Visual Basic

Standards for Project Evaluation

1. Project Performance

a) Project works according to specifications.
- Your project must follow the requirements outlined in the textbook assignment as well as any additional requirements assigned by your instructor.

b) Output is accurate.
- It is not enough to just produce output (displayed or printed). Your output must be complete and 100 percent accurate.

c) Logic is efficient.
- Use the most efficient logic that the textbook or your instructor has covered.
  Examples:
  Declare and use appropriate variables for calculations.
  Use With/End With rather than repeat an object name.
  Use Select Case rather than If…Then…Else when appropriate.

d) Meets all requirements.
- Follow any special rules and requirements of your instructor.

2. Disk Storage

a) Folder name and file names match those listed above
- Fill in the top of the Project Evaluation Sheet, including the name of the disk folder where the project exists and the name of the .sln file for the project. The names you enter must exactly match those found on the diskette, including correct spelling and punctuation.

b) Folder contains all necessary project files (no extra files)
- A folder on diskette was created specifically for this project. The folder must contain all of the project's files. Files unrelated to the project should be moved or deleted.

c) Project runs as submitted
- Make sure that the project compiles and runs correctly from the diskette that you submit. Problems can occur when any of the project's files are not stored in the project's folder. If your project cannot locate a required file - your project will not run!
- Test the project; make sure to test it on a computer other than the one on which you wrote it.

3. User Interface

a) Professional appearance

Layout, placement, spelling
- Designing the user interface is a critical component in the project's construction. Anything that will appear on the screen for the user (forms, message boxes, etc.) must be presented in a professional format. The screen design should be easy to understand and "comfortable" for the user.
- Use correct spelling and punctuation.
- Name the form and change the Text property. The Text property appears in the form's title bar.
- Set the StartPosition and WindowState properties to determine where/how your form will appear.
- Make sure that any text on the form displays completely, during both design time and run-time.

b) Follows Windows standards
- Follow industry standards in relation to color, size, and placement of controls. (Refer to Chapter 2, "Designing the User Interface".)

Keyboard access keys (& in Text property, AcceptButton, CancelButton set)
- Use keyboard access for all buttons and menus. For some assignments, you will also be required to include access keys for text boxes, radio buttons, and check boxes.
- Follow industry standards whenever possible; use the *X* of Exit, and the *S* of Save.
- Do not give two controls the same access key.
- Set one button as the AcceptButton (it will respond when the user presses the Enter key), and set one button as the CancelButton (it will respond when the user presses the ESC key). Be aware that some users are accustomed to pressing the Enter key instead of using the Tab key to jump to the next field.

Tab order is correct.
- Make sure the focus is on the correct object when a form displays. The tab order must proceed correctly when the user presses the Tab key.

4. External documentation

a) Professional presentation.

b) Project Folder: outside label, evaluation sheet, divider tabs, disk or other media is securely attached, all printouts are included (forms, code, etc.).

- The folder, label, and printout must be neat. The entire project must look professional.
- Use a folder with three prongs. Fasten the pages into the folder.
- Write your name and the project number on a label on the front of the folder. This information must be visible without opening the folder.
- Include a Project Evaluation Sheet, with the student information completed, as the first page in your folder.
- Use divider tabs to separate the contents of the folder. Label the divided sections in the folder as listed below, and in this order:

  Note: These requirements are for *each form* in the project.

  1) Interface Plan (Opt.) --- A sketch of each of the screens the user will see when running your project. Show the forms and all the controls that will be used. Indicate the names for the form and each of the objects on the form.
  2) Properties Plan (Opt.) - List the properties and settings for each object on the form. The list should contain three columns: Object, Property, and Setting.
  3) Code Plan (Opt.)-------- Pseudocode for each event procedure.
  4) Form Image ---------- Graphical version of the form printed (press the PrintScreen button on the keyboard and paste the screen capture into a word processor).
  5) Code ---------- The form's code printed using File/Print.
  6) Printer Output---------- Output produced from statements located in an event procedure. (if required by project – Printing is discussed in Chapter 7).

  NOTE: Requirements may vary, depending on instructor preference.
- Your disk or other media must be securely attached to the folder. (You can create a pocket on the inside back cover of the folder.)

5. Standards and conventions

a) Object names.

- Do not keep the default names assigned by Visual Basic, such as Button1 and Label3. The exception to this rule is for labels that never change during project execution. These labels usually hold items such as titles, instructions, and labels for other controls.
- Use good, consistent names for objects.
- Use camel casing and end each name with the class of the object. (Refer to Chapter 1, "Naming Rules and Conventions for Objects.")
- Object names with multiple words should have the first character of each word capitalized. Examples: redRadioButton and lastNameTextBox.
- Do not name your objects with numbers.
- Object names must begin with a letter or an underscore. The name can contain letters, digits, and underscores. An object name cannot include a space or punctuation mark and cannot be a reserved word.

b) Identifiers for Variables and Named Constants.

- Follow the textbook's standards and conventions when naming (identifying) your variables and constants. (Refer to Chapter 3, "Naming Rules" and "Naming Conventions".)
- Variables: Identifiers must be meaningful. Do not abbreviate unless the meaning is obvious and do not use very short identifiers, such as X or Y. Include the class (data type) of the variable. Begin with a lowercase letter and then capitalize each successive word of the name. Example: firstNameString.
- Constants: The name portion should be entirely in uppercase, with multiple words separated by underscores. Examples: TAX_RATE_Decimal, COMPANY_NAME_String.
- Identifiers for variables and named constants may consist of letters, digits, and underscores. They must begin with a letter; they cannot contain any spaces or periods, or be reserved words. Reserved words are words to which Basic has assigned some meaning, such as print, name, and value.

c) Option Strict On / All variables declared

- In VB.NET, Option Explicit is turned on by default. The best practice is to turn on Option Strict either in code or in the Project Designer. Make sure the Option Strict On statement appears above the Public Class statement. If Option Strict is turned on, variables must be declared, regardless of the setting of Option Explicit. (Refer to Chapter 3, "Option Explicit and Option Strict")
- Declare local variables at the top of a procedure, below the remarks.
- Declare module level variables and constants in the Declarations section.

d) Remarks in the Declarations section.
- Include the following information at the top (Declarations section) of every form in your project:
  Project number
  Date
  Programmer Name
  Program Description

e) Descriptive remarks in every procedure.
- Write remarks describing the purpose of the procedure at the top of every procedure. Use correct spelling and punctuation in all remarks.

f) Proper indentation.
- Follow the textbook coding conventions and indent all lines in sub procedures and sub functions.
- Do not indent the lines in the Declarations section of a form.
- Indent all lines inside If/Then/Else, With/End With, For/Next, For/Each, Do/Loop, Select Case, Type/End Type, Try/Catch for readability.

Examples:

```
If nameTextBox.Text <> "" Then
    messageLabel.Text = "Hello there, " & nameTextBox.Text
Else
    messageLabel.Text = "Tell me who you are."
End If

With FontDialog1
    .Font = messageTextBox.Font
    .ShowDialog()
    messageTextBox.Font = .Font
End With

For counterInteger = 1 to 100
    If counterInteger = matchInteger Then
        messageTextBox.Text = "Match found at line number " & counterInteger
    End If
Next counterInteger

Try
    quantityInteger = Integer.Parse(quantityTextBox.Text)
    quantityTextBox.Text = quantityInteger.ToString()
Catch theException as FormatException
    MessageBox.Show("Error " & theException.Message)
End Try
```

g) Proper blank lines.
- Insert one blank line after the remarks at the top of each procedure.
- Insert one blank line after the declarations for variables and constants.
- Insert one blank line between procedures.

h) No unused procedures included.
- Clean up! Delete any procedures that do not contain code.

---

6. Timeliness: See course syllabus for late penalties.

7. Extra credit: