FIT 100

Lab 9: Creating the "What's Your Sign, Dude?" Application

Winter 2002

Recommended Reading for Lab 9:

- 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 will 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:

- Understand the importance of object names when changing properties and assigning values with code.
- Be precise in the declaration of variables and understanding how to assign values to them.
- Take input from an unknown user and assign it to a variable and use it as part of the logic of the program.
- Get familiar with conditionals and error trapping.
- Steadily add pieces of code to your program and increase its complexity.

TO DO:

1. **Create a folder** on the local drive called **Lab9.** You will save the project and the form in this folder.

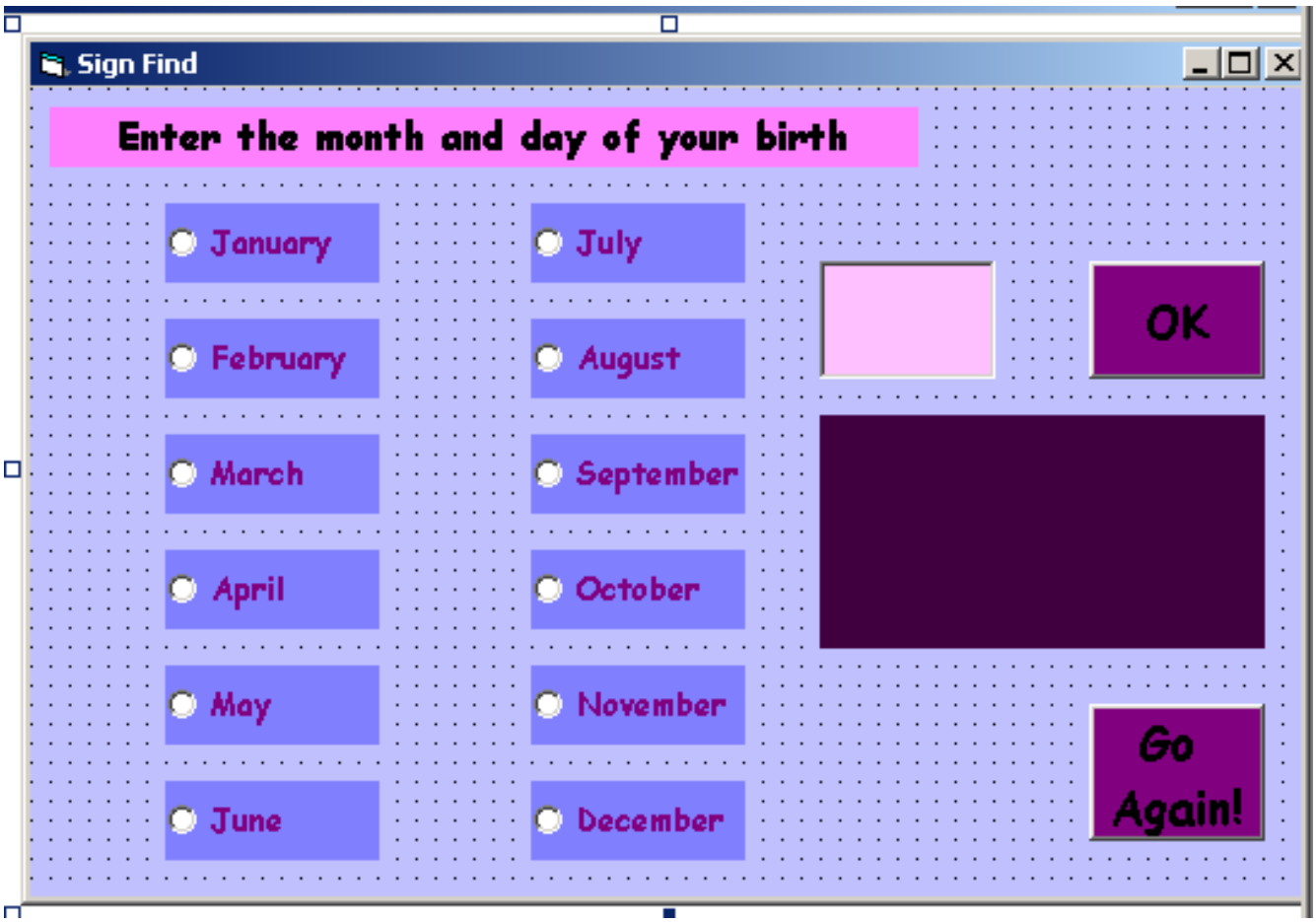
2. **Open up Visual Basic.**

   If you have created a form and all the required controls for Lab 9 already, download it from Dante into the Lab9 folder. Then go to that project and open it up.

   If you have not created the form yet, open a new Standard EXE project. Add the following objects to the form:

   - 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:



3. Make the captions of the labels and option buttons similar to the interface above.

You will need to give meaningful names to these objects so that as you code, you will be able to identify them properly. For example, a good name for the label object that invites the user to enter some information would be **lblEnter**. A good name for the January option button would be **optJan**.

We suggest that you use the following VB naming convention. The name starts with a prefix that indicates the type of the object, followed by some short string corresponding to the meaning of the object. For example, in **optJan**, "opt" indicates that this is an option button; "Jan" indicates that this is the button for January. The prefixes are listed in the following table:

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

4. Change the names of your objects following this naming convention. *Do not include any spaces in an object name*.

****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.****

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

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

7. Save the project and the form to the Lab9 folder. Be sure to FTP this folder to your account on dante before leaving.

Logic of the program:

Once you have the form (interface) taken care of, it's time to deal with the program logic – and then create the corresponding code. Here is how your program should behave.

When the user starts your program, 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 appears on the form that states their Zodiac sign The Go Again button appears as well. At this point the text box and OK button disappear (they have done their job). Clicking on the Go Again button makes them reappear and ready for more user input.

An important detail to notice is that the signs of the Zodiac do not have a one-to-one correspondence with each month. If the birth date falls on the earlier part of the month, the person is of one sign, which we will call "low sign." Otherwise, the person is of another sign, which we will call "high sign." The day of the month up to (and including) which the person is of the low sign will be called the "change day." Finally, low sign, high sign and change day all depend on the month of the birthday.

You will need to create a place to store the two Zodiac values and the change day for each month, and also to store the input from the user. Finally, you will need to use a conditional to decide whether the person's sign is the low sign or the high sign.

Creating Variables to hold values:

8. To store the information discussed above, declare 4 separate variables at the top of your program after you enter the **Option Explicit** statement as follows:

LoSign and **hiSign** - to hold the low and high signs for the selected month; they should be of string data type;

ChangeDay - to hold the change day for the selected month; it should be of integer data type;

birthday - to hold the value entered by the user that is their birth date; it should also be of integer data type.

Assigning Values to Variables:

The program will store the low and high signs and the change day into the above variables when the user clicks on an option button for some month. Correspondingly, the code to do so needs to appear as a handler for the click event.

9. In the code window select, in turn, the option button for each month, and then the click event for that button. Then enter code to assign values to the **loSign**, **hiSign** and **changeDay** variables according to the following table:

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

10. Test your code by running it. Click on one of the month buttons. Your code for that event should have been executed. Has it? Interrupt the program by clicking on the Pause (or "Break," in VB jargon) button on VB toolbar. Click on the code window and bring your mouse, in turn, over the name of each of the four variables at the top of the window (after the **Option Explicit** statement). You should see the tip box indicating the current value of that variable. Be sure the loSign, hiSign and changeDay variables have correct values for the month you clicked on last.

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

Click Event Procedures

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

    a. 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 of the text box object.

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

c. 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 "!"**

   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.

12. 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.

13. Save your project.

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

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. We say that an error has occurred. The program may want to react to such error by Error Trapping. For example, the program may let the user know they have entered incorrect information and ask them to enter it again.

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
```

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 checking input. You will use a Message box in the error

handler statements as a way to alert the user to change their input.

15. Add the code for the error handler, indicated 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 (except the statements that are in bold – those should look same):

```
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.

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

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 we need another conditional to take care of those potential errors.

17. Add the statement in bold below (adjust the variable names as appropriate). 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 we will handle that later.

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

Creating an Executable File:

19. Make **SignFinder** executable when you are finished and happy with it. Save your project and FTP the entire Lab9 folder containing your project, form and executable to your Dante account.