Introduction

Objectives

- To understand the concept of a "control object" as it pertains to windows and programming in Visual Basic.
- To learn how to launch the Visual Studio Integrated Development Environment and create a new project.
- To familiarize yourself with the Label, TextBox, Button, and Form control objects and the properties associated with them.
- To understand the importance of preparing a "requirements definition" for gathering information that will be needed in a project and how to create one.
- To be able to design simple windows interfaces using the basic control objects.
Event-driven and OOP

Visual Basic has been described as an "Event-driven" and an OOP ("Object Oriented" Programming) language. What do these terms mean?

An "event-driven" programming language is one where programs are written to respond to actions taken in a windows environment. These actions, called "events," include all the typical things that a user can do in a window with a keyboard and mouse, including:

- Click on a mouse button.
- Click on a virtual button in a window.
- Scroll horizontally or vertically the text in a window.
- Enter data through an input field.
- Display text or images in a window.
- Click on radio-buttons or check boxes.

An "object-oriented" programming language is one where programs are written to create and manipulate virtual representations of real-world collections of objects. The objects that we may wish to represent (or "model") may be as complex as a person or as simple as a check-box in a window. How these objects are to be represented in a computer program must usually be defined by a programmer such as you. You must decide on what characteristics or properties are necessary to describe sufficiently any member belonging to a particular collection of objects so that it can be modeled satisfactorily in a computer program. In "programmer-speak," the members are called "instances" and the collection is called a "class," and so you will find programmers talking about "instances of a class" rather than members of a collection.

Understanding the concepts of "event" and "object" are central to writing programs in Visual Basic, so you will find much of Study Guide and the textbook devoted to how to write programs that respond to events and how to manipulate objects.

When providing Visual Basic programs and source code as examples in this Study Guide, the following conventions will be adopted:

1. This Study Guide will use Visual Basic class names, such as "Button" and "TextBox" when referring to objects such as "button" and "text box."
2. Two important object classes in Visual Basic are "Textboxes" and "ListBoxes." The textbook suggests using a ListBox for displaying output. For now this Study Guide will use the TextBox for this purpose. We will introduce the ListBox later. If you are unfamiliar with Windows, you should review the relevant sections in Appendix B of the textbook. Notepad may also be useful to you in creating test files for later assignments in the course.
3. In this Study Guide, words and phrases in italics are to be replaced by Visual Basic identifiers and values to create instances of the description provided in italics.
Control Objects

To keep things simple, the examples in this Study Guide as well as your programs will initially employ only five classes of objects:

1. **Label objects** provide you with a way to place titles in your window. The actual title displayed is a property of the Label object and is provided as part of the programming.
2. **TextBox objects** allow you or another user to enter data into a program or to display output so that the user can observe the result.
3. **Button objects** provide a virtual button for you or another user to click on in order to initiate some action. The effect of the action is usually described by a caption on the button. The text of this caption is a property of the Button object and is provided by the programmer in the development of the application.
4. **ListBox objects** allow you to display a list of values as output.
5. **Form objects** are used to represent the entire input/output interface window. Most applications require one or more form objects, depending on whether a single window is needed to collect all the input and display all the output. A combination of objects such as Labels, TextBoxes, and Buttons is placed on each form object.

These are the components of the form that you (the programmer) deem necessary to convey what to do to the user and provide a means for the user to supply any required input and observe any resulting output.

Because instances of all of these types of objects except Forms provide the user with the ability to affect the behaviour of a program, Visual Basic programmers refer to these objects as "control objects" or simply "controls." The "Toolbox" window of the IDE provides these and many more control objects for you to use as a programmer in implementing your Visual Basic programs. To display the toolbox, check the tab labelled "toolbox" on the upper left side of the window.

To be sure you can get started and to familiarize yourself with these controls, you should perform the walk-throughs provided in Section 2.2 of your textbook. Be sure you actually do them rather than just reading that section of the text. In general, you should make it a practice to enter and run most, if not all, programming examples that this Study Guide suggests you examine. You cannot learn how to program without actually "doing it." Like any natural language where you must speak and write in the language in order to learn it, so it is the case that you cannot learn a programming language without actually writing programs and running them. Also, just writing the assignment programs is certainly not enough practice!
Interface Design

An important task in writing most programs is to define a suitable input-output interface. It provides the means for users of your program to communicate with it in a meaningful way. Hence, we often refer to this interface as the "user interface" of our program. Even though you may not have written a program previously, if you have used a computer you will be familiar with dealing with user interfaces since that is the purpose of a window. You can probably recall some examples of programs with windows interfaces (also called "graphical user interfaces" or "GUIs") that were easy to use and some that were difficult. Finding the right combination of boxes, labels, text, pictures - all typical components of a window - is central to the design of a good (that is, "user-friendly") user interface.

In Visual Basic and many other languages, each of the components of a GUI is implemented as an instance of some class of objects. That is, each type of component can be described by a number of properties. For example, a virtual button (that is, something in a window you can click on to make something happen) might have properties that relate to its appearance, such as its size, colour, and the text of the label appearing on it.

In Visual Basic, the types of components you might wish to include in a window are represented as classes of objects. Thus there are the Button class for button objects and the TextBox class for text fields.

To aid programmers in writing their programs, suppliers of programming software often provide a window-based program called an "Integrated Development Environment" or IDE. This program provides a number of tools that allow programmers not only to write and edit programs but also to create the user interfaces for interacting with the program. As well, other tools are provided to save programs and user interfaces and otherwise manage the development of a typical programming project.

For VB, Microsoft provides such an integrated environment, called "Microsoft Visual Studio," or more simply "VS IDE." To launch the VS IDE for Visual Basic:

1. Click on start (lower left-hand corner).
2. Click on "All Programs."
3. Click on the "Microsoft Visual Studio 2012" folder.
4. Click on "Visual Studio 2012" to launch.

A window will appear, called the Visual Studio IDE Start Page. This is the starting point for developing new programs as well as editing existing ones. This window is not exclusively for the development of VB programs. Instead, you can choose the desired programming language from a list of choices. For this course, when you start a new project, you should always select "New Project," and since you will always be constructing programs that allow the user to interact with the program via a window, you should choose "Windows Forms Application" as your template.
Input/Output Requirements

When you create a new program, called a "project" in Visual Basic, the IDE window displays a Form object as a smaller window with the default name "Form1" displayed in the smaller window's header. This smaller window is an "instance" of the "Form" class of objects. Each time you create a new project, another instance of the "Form class" is created, which is also called "Form1." You can also create several Form objects to be used in just one project. You will need to make several if you wish to have users interact with your program using more than one screen. How to do make these will be described later.

Initially "Form1" represents a blank window, available for you to design so that it includes all the input and output features that you will need for your program. Therefore, before you can design a suitable window for your program, you need to know what your program will require to perform its task and what output you wish to have your program display. The task of determining this information is part of an activity called "requirements definition."

When a programmer develops a requirements definition, he or she tries to provide as complete a description of what the program should do as possible. In determining the program's behaviour, the programmer will usually be able to identify what the program needs to perform its task (i.e., the inputs) and what the users expect to see from the program (i.e., the outputs). Therefore, you should begin every programming assignment by analyzing the text of the original problem description in order to identify the requirements.

The results of your findings can be tabulated in a list. In particular, for each of the inputs and outputs you identify, you can design its representation as the component of a window for your windows form. The collection of proposed component objects can then be positioned on the form in order to find a satisfactory layout. For example, try moving the objects around the form that you created when you did the walkthrough in the textbook. This capability allows you to design a user interface that is pleasing to look at as well as easy to use - important considerations if you want to make your program "user-friendly."

To illustrate a simple requirements analysis, suppose you wish to design a "loan calculator" program. From this rather brief description of the problem, you may infer that you will need to know the principal, the interest rate, and the period of the loan. It may also be reasonable to conclude that the purpose of the program is to determine the monthly payment, and therefore you conclude that the value of the payment will be the output. From this analysis, the following requirements definition can be tabulated:
INPUT REQUIREMENTS

<table>
<thead>
<tr>
<th>INPUT</th>
<th>TYPE</th>
<th>PURPOSE</th>
<th>REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principal</td>
<td>numeric</td>
<td>amount of loan required</td>
<td>TextBox</td>
</tr>
<tr>
<td>Interest Rate</td>
<td>numeric</td>
<td>annual lending rate</td>
<td>TextBox</td>
</tr>
<tr>
<td>Period</td>
<td>numeric</td>
<td>lending period (years)</td>
<td>TextBox</td>
</tr>
</tbody>
</table>

OUTPUT REQUIREMENTS

<table>
<thead>
<tr>
<th>OUTPUT</th>
<th>TYPE</th>
<th>PURPOSE</th>
<th>REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payment</td>
<td>numeric</td>
<td>monthly payment</td>
<td>TextBox</td>
</tr>
</tbody>
</table>

It is immediately apparent that we will need three TextBoxes to permit the entry of the necessary data as well and one TextBox to display the result. Of course, it will not be apparent what the purpose of each TextBox is unless we provide a description beside each TextBox. "Label" objects are useful for this purpose, so we should provide three labels for the inputs and possibly one more to enable the user to identify and interpret the meaning of the output. We can also use a label object to place a title on our Form.

We are still not done with this analysis, however. The TextBoxes only allow us to enter or display data. We need to be able to tell the program to calculate the result once we have entered the necessary data. We can use a "Button" object for this purpose. The "event" of clicking on the Button will be programmed to cause the application to perform the required calculation when the user is satisfied with the values that have been entered in the TextBoxes.

Event Requirements

As well, the user may wish to do a series of calculations, perhaps to compare different rates or periods. An additional requirement then is to "erase" a set of results by blanking out the previous contents of the TextBoxes so that the user can enter new values. The capability for providing this type of "event" can also be addressed with a Button. And, of course, another obvious event that is required if we allow the user to compute different results is the ability to quit the application. This event too can be implemented with a Button. All of these requirements can also be tabulated:

EVENT REQUIREMENTS

<table>
<thead>
<tr>
<th>EVENT</th>
<th>EFFECT</th>
<th>REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calc</td>
<td>Determine Payment</td>
<td>Button</td>
</tr>
<tr>
<td>Clear</td>
<td>Clear all boxes</td>
<td>Button</td>
</tr>
<tr>
<td>Quit</td>
<td>Stop the program</td>
<td>Button</td>
</tr>
</tbody>
</table>
At this point, if we decide that we have addressed all the input/output needs of the application, we can then prepare a layout of where to position the eleven objects (4 Textboxes, 4 Labels, 3 Buttons) we have identified in our analysis. Note that while the requirements analysis identifies what the program is to do with the inputs - that is, compute a monthly payment - we have not yet specified how that computation will be performed: that is, the algorithm that will be used. This fact illustrates an important aspect of the initial problem analysis. Its purpose is to determine what is to be computed but not how to achieve that computation. That task is carried out in a subsequent step of programming called "algorithm design," which will be described presently. However, even though we may not have specified our algorithm, we can begin implementing our solution by developing the interface from the requirements analysis that has already been performed. The following is a proposed interface-window to accommodate the identified requirements:

![Proposed interface-window](image)

Note that one additional Label has been added: the title "PAYMENT CALCULATOR."

**Properties**

All control objects on a Form as well as the Form itself are described by a set of properties. All instances of the same type of object have the same set of properties. However different values can be assigned to the set of properties for each instance. This helps to characterize that object and distinguish it from other instances. The collection of values assigned to the properties of a particular object is called its "state.

In VB programming, it is usually necessary to set the state of an object to an initial value. When you first create an instance of a windows object, the VS IDE does just that; it assigns "default" values to the properties, and these are the values you observe when you look in the Properties Window of the object you have just created.

However, you may wish to have an initial value that is different from the default value. It is also sometimes necessary to change the state while the program is running. In either case, to do so requires that you change the value of one or more of the properties of the object.

When the objects whose state we wish to initialize are predefined windows objects (i.e., TextBox, Label, Button, or Form), there are two ways to initialize the state of the object:
1. By using the "Properties Window" when the Form is designed and an instance of an object is added to it;
2. By using an assignment statement as described above.

Using the Properties Window is the most convenient way to initialize properties because it produces a visible result that appears immediately on the object whose properties are being set during the construction of the Form. As mentioned before, some of the properties are automatically set to particular values as a consequence of placing an instance of the object class on the Form and sizing it. You can see this effect, for example, if you place a TextBox on your Form and examine some of the properties listed in the Properties Window for that TextBox:

1. Place a TextBox on a Form object in the VS IDE. It can be any size.
2. Open the "View" menu at the top and click on "Properties Window."
3. Scroll the Properties list until the attributes "Size" can be observed. Click on the small "triangle" to the left of the text "Max Ht."
4. Note the value of the width and height.
5. Using the sizing handle of the TextBox, change its width.
6. Note the width again. You should observe a change in value.

Using the Properties Window of the VS IDE, you can only initialize the properties of windows objects; that is, the value assigned to each property when the Form is designed will not be changed during the execution of your program. If, however, a property such as the colour of a button may need to be changed when the program is running, then a different method for assigning values to the properties is required.

Preparing Forms

In order to gain experience in preparing forms, you should perform the walk-throughs provided in Section 2.2 of the textbook if you have not yet done so. They will acquaint you with the VS IDE tools that create instances of Forms, TextBoxes, Labels, and Buttons as well as techniques for defining and revising their properties. Then, to give you more practice with the features of the VS IDE, take the time to try to construct the payment calculator form (Form1):
If you haven't performed all the steps in the walk-throughs, this example will give you an opportunity to identify some of the properties associated with each type of object that are listed in the Properties Window. You can display the Properties Window in the VS IDE window by following these steps:

1. Be sure you are displaying the design window, then select one of the windows objects in your Form by clicking on it once.
2. From the View menu, select "Properties window" or press function key F4.
3. All the properties and their current settings for the object you selected in (1) will be displayed in a scrollable window, located on the lower right side of the design window.
4. You can change the value of each property by editing the existing text beside each property in the Properties Window.
5. There are two ways to choose a different object in order to examine its properties:
   - Return to the Design window by clicking on the "tab" at the top of the window labelled "Form1.vb[design]." Then repeat these steps beginning at step 1.
   - Use the pull-down list of objects that is available underneath the Properties Window header. On the line that names the object whose properties are currently displayed, click on the small triangle located at the right-hand side. Then, select the name of the object whose properties you wish to examine.

For example, you can provide the text of the Label objects as well as that on the Button objects by modifying the "Text" property of these objects. Also, the Label, "PAYMENT CALCULATOR," is centred and a 12-point font is used while the other Labels as well as the text in the Textboxes are right justified. You will need to access the Properties window for each of these objects and change the appropriate property accordingly to obtain the same effect. Note too that the background of the output window has also been changed from the default "white" background to an "off-white" colour.

**Assignment Statements**

An assignment statement is one of the most common type of statement in many programming languages. As the name suggests, its role is to assign values to labels called variable names. When a variable name is defined, memory is allocated and associated with the variable name. When an assignment statement is executed, a value is stored in the memory associated with the variable name.

For example, the following Visual Basic statement is a simple assignment statement:

```vbnet
course = "CMPT 110"
```

The "=" symbol should not be read as "equals" however, because this statement is not to be interpreted as an equation. Rather the "=" symbol should be read as "is assigned the value ..." In the example above, the assignment statement should be read as 'The variable course is assigned the value "CMPT 110".' The label course is a variable name, "CMPT 110" (including the quotation marks), is a value called a string
and the statement indicates that the value "CMPT 110" is to be stored in the memory associated with the variable name course.

Rather than just a single value, variable names can have a number of values associated with the name. In Visual Basic, each time you select a type of control object from the toolbox, a unique name is created by the Visual Basic IDE for that object, called the "object name." For example, the first button you create is called "Button1", the second is called "Button2" and so on. These are examples of variable names and the memory is associated with each of these variable names is sufficient to store the values of all the properties of each instance of this type of object.

An "object name" should not be confused with the "Text" property associated with that object. The "Text" property is used to assign labels to an object that are displayed on the object within the Form. In the following example, "STOP" is not the name of the button object, it is the value of the "Text" property of the button object:

Although not apparent from the Form, the button object's name is "Button1".

The value of each property can be referenced by applying a suitable "property selector" to the variable name of the object. The syntax for applying a property selector to the name of an object is:

Object Name.Property Selector

Every class of objects has a set of property selectors, each defined by an identifier. For example, the property selector of the "Text" property of a Button object is called "Text". The following assignment statement assigns "STOP" to the Button object, Button1, as shown in Form1 above:

```
Button1.Text = "STOP"
```
"Text" is also the name of the property selector of the "Text" property of a TextBox object. This property is the value that is displayed within a TextBox on a Form. For example, TextBox1 can be set to "CMPT 110" with the Visual Basic assignment statement:

TextBox1.Text = "CMPT 110"

**Assigning Properties**

While it is convenient during the design of your windows interface to use the Properties Window of the VS IDE, remember that you can only initialize the properties of windows objects in this way. The use of the assignment statement is more general.

If you know the name of the object and the name of the property that you wish to initialize or change, then you can do so with an assignment statement having the following Form:

```
Object Name.Property Name = new value
```

Because assignment statements are placed within your program, they are performed when your program is running. This fact permits you not only to set properties to initial values but also to change them if and when circumstances dictate: that is, when a particular "event" occurs.

In the Payment Calculator, the contents of the TextBoxes were initialized by VB as "empty" TextBoxes, the default value of the "Text" property of these objects. However, this procedure can also be done as part of the program by including the following assignment statements in the program:

```
TextBox1.Text = ""
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""
```

These statements form part of the body of a program that reacts to events that will occur to the objects in the user interface. These events include clicking on the "Calc," "Clear," and "Quit" buttons as well as loading the user interface itself when the execution of the program begins.

In this application, it is actually necessary to set these properties with the assignment statements because the requirements analysis suggests that we may want to clear these fields more than once. If the user wishes to perform more than one calculation, then we will need to clear the input fields of each TextBox for each calculation by resetting the values of the Text property.
As you will see, changing the values of some of the properties of a windows object is a common thing to do as a result of an event occurring. How you write programs to handle events will be examined in more detail in the next unit.

Updated August, 2017