Class

- Here is the code of the class module for the Dog class shown on the right:

```
' This is the Dog class

' Attributes of a dog
Public Name As String
Public Colour As Long
Public Weight As Double
```



Making a Dog from the Class

- After defining the Dog class, let's use the class to create a dog, Snoopy
- First, we need to use a variable to store the dog that we are going to create, for example:

```
Dim Snoopy As Dog
```

Variable name — 'Dog' is the data type, i.e. the Dog class

- We can then make a new dog from the class using the *New* keyword, like this:

```
Set Snoopy = New Dog
```

Create a new instance of the 'Dog' class

Setting Up the Attributes

- At this stage, the `Snoopy` variable stores a newly created Dog object
- However, the attributes of `Snoopy` are not set
- We need to give appropriate values to the attributes, like this:

```
Snoopy.Name = "Snoopy"
Snoopy.Colour = vbWhite
Snoopy.Weight = 30
```

Name: Snoopy
Colour: White
Weight: 30kg

As you know from using Range, you can read or write an attribute by using a '.' followed by the attribute name

Making Another Dog

- Similarly, we can create another Dog object, Odie, using the following code:

```
Dim Odie As Dog

Set Odie = New Dog
Odie.Name = "Odie"
Odie.Colour = vbYellow
Odie.Weight = 25
```

Name: Odie
Colour: Yellow
Weight: 25kg

Using Both Dog Objects

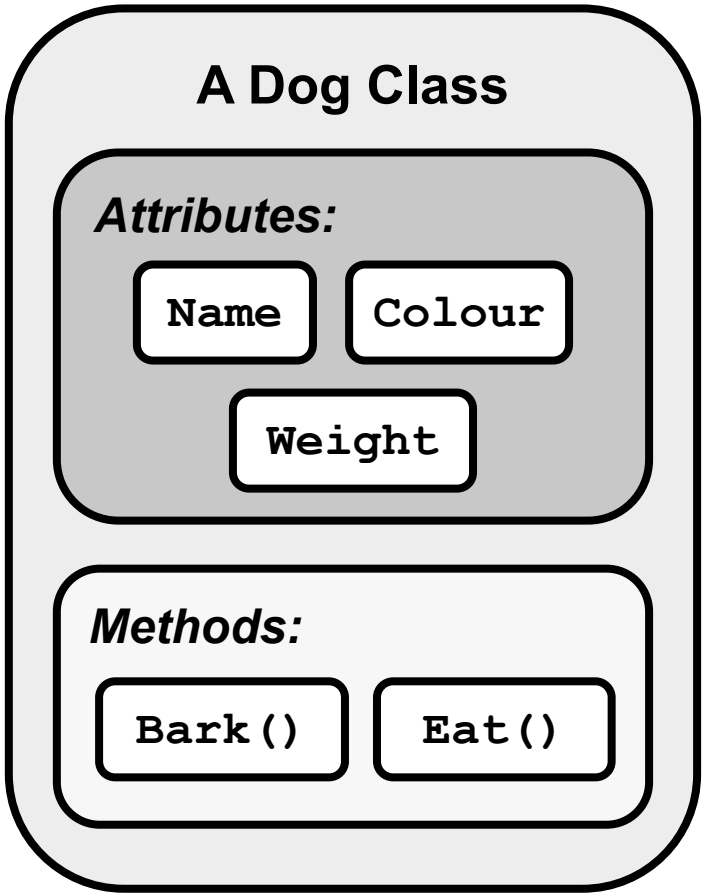
- We have two Dog objects now
- Let's show their names using the following code:

```
Msgbox "My first dog is called " & _
       Snoopy.Name & "." & vbNewLine & _
       "And my second dog is called " & _
       Odie.Name & "."
```

`vbNewLine` simply moves the text to the next line

Extending the Dog Class

- So far, the Dog class does not do anything
- Let's make it more interesting by adding two methods to the class:
  - `Bark()`
  - `Eat()`



Making a Class Method

- A class method is any function or subroutine created inside a class module
- For example, we can create a `Bark` method, which is a subroutine, in the class like this:

```
Sub Bark(ByVal Woof As String)
    MsgBox Woof,  , Name
End Sub
```

- You can create local variables as usual in a class method, or you can access a class attribute by simply referring to its name

Using the Class Method

```
Sub Bark(ByVal Woof As String)
    MsgBox Woof, _, Name
End Sub
```

We don't need the message box icon here so the parameter is omitted

- After making the method in the class, we can use the method from one of the dog objects like this:

```
Snoopy.Bark "Hello!"
```

- A message box will be shown after running the above line of code

Creating an Eat Method

- Let's add another method, `Eat`, to the Dog class
- The idea of the `Eat` method is:
  1. The dog's weight attribute increases after eating
  2. The dog barks when the dog is full

```
Sub Eat()
    Weight = Weight + 1

    If Weight >= 35 Then
        Bark "Oh dear! I am full!"
    End If
End Sub
```

Here the `Eat` method uses another method, `Bark`, from within the same class

Using the Eat Method 1/2

- Using the `Eat` method we can use a loop to keep on feeding the dogs

```
Dim DogToFeed As String
Do
    DogToFeed = InputBox("Feed which dog?")

    If DogToFeed = "Snoopy" Then
        Snoopy.Eat
    ElseIf DogToFeed = "Odie" Then
        Odie.Eat
    End If
Loop Until DogToFeed = ""
```

Feed the dog whose name was typed in

Stop feeding the dogs if the user didn't type anything

Using the Eat Method 2/2

- Here is an example of feeding Snoopy five times: