FIT 100

Lab 10: Creating the "What's Your Sign, Dude?" Application

Spring 2002

Reading for Lab 10:

- p. 97-105 in Chapter 5 of *Computer Programming Fundamentals with Applications in Visual Basic 6.0*
- p. 151 -154 in Chapter 7 of *Computer Programming Fundamentals with Applications in Visual Basic 6.0*

Introduction

Today you create a program and call it **SignFinder**. In this program, a user will click on the radio button for the month of their birth but they will enter the exact date of their birth as well. When the OK button is clicked, the program will have to decide which sign to display depending on where their birth date falls in the month.

Objectives

- To understand the importance of object names when changing properties and assigning values with code.
- To be precise in the declaration of variables and understanding how to assign values to them.
- To take input from an unknown user and assign that value to a variable and use it as part of the logic of the program.
- To steadily add pieces of code to your program and increase it's complexity.

1. Creating the Interface for SignFinder:

If you have created a folder holding the form and all required controls for Lab 10 already, download it from Dante to the desktop. Then go to that project and open it up and move to step 2.

If you have not created the form yet, open a new Standard EXE and in the form, enter the following controls:

A. **Create a folder** on the C drive called **Lab10_yourname.** Save the project and form in this location later.

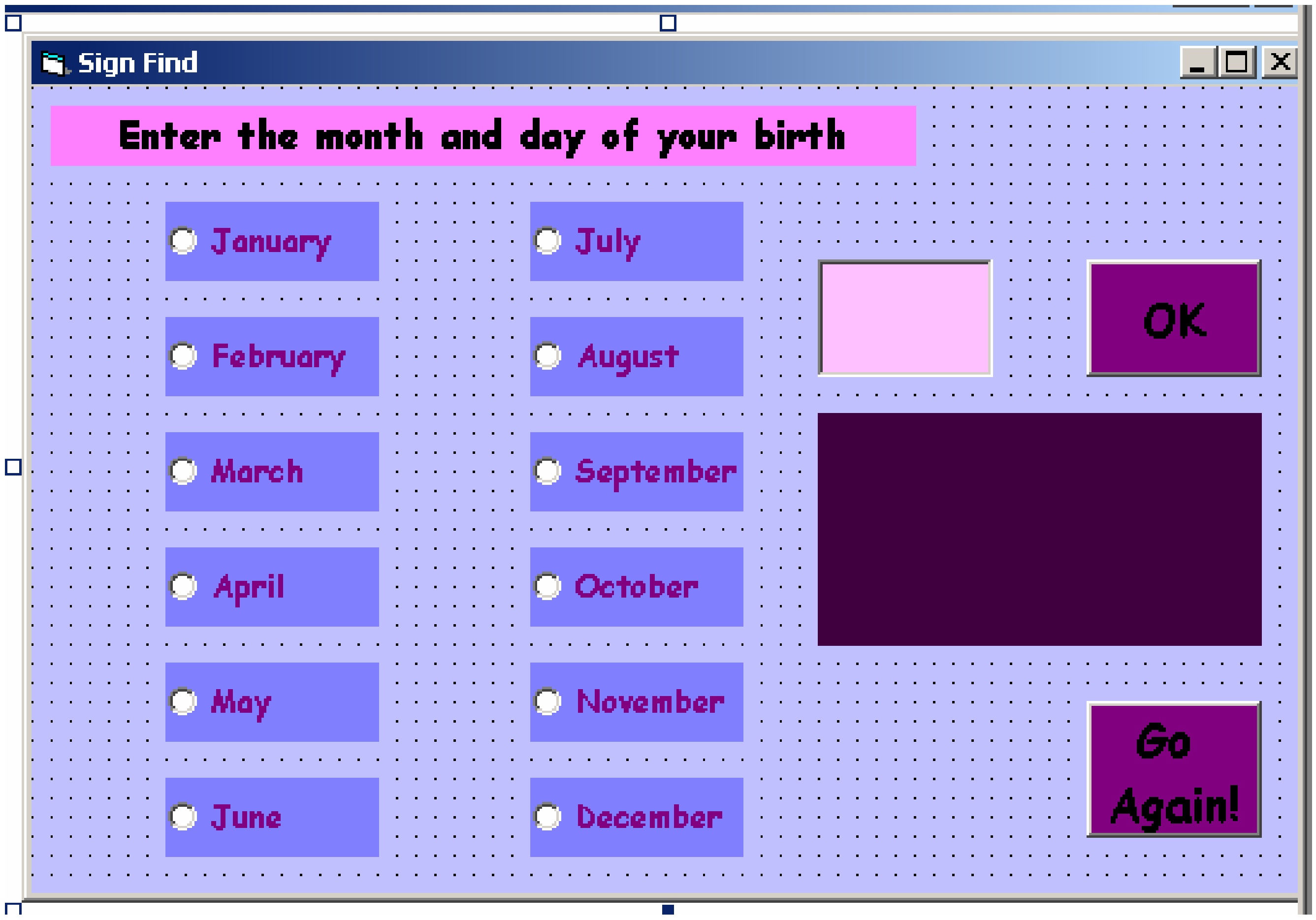
B. **Open up Visual Basic.**

Your form (the interface) for this lab will require the following objects:

2 labels
2 command buttons

1 text box
12 option buttons (radio buttons)

The interface may look similar to the one below, but use whatever colors and setup that works for you:



****A good way to keep track of the names would be to print out the figure above and put the name you will give each control next to its image****

C. Make the captions of the labels and option buttons similar to the interface above. The names of each object should be meaningful to you so that as you code, you will be able to identify them properly in the statements. *Do not include any spaces in an object name.*

What’s in a Name?

A good name for the label giving the user the information to enter their birth information might be: **lblEnter**

A good naming scheme for the option (radio) buttons would be to start each with “opt” and follow it with the month they identify. An example for the January button would be: **optJan**

Name the controls (objects) that are left with names that are usable for you. A suggested naming convention would be the following:

Object Type	**Name starts with**
Form	frm
Command Button	cmd
Text Box	txt
Label	lbl
Option Button	opt
Picture Box	pic
CheckBox	chk
Timer	tmr

D. Change the font colors and background colors to something pleasing.

E. Change the name of your project to **SignFinder** in the **Project > Project 1 Properties** menu and name the form object as **frmSignFinder.**

F. Save the project and the form to the Lab10_yourname folder.

Logic of the program

Once you have the form (interface) taken care of, it’s time to deal with the logic – the code.

The detailed logic of the program is as follows:

A user runs your program at their computer. The form pops up and they are requested to select the option button that states the month of their birthday and to enter an integer value that is their birth date. Once they have done this, they click on the OK command button and a label will appear on the form that states their Zodiac sign.

The problem the programmer encounters is that the signs of the Zodiac do not have a one to one correspondence with each month. Depending on the day involved in any month, a person may be one of two signs. This calls for use of a conditional to help the program decide.

But first, there are a number of things to consider:

Where do you store the Zodiac values?

Every month will potentially have 2 to choose from.

Where do you store the input from the user?

Where do you store the date for each month that will determine whether a user is one sign or another?

2. Creating Variables to hold values

A. Declare 4 separate variables at the top of your program after you enter the **Option Explicit** statement:

- The first variable will hold the Zodiac sign value for the lower half of each month and be a string data type: **loSign**
- The second variable will hold the Zodiac sign value for the upper half of each month and be a string data type: **hiSign**
- The third variable will hold the value of the day each month when the signs change and be an integer data type: **changeDay**
- The fourth variable will hold the value entered by the user that is their birth date and be an integer data type: **birthday**

3. Assigning Values to Variables

A. In each option button click event, assign the correct sign values to the **loSign**, **hiSign** and **changeDay** variables:

Month	loSign	hiSign	changeDay
January	Capricorn	Aquarius	19
February	Aquarius	Pisces	18
March	Pisces	Aries	20
April	Aries	Taurus	19
May	Taurus	Gemini	20
June	Gemini	Cancer	20
July	Cancer	Leo	22
August	Leo	Virgo	22
September	Virgo	Libra	22
October	Libra	Scorpio	22
November	Scorpio	Sagittarius	21
December	Sagittarius	Capricorn	21

B. Where do you go from here?
If a user clicks on any of the radio buttons, the values will be assigned to the variables correctly.

Now it's time to work with the OK button click event. Most of the program logic occurs here.

4. Click Event Procedures

A. In the **cmdOK_Click()** event do the following:

- Assign the value the user enters into the text box to the variable **birthDay.**
  To get to the value entered in the text box, access the Text property.

- Make the label which will display the person's sign and the Go Again command button appear and the text box and OK button disappear. Set the visibility properties to true or false where needed.

- Write a general conditional that will check to see if the value the user has entered is greater than the **changeDay**.

  If it is, then the label that you are using to display the Zodiac sign should display **"Your sign is " & hiSign & "!"** in the caption.

  If it isn't greater than **changeDay**, then the caption will read: **"Your sign is " & loSign & "!"**

> ****NOTE****
> The **&** symbol is used to concatenate string values together [place them together, side by side]. If a value in a variable is an integer, it is converted to a string value for purposes of concatenation.

B. In the Go Again button click event, write the appropriate statements to make the labels and text boxes and buttons appear again to start the program over.
(Think about resetting the option buttons and clearing out the text box.)

C. Save your project.

D. Run your project. Is everything working? What happens if the user enters a letter instead of an integer?

5. Simple Error Trapping

> Often the user will not enter the information we (as programmers) wish them to enter. If they do, for example, enter letters when we expect numbers and we haven't planned for it, it will stop our program. One form of Error Trapping is to let the user know they have entered incorrect information and ask them to enter it again.

A. The syntax used to start an Error Trap statement is the following:

```
Private Sub ….. (this could be any event procedure)

        On Error GoTo <name of error handler>

                <other code statements in the procedure>

        Exit Sub

        <name of error handler>
```

```
                <statements to be executed when error occurs>

        End Sub
```

B. The best place to trap any errors in this program is in the OK button click event, since all the other logic is taking place there as well. Call the error handler **CheckInput**, since it is <u>checking</u> <u>input</u>. You will use a Message box in the error handler statements as a way to alert the user to change their input. The code for the error handler is in color and bold below.

Remember, if you haven't used the same variable names and object names IT'S OK! Your code will look different from the statements below that aren't in bold:

```
        Private Sub cmdOK_Click()

                On Error GoTo CheckInput

                birthDay = txtUserInput.Text

                If birthDay > changeDay Then
                        lblZodiac.Visible = True
                        lblZodiac.Caption = "Your sign is " & hiSign & "!"
                Else
                        lblZodiac.Visible = True
                        lblZodiac.Caption = "Your sign is " & loSign & "!"
                End If

                cmdOK.Visible = False
                cmdGoAgain.Visible = True
                txtUserInput.Visible = False

                Exit Sub

                CheckInput:

                MsgBox "Please enter a valid date and check a month" ,
        VBOKOnly

                txtUserInput.SetFocus

        End Sub
```

A Message Box is used in VB 6 to communicate with the user. It can alert them to enter valid data, confirm an action or simply keep the user informed. Here the box will use an "OK" button as a way to let the user confirm the message.

C. Run your program.
What happens if you enter a letter now?

D. What happens if you enter a number that is negative or above the days in the month? The error trap doesn't yet have a way to account for that, so add another conditional to take care of those potential errors.

The statement to add (if you have used the correct variable names) is in bold below. Notice the Error Trap name is in red:

```
                birthDay = txtUserInput.Text
```

```
If birthDay < 1 Or birthDay > 31 Then GoTo CheckInput

If birthDay > changeDay Then
        lblZodiac.Visible = True
        lblZodiac.Caption = "Your sign is " & hiSign & "!"
Else
        lblZodiac.Visible = True
        lblZodiac.Caption = "Your sign is " & loSign & "!"
End If
```

The bold code statement above now checks to see that the number a user enters is inside of a certain range (between 1 and 31). If it isn't, the message box comes up again and tells them to check their input.

This error trap isn't perfect (what about months with less than 31 days?), but I'll let you add to the code to perfect it if you like!

E. Run your program again.
Try to enter information that will "break" it. Can you do it?

6. Creating an Executable File:

A. Make **SignFinder** executable when you are finished and happy with it. Save your project and FTP the entire **Lab10_yourname** folder containing your project, form and executable to your Dante account. Your TA will let you know if they wish to see this program by the end of this lab or the beginning of the next.

B. Remember to log off the machine as you leave.

7. Lab Notebook Questions

A. Write a conditional that will test the following:

A person can buy sodas or juices at a soda stand. A soda, if that is what the want, costs 75 cents. Since you only sell soda or juice, if they don't want a soda then they must want a juice. Juice costs $1, unless it's cranberry juice. Cranberry juice is $1.25.