MALMÖ HÖGSKOLA

Malmö University
School of Technology

Programmering med Visual Basic, grundkurs
Programming Using Visual Basic, Basic Course

Assignment 1 – Console Application

Apu's Gas Station

Mandatory

Farid Naisan
University Lecturer
Department of Computer Sciences

Introduction:

This assignment has two parts. In the first part, you will create a console application, Apu's Gas Station, and then define one of your own objects in Part 2 in order to get more training in getting started.

In the following section we´ll discuss the Quality Standards and Guidelines for this assignment. Some of the topics there may be hard to grasp as you haven't read the description for the rest of this assignment. Please skim through this section to start with and if there are things you don't understand, read the rest of the assignment and then, after you have completed the assignment, read this section again in order to see if your solution achieves the Quality Standards and Guidelines required..."

Quality Standards and Guidelines

Requirements:

- ✓ In both Part 1 and Part 2 of this assignment, each project must have at least one class and one (and the only one) module, (for example a Class **GasPump** and a Module **GasStationProg** in Part 1). Every class (the module is also a class) is to be saved in a separate file, for example **GasPump.vb** and **GasStationProg.vb** respectively. If you are using Visual Studio, it will prepare a default Module for you.
- ✓ You can use a standard text editor such as Windows Notepad.exe to write the source code, compile and run from the command line, or use other development environments (known as IDE, Integrated Development Environment) such as Visual Studio to create the projects, write code, build and run from the IDE.
- ✓ All instance variables must be declared as private.
- ✓ Both Option Strict and Option Explicit must be set to "on" **in all code files**. Copy the following two lines at the top part of each of your code files, including the code file of your MainForm (used in later assignments):

```
Option Explicit On
Option Strict On
```

- ✓
- ✓ Make your own assumptions whenever you find the instructions unclear. Document your assumptions directly in the code or as an extra note to your instructor.
- ✓ The assignments are designed for beginners. You may therefor practice improvements, optimizations and enhancements to the coding structure and the user interface if you are more experienced. However, certain instructions may be marked as mandatory in which case you are expected to follow the instructions.
- ✓ Write your name and date in the top of all your source files (as comments).
- ✓ Do not forget to document your source code by writing comments explaining your code above or in front of your statements

Quality

- ✓ The application must have been compiled, tested and run satisfactorily before it is submitted. It is very important to maintain a good style and code structure and a normally functioning program, rather than presenting a well function program, but with poor code quality. Higher grades are given to projects containing well-structured and reasonably documented code, and a satisfactorily working program.
- ✓ All identifiers (class names, variable names and method names) should be chosen carefully so that they express their purposes. The suggested solutions to the exercises can be taken as a guide. Short names that are not expressive (ex a, b, fnc, etc.) should be strictly avoided.

Assessment:

- ✓ The assignment receives a letter grade A-F which can be then translated to a grade type of your choice. Projects that do not meet the minimum criteria for a passing grade will be returned for complementary work. The final result for all assignments will be determined as a weighted average of all assignments plus the instructor's judgment of the progress you make throughout the course.

Submission

- ✓ The files are to be packed into a ZIP, 7Z or RAR file and submitted inside It's Learning (ITS). Make sure to include all the files that are part of your project. If you are using Visual Studio for this assignment, include all your project's files and subfolders in your ZIP or RAR file. Sending only the **sln** or **vbproj** file is not enough. Include even the folder **My Project**.

Although it is allowed to discuss solutions and ideas with other classmates, the assignment is done and submitted individually. Make good use of the forum on ITS for this assignment, but when helping others do not put more than a few (4-5) lines of code. You must apply your own solution and you are not allowed to copy and submit the same code as any other. It is expected that you have gone through all the recommended readings, done exercises and eventual quizzes before starting with this assignment. If you understand and can follow the optional exercises, and the code examples available for this assignment, you shouldn't face any problem in completing this assignment.

Assignment 1

1. Objectives

- To work with simple classes and objects
- To create a first VB program as a Console Application
- To work with primitive data types such as text, numbers and logical values.
- To use simple variables to store values and write methods to manipulate them.
- Write data to a console window by using the **Console.Write** and **Console.WriteLine**.
- Read data from a console using the **Console.Read** and **Console.ReadLine**.

It is assumed that you have studied the related chapters in your textbook, reviewed the reading materials, studied the code examples and done the optional exercises, available on ITS before starting this assignment.

2. Description

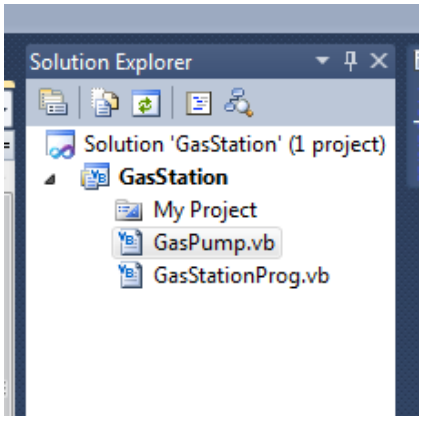
This assignment consists of two parts. In the first part you are given a problem with some instructions on how to solve it. In the next part you are required to find and implement a task by yourself.

3. Work Plan

- Do the Part 1 first and then Part 2. In Part 1, you should program the **Gas Station**, and in Part 2 you choose an object by yourself and program it just as in Part 1.

- In both of these parts you write two classes (one class and one module), the module with the method Main and one class that programs an object.

- Create a Visual Basic Console Application project using Visual Studio. The IDE will create a solution for you and put the project into this solution. In this document we refer to this project as Assignment1. In the creation dialog box you have a choice for choosing the location of your project. Select a new folder (GasStation, or Assignment1) on your computer. This way you will not mix files and components from different projects.



- The IDE has also prepared a default **start Module,** Module1.vb, for you in which the IDE has prepared a Main method.

- Rename the **Module1.vb** to **GasStationProg.vb** in the Solution Explorer. It´s inside the Main method we´ll create an instance of the **GasPump** class to start the program. Don't forget to save your project often.

4. Assessment

In order to get a passing grade, Part 1 must be completed. To get a higher grade, even Part 2 must be carried out with good quality considerations. Unsatisfactorily done projects will be returned for completion.

Mandatory for both parts: One (and no more) module as a start class with the Main method (for ex GasStationProg) and one class for solving the problem (for ex GasPump). The module should not contain any instance variables or methods other than the Main method.

5. New Terms

User	A person that runs the program. You are the user when you run and test your program.
Read Input	The user gives input through the keyboard (or mouse) to your program. The input can be a text or a numerical value but is received and interpreted by your program as text. The text needs to be converted to the correct data type (Integer, Double, String, etc). The values are then saved in variables inside your class for manipulation.
Print or show results	Show somehow the results to the user using the Console.Write or Console.WriteLine methods.
Application	All files and components that makes your program.
Console Application	An application that is run on a command prompt window (see the figure below).

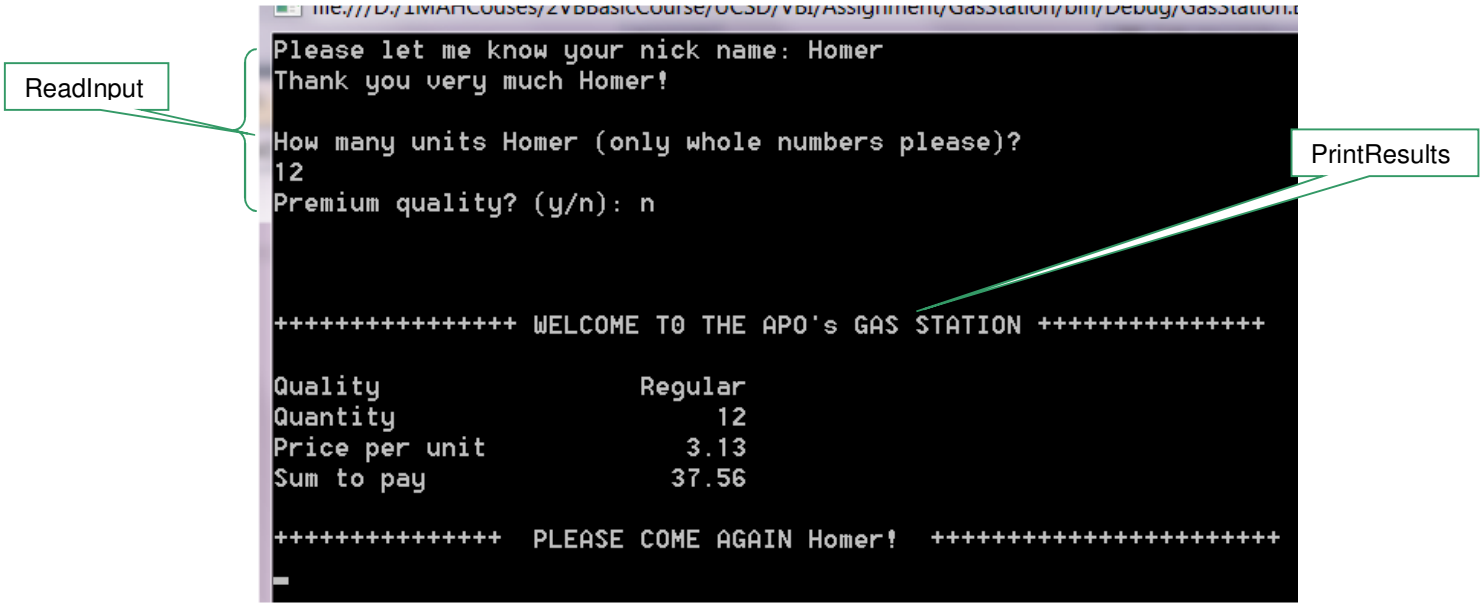
6. Part 1: Gas Station

Apu's Gas Station would like to offer his customers pay-at-the-pump capabilities. The pumps need a software that interacts with the customers when purchasing fuel. Apu will try this service first on his gasoline pumps and wait with Diesel and other types until a later occasion in the future.

Your job is to write a simple computer program that interacts with the user through a simple interface. The user should be able to select the quality of gasoline, either Regular or Premium, and input the volume in whole liters. The program should then calculate the amount due and print a receipt on the screen.

As to the price and unit (liter), you may pick the prevailing prices and unit used in your city at the moment. You may also use the prevailing currency.

The pumps require that the customer pays for the fuel before filling up the tank. Therefore the user needs to input the amount of petroleum (in whole liters) and quality. The program then calculates the amount to pay and prints a receipt as shown in the run example below.



6.1. Instructions

Write one module (only one), **GasStationProg**, that contains the **Main**-method and one class **GasPump** that carries out the job:

- Asks the customer for her/his name, quantity (volume to tank) and the quality of gasoline, Premium or Regular.
- Calculates and presents the amount to pay (something like the run example above).

Begin writing the **GasPump** class according to the following specifications:

6.2. The GasPump Class

Create a text file and save it as **GasPump.vb** in your project folder. Write then the class **GasPump** according to the following guidelines:

6.2.1. **Fields** (aka instance variables, or attributes): **customerName** (String), **quantity** (Double), **premiumQuality** (Boolean) for storing the user's input into the program, and **totalToPay** (Double) for output from the program to user.

You can hard-code the values of petroleum in a couple of constants as follows:

```
Private Const regularPrice As Double = 13.08  'Lund 2011-01-21
06/30/2010
Private Const primePrice As Double = 13.56
```

Note: In later assignments, we will learn to reduce the number of output variables as much as possible by replacing them with methods in which the value is calculated and returned every time the method is called. We will also later on learn how to isolate data presentation (user interface interactions) from data logic.

6.2.2.All fields are to be declared as Private.

6.2.3.**Methods:** Write a method, **Start,** that encapsulates the following steps:

- Reads input from the keyboard, (**customerName, quantity, premiumQuality**) and saves them into the instance variables (fields).
- Calculate the total amount to pay and save the result into your variable **totalToPay.**
- Print a receipt using the calculated values (output) as shown in the example above.
- To help you here, the skeleton of some of the methods are provided below.

```
Public Sub Start()
    '1.  ReadInput
    ReadInput()
    '2.  Calculate
    CalcTotalToPay()
    '3.  Show results
    PrintReceipt()
End Sub
```

6.2.4.Write the following methods into your class and complete the internal methods used:

```
Private Sub ReadInput()
    '1.  Read customer name
    ReadCustomerName()
    '2.  Read Quantity
    ReadQuantity()
    '3.  Read Quality
    ReadQuality()
End Sub
```

The methods contained in the above two public methods are used internally in the class (methods not called from outside the class) and therefore should be declared as Private.

```
Private Sub ReadCustomerName()
    'Write your code here
End Sub

Private Sub WriteThankU()
    'Write your code here
End Sub

Private Sub ReadQuantity()
    'Write your code here
End Sub
```

```
Private Sub ReadQuality()
    Console.Write("Premium quality? (y/n): ")

    Dim response As Char = Console.ReadLine().Chars(0)
    If ((response = "y") Or (response = "Y")) Then
        premiumQuality = True
    Else
        premiumQuality = False
    End If
End Sub
```

6.2.5. Complete also the other methods in the same way.

6.2.6. The method **WriteThankU** is intended to be called from the method **ReadCustomerName** after the name is read from the user.

6.2.7. When you are done with the above, you are done with the **ReadInput**. You now have to write the two other methods contained in the **Start** method.

```
Private Sub CalcTotalToPay()
    'Write your code here

End Sub

'Print results
Private Sub PrintReceipt()

    'Write your code here

End Sub
```

You may write more methods for other tasks that can be separated from the body of another method.

6.3. The GasStationProg Class

6.3.1. In order to test the **GasPump** class we need to complete the program by a start Module with a Main-method. This method will be the one the CLR (the Common Language Runtime) will look for to start your program. Although you can have methods and instance variables in this class (module) just like any other class, it is recommended to keep this class as small as possible.

3.1.1 Based on the above recommendation, this Module should only contain the **Main** method and shall not have any instance variables.

3.1.2 Create an object of the **GasPump** class (see below) in the **Main**-method and then call its Public method **Start**. You may use the following code.

```
Option Explicit On
Option Strict On
' This is a comment line and will not be compiled

' GasStationProg.vb
' Created:  By Farid Naisan 06/30/2010
' Revised:
'
''' <summary>
''' This is a comment block that is xml-compatible.  There are programs
''' (plug-ins, http://www.roland-weigelt.de/ghostdoc/) that can translate
''' these automatically to html pages.
'''
''' This program is made for pay-at-the-pump gas stations.It interacts with
''' the customer and calcualtes the amount to be paid before filling up the car.
''' </summary>
''' <remarks></remarks>
Module GasStationProg
    ''' <summary>
    ''' This is where the program begins when the program starts.
    ''' </summary>
    ''' <remarks></remarks>
    Sub Main()
        Dim pump As GasPump 'Declare a ref variable
        pump = New GasPump()  'Create to object
        pump.Start()

        'Make the Console window wait on the screen ("Read nothing")
        Console.ReadKey()
    End Sub
End Module
```

And these lines are henceforth **super-mandatory**, in all vb files and all assignments!

All your vb files must henceforth contain this info, in all assignments.

Document your code like this so others can read your mind!

7. Part 2: Program Your Favorite Object

7.1. Look around you at home, at your work or where you´re now. You may find numerous objects, a chair, a baby, a friend, a car etc. around you. Choose your one favorite object and program it as in Part 1.

7.2. For the object you have chosen, determine at least three attributes (fields) that best describes the object of this type. Then determine a data type (String, Integer, Double etc.) that the attribute can be represented by.

7.3. Think about at least two operations (methods) that can be performed on the object using the attributes.

7.4. The object does not have to be a physical thing; it may be any object like “Address”, “Movie”, etc.

7.5. Create a new Project in your Solution in VS and follow the same procedure as in Part 1.

8. Links

MSDN: How to Create a Console Application (click on the VB tab)
http://msdn.microsoft.com/en-us/library/ms438026.aspx

Compiling:
http://msdn.microsoft.com/en-us/library/s4kbxexc.aspx

System.Console:
http://msdn2.microsoft.com/en-us/library/system.console(VS.71).aspx (Check the class members: Read, ReadLine, Write and WriteLine).

Good Luck!
Programming is fun. Never give up. Ask for help!
Farid Naisan,