



**MALMÖ HÖGSKOLA**

**Malmö University**

**School of Technology**

Programming Using Visual Basic, Basic Course

# Assignment 2

## Selection and Iteration Algorithms

TimeTracker

**Mandatory**

[Farid Naisan](#)

University Lecturer

Department of Computer Sciences

## Table of Contents

1. Objectives .....	3
2. Description .....	3
3. The Project .....	3
4. Basic Requirements .....	5
5. Advanced Requirements .....	8
6. Step by Step Instructions.....	8
A work plan .....	8
The Menu Class .....	9
The Input Class .....	11
The WholeNumbersForAdd Class (for-statement) .....	12
The FloatingNumbersWhileAdd Class (while-statement) .....	13
The CurrencyConverter Class (do-while-statement) .....	15
The Time Tracker WorkingSchedule Class .....	16
Some hints .....	17

## Assignment 2

### 1. Objectives

The main objective is to provide training in iteration and selection algorithms, things that are a part of a programmer's everyday life.

Several code excerpts are provided in this assignment to prepare you for your future assignments.

It is expected that you follow the guidelines given in the document "General Quality Standards and Guidelines" before beginning with this assignment.

### 2. Description

This assignment consists of a number of sections. In the first two sections, you will be writing a program that sums up numbers. Then you will write a class for converting the local currency to a foreign currency. In the last section, you are asked to write a program to help a worker calculate what weekends and nights to work.

A menu is presented to the user at the program start. The user selects an item and the program performs the job using an object of the class related to the corresponding task.

In the first parts of the assignment some parts of the code is given to help you get started. If you wonder why the code is given in image format, the reason is to avoid copy paste. To write the code by yourself makes you more aware of the structure and the syntax and gives you the possibility to ponder why things are written in a certain way.

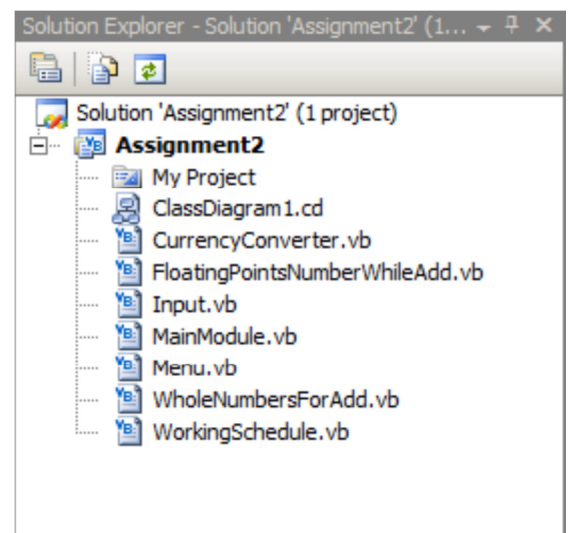
The input and output operations are performed from and to a console window. A different iteration algorithm is used in every part, except in the last section where you are given the freedom of selecting the algorithm by yourself.

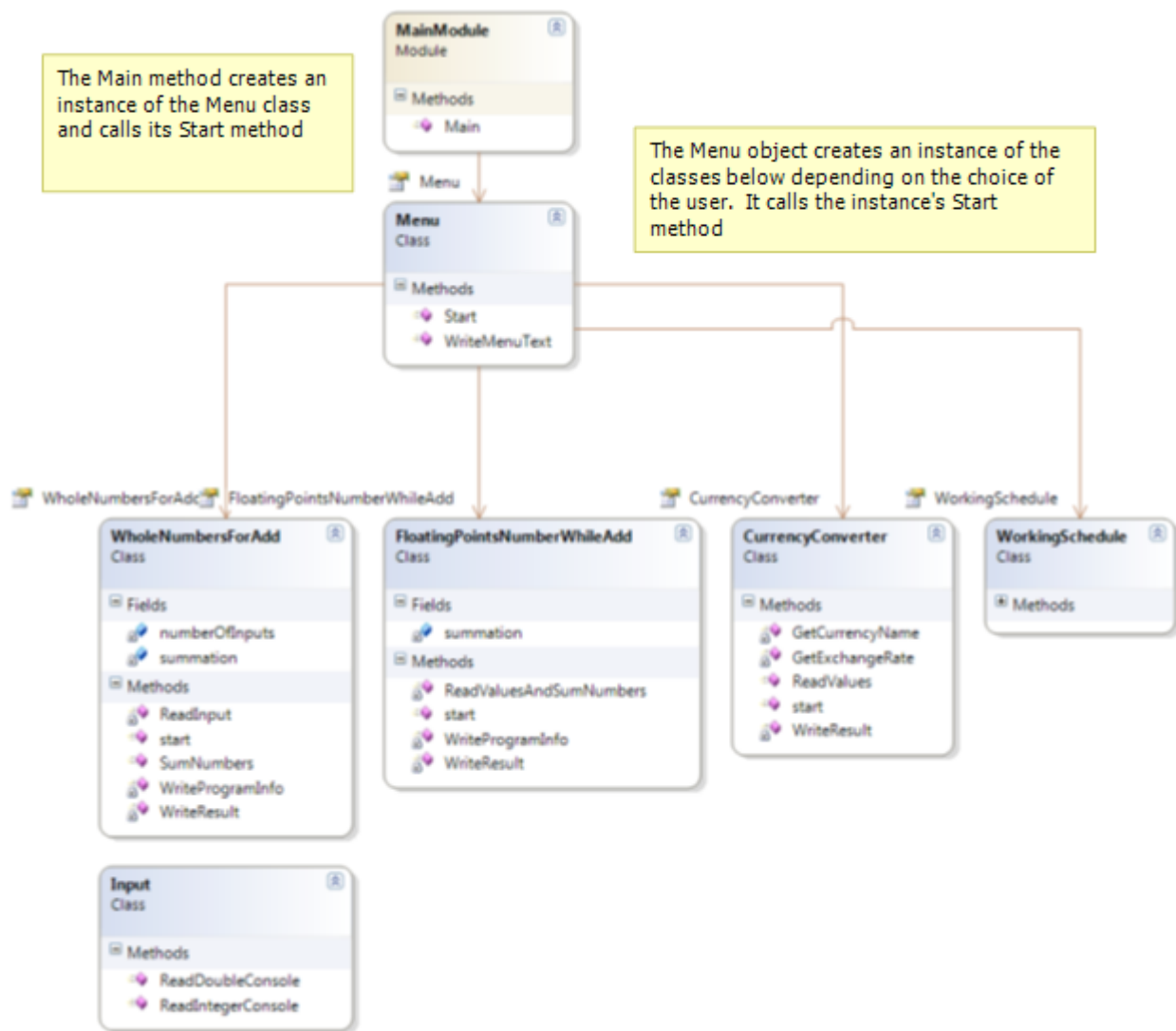
The assignment will also use both forms of selection algorithm, e.g. if-else and switch statements. A good deal of hints and guidance is provided in the first parts to prepare you gradually to get on your own.

It is not meant to use arrays in this assignment, but if you are an experienced array user, you may go ahead and do that. We will be working with arrays in a later assignment.

### 3. The Project

The figure shows a project created in Visual Studio, containing the classes that you are expected to write. Even if you don't use any [IDE](#) and write your code in a simple editor, make sure that you gather your source code files under a separate directory for this assignment. This is the folder that will be referred to as "the project" in this document.





The classes are also illustrated graphically in the diagram that follows. The diagram is called a “Class Diagram” in object-modeling languages like UML (Unified Modeling Language). For more information about UML, search on Google for “UML tutorial” or visit <http://www.omg.org/>.

The boxes represent the classes and the arrows show the association between them. The arrows go from the class that uses, i.e. creates an instance of, to the class that it points to. This sort of association is known to have a so called “Has-a” relation, meaning that one object has another object as its component.

Each box has three compartments, for class-name, for attributes (fields) and for methods of the class.

When an object depends on another object, or in other words, when an object is built up of (contains) other objects, it is called association. The association is of the type “has a” relation which means that the object contains the other object. A Car object has for example a number of Wheel objects, a Cd-player object etc.

An object of the MainProgram class “has an” object of the Menu class which in turn has objects of the classes FloatingNumberWhile, ExchangeDoWhile etc. Notice that the arrow used for this purpose is drawn with open head.

## 4. Basic Requirements

Here follows the basic and minimum requirements that are required to pass the assignment. So before handing in the assignment go back and read this section to check that you have fulfilled these requirements.

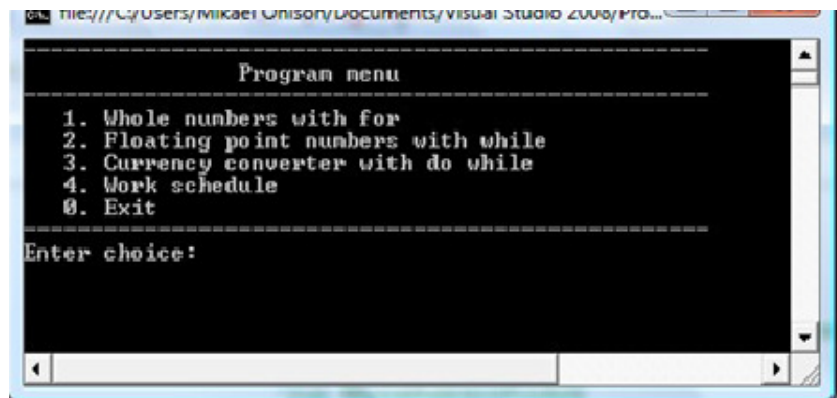
4.1. You can implement the application as a console application or a windows form application (the description below is for a console application so we recommend that you write your program as a console application). If you choose to implement your program as a windows application you need to use the same loops, which are specified later, as if you were to implement a console version.

4.2. Each of the classes that you implement in this assignment and in all the following assignments needs to have their own file. It is not allowed to have two classes in the same file.

4.3. You must have the following classes/Modules:

**a. MainModule**

The class starts up the application by initiating the Menu class and calling its Start method. Here you must use case-switch to handle the user input.



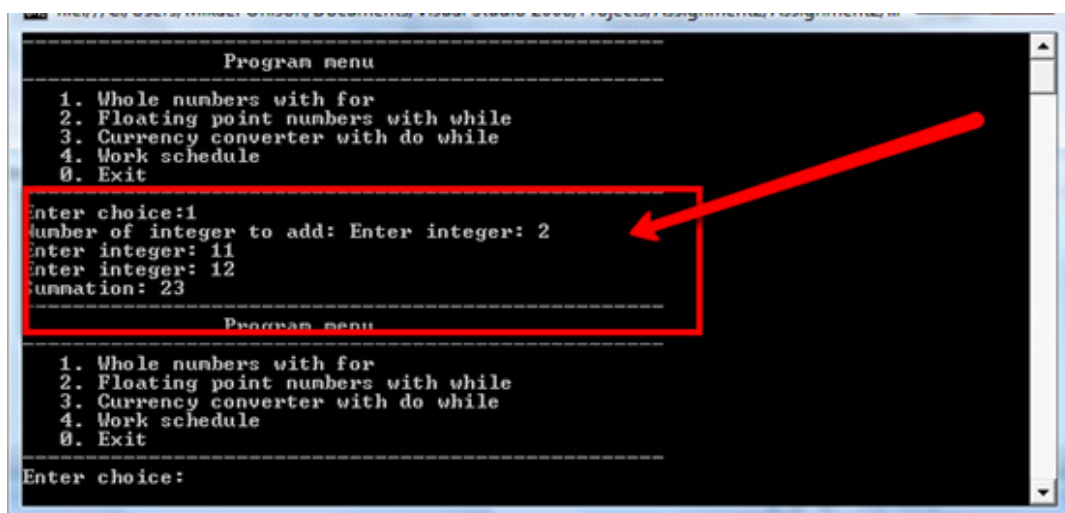
**b. Menu**

Displays the menu and reads the menu input. The menu should look something like in the image above:

**c.** You might change the looks but it should contain all of the 5 different choices. The menu is to be run in an endless loop and does not stop until the user chooses to exit.

**d. WholeNumbersForAdd**

Uses a **For loop** and asks the user for how many **whole numbers** to read. The class is responsible for reading in the numbers and a summation of them (adding a negative number will subtract the number). Finally it will show the result when all the numbers have been added.



```

Program menu
-----
1. Whole numbers with for
2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
5. Exit

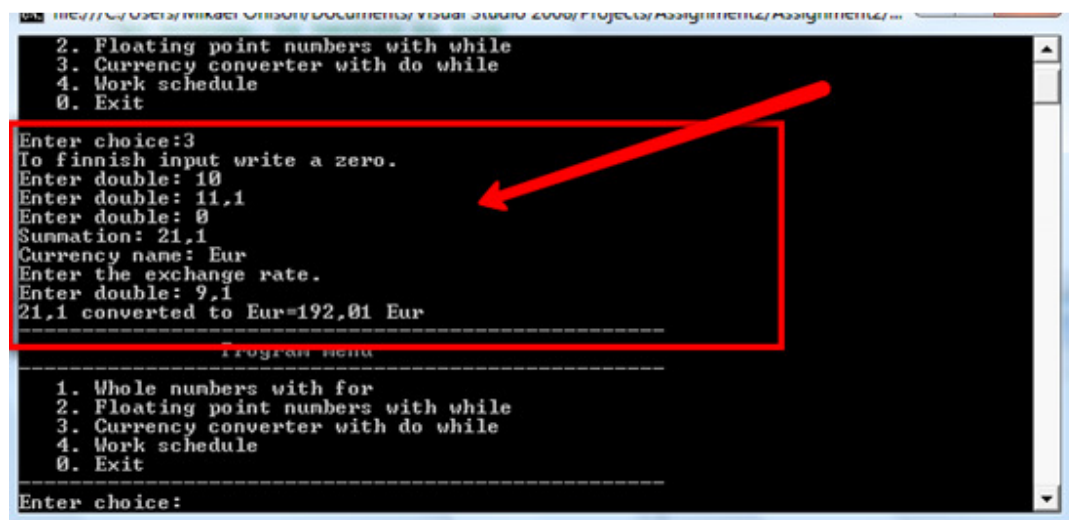
Enter choice:1
Number of integer to add: Enter integer: 2
Enter integer: 11
Enter integer: 12
Summation: 23

Program menu
-----
1. Whole numbers with for
2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
5. Exit

Enter choice:
  
```

#### e. FloatingPointsNumberWhileAdd

Uses a **While loop** (not a Do While). Reads **floating point numbers** from the user and does a summation of them. When the user writes a zero the loop ends and the summation is viewed (adding a negative number will subtract the number).



```

2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
5. Exit

Enter choice:3
To finnish input write a zero.
Enter double: 10
Enter double: 11,1
Enter double: 0
Summation: 21,1
Currency name: Eur
Enter the exchange rate.
Enter double: 9,1
21,1 converted to Eur=192,01 Eur

Program menu
-----
1. Whole numbers with for
2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
5. Exit

Enter choice:
  
```

#### f. CurrencyConverter

Uses a **Do-While loop** (not a While loop) Reads **floating point numbers** and adds them together (as above) and when the user writes a zero the input finishes. The application then asks for the currency name to exchange to and the exchange rate. Finally it displays the exchange rate for buying the currency entered.

```

2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
0. Exit
Enter choice:3
To finnish input write a zero.
Enter double: 10
Enter double: 11,1
Enter double: 0
Summation: 21,1
Currency name: Eur
Enter the exchange rate.
Enter double: 9,1
21,1 converted to Eur=192,01 Eur
-----
Program menu
-----
1. Whole numbers with for
2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
0. Exit
-----
Enter choice:

```

### g. WorkingSchedule

In this class it is up to you to decide what kind of loop to implement (if you want, you can even use a recursive function). You have to write the reason or rationale of your choice, in a comment, describing why you selected the specific loop. The worker, who will use your program, has to work every third weekend with start week number six and every sixth week he works night with start on week one.

You must make the program show the worker when he is supposed to work weekends and when he is supposed to work night. This should be divided into two parts but in one class (one for weekends and one for nights). See the illustration to the right. This part of the program will have its own menu. To exit to the main menu a zero is given as input.

You are **not allowed to do any of the calculations in the constructor**. You must use at least one function/sub in each of the classes to pass. Call the function in each class Start or something similar. You may use the constructor (as intended by a constructor) to initiate class values, but in this assignment you do not need to use class variables, local function/sub values are fine. The more you divide the code in separate functions the better. Smaller functions (and more generic) help providing reusability of code.

```

Program menu
-----
1. Whole numbers with for
2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
0. Exit
Enter choice:4
-----
Work schedule
-----
1. Weekends to work
2. Nights to work
0. Exit
Enter choice:1
6      9      12      15      18      21
24     27     30     33     36     39
42     45     48     51
-----
Work schedule
-----
1. Weekends to work
2. Nights to work
0. Exit
Enter choice:2
1      7      13      19      25      31
37     43     49
-----
Work schedule
-----
1. Weekends to work
2. Nights to work
0. Exit
Enter choice:

```

## 5. Advanced Requirements

1. These requirements are not mandatory but if you wish to get a higher grade (A, B or VG) than a passing grade (C or G), you need to implement these too, in addition to all the basic requirements.
2. You must divide the code in each class into more than one **Function/Sub**.
3. The code must be functionally optimized (not speed wise, design is more important than speed in this course) e.g. where it is possible to reuse a function or part of the code you must do it.
4. You need to have a class called **Input** that contains shared functions (functions that can be used without creating an instance of a class) that handles reading of integers and floating point numbers, e.g. two functions called **ReadIntegerConsole** and **ReadDoubleConsole** (if you are doing a windows form application then add the functions **IsInteger** and **IsDouble** instead to validate the input in the form before using it). This class is perfect to put shared functions later on in the course that validates input from the user. Keep this class during all assignments and add validation functionality to it.

Since this is one of the first assignments you will be given the code for this further down. Later on during the course you will have to do some research by your own to be able to fulfill the advanced requirements.

5. If the value entered is invalid the application must not break (e.g. a number is expected but something else is entered). Show an error message and then continue. This is best handled in the Input class above.

## 6. Step by Step Instructions

### *A work plan*

When developing applications it is very important to be organized. Think of what to do and divide it into small parts. A project can be hard to grip but when you start to divide it into smaller and smaller parts and steps it gets easier to grip. In this assignment we will help you out with the planning and the steps but in the coming assignments lesser and lesser instructions will be given.

The first step is to create a so called solution and a new project in Visual Studio. A solution is a container for projects and can contain one to many projects. The solution information is saved in a file with the **.sln** extension.

A project in Visual Studio manages classes, files, components and resources that are needed to make a program. The project information is saved with the extension **.vbproj** and it is this file you open when you load an existing Visual Basic project.

When you create a new Visual Basic project a solution is automatically created for you so for the time being you do not need to focus on the solution. If there is no solution file for a project then it will be automatically created when you open the project.

A tool or application like Visual Studio that integrates many development tools in one application is called IDE (Integrated Development Environment).

Start Visual Studio and choose "Create Project". Choose Visual Basic and Console application. Choose a folder to save to (or if you are happy with the default folder just check the path to the folder so you remember it later on) Write a name for your project, Assignment2 for instance, and press OK.



1. Visual Studio now prepares a start module called Module1.vb. You can see it in the solution explorer that is normally located to the right of the screen. You can also see the project named Assignment2 and above it the solution also named Assignment2.
2. Rename Module1.vb to **MainModule.vb** in the solution explorer by right clicking the file name and choosing rename. Read the pop up window and answer yes. In the method Main we will later on create an object of the class Menu.
3. The next step will be creating the Menu class.

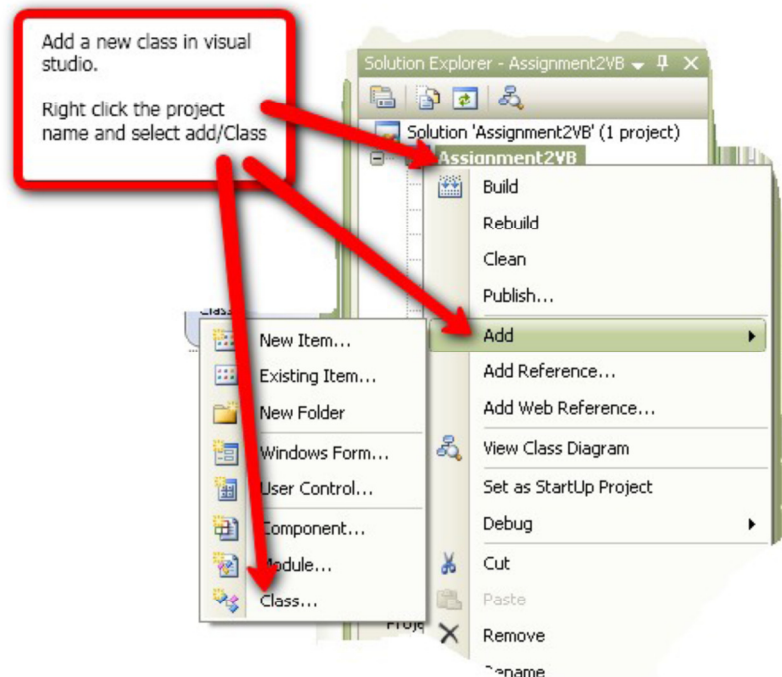
## The Menu Class

The menu class will do the following at first:

- Display the menu
- Read input from the user and save it in class variables
- Display the message "Your selection was xx, and that is not implemented yet".

We will then add functionality to it after we have completed the other required classes. The menu class will show the menu with the five available choices 0-4 as illustrated before. The application will then wait for input from the user. It will then create an object of the class that you have written for that menu choice and call the Start function for that object.

1. Right click on the project in the Solution Explorer and select Add, then Class.
2. Name the class and press Ok.
3. Write a method in the menu class with the name Start. Write the base for a loop inside it. The loop should show the menu, wait for input and then show the menu again. The menu should iterate until the user writes a zero. The method can be written as a Sub or a Function. The menu should call another method that is responsible for writing to the console called WriteMenuText. Further down a piece of code of the menu class is shown. Don't forget to save often!
4. Write the method WriteMenuText. The method should display the menu and the input prompt.



```

''' <summary>
''' The main menu class.
''' Hadles the main functionality of the program.
''' </summary>
''' <remarks></remarks>
Public Class Menu
    ''' <summary>
    ''' The main loop of the program
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub Start()

        Dim choice As Integer = -1

        While (choice <> 0) 'loop until the user writes a 0
            WriteMenuText() ' Display the meny

            ' Read the user input (0-4)
            ' We assume that the user writes a correct input at the moment
            choice = Convert.ToInt32(Console.ReadLine())

            Select Case choice
                Case 0
                    'Do nothing. The 0 input is handled by loop
                    '(needed or else the case else will be executed)
                Case 1
                    'run WholeNumbersForAdd
                    Dim c As WholeNumbersForAdd = New WholeNumbersForAdd
                    c.start()
                    Exit Select ' not needed but are recommended
                Case 2
                    'run FloatingPointsNumberWhileAdd

```

5. It is time to run the application for the first time. Since you have not written the function **WholeNumbersForAdd** you need to uncomment the two lines of code above in the case 1. Instead write a message to the user that states that this selection is not available yet.
6. Go back to the **MainModule** and create an object of the Menu (even though it is not


```

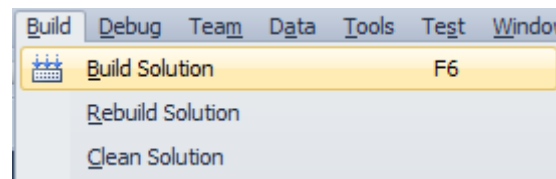
Module MainModule
    ''' <summary>
    ''' Starts the application
    ''' </summary>
    ''' <remarks></remarks>
    Sub Main()
        Dim menu As Menu = New Menu
        menu.Start()
    End Sub
End Module

```

completely finished yet). Call the Start function of the object (see code below).

7. Build the application (compile and link) by selecting Build Solution via the Build menu.

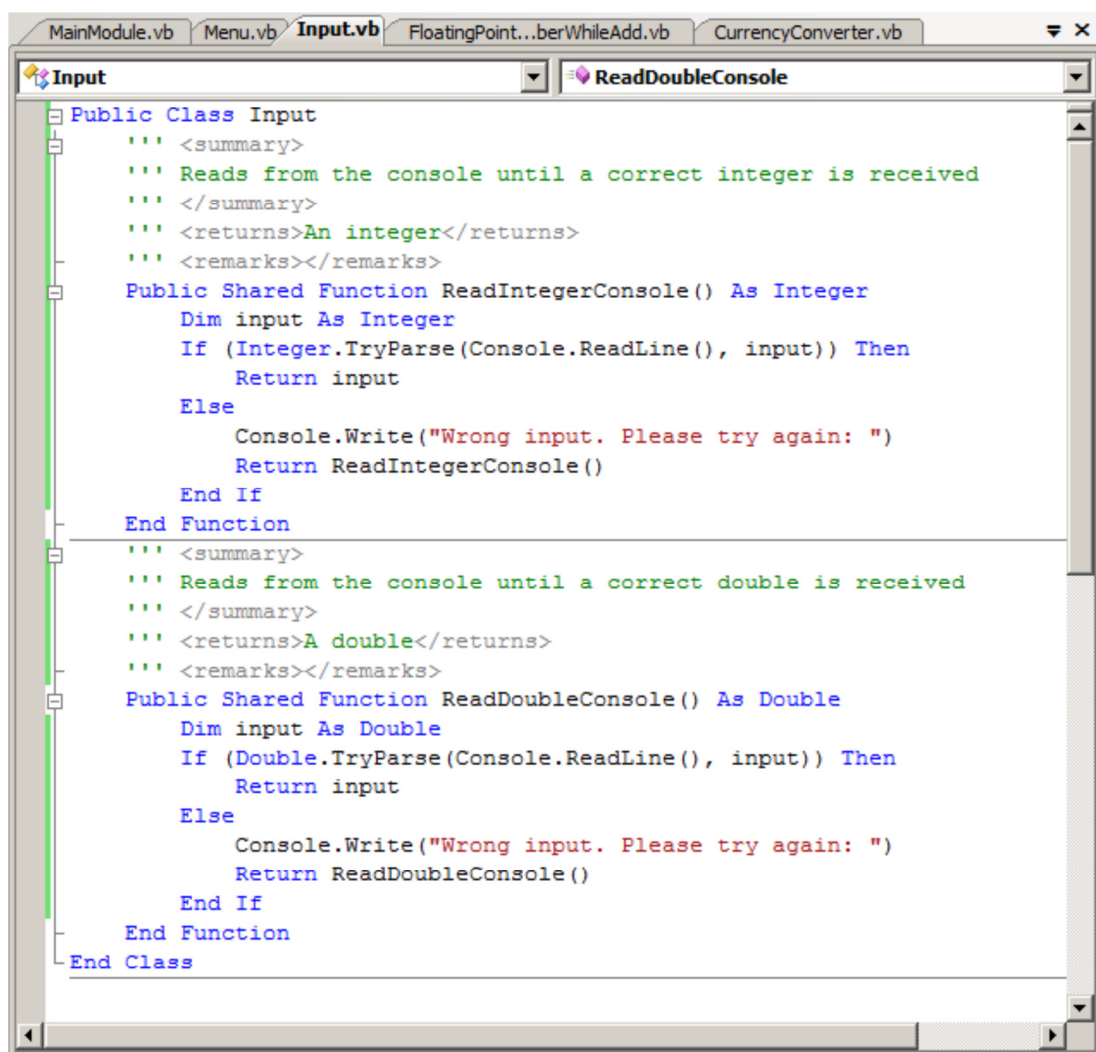
- Run the application by pressing the icon  or by pressing F5 if the build was successful. See to it that the program works well so far. Test that the program only exits when the user enter a 0. When you are happy with the functionality, continue to the next step.



## The Input Class

As you might have noticed, if the user enters a text instead of a number the application hangs. This is very annoying for the user and gives a bad user experience. So before converting the text to a number it is always better to check that it really is a number. Since this is a bit advanced you are given the code needed to do this. Use of the Input class is not mandatory.

- Create a new class called Input.



- Add the two functions that are shown above.

The functions in the picture on the previous page are of the type shared. This means that you do not need to make an instance of the class to use them.

```
Public Sub Start()

    Dim choice As Integer = -1

    While (choice <> 0) 'loop until the user writes a 0
        WriteMenuText() ' Display the menu
        ' Read indata (0-4)
        choice = Input.ReadIntegerConsole()
    End While
End Sub
```

3. Change the Menu class to use the **ReadIntegerConsole** function instead of reading directly from the console as shown below.
4. Build and run the application again to test it. Now we have a class that guaranties that we will get the correct input when we ask for either an **Integer** or a **Double**. We will now use these functions in the rest of the application when we need to read a number value.

### The WholeNumbersForAdd Class (for-statement)

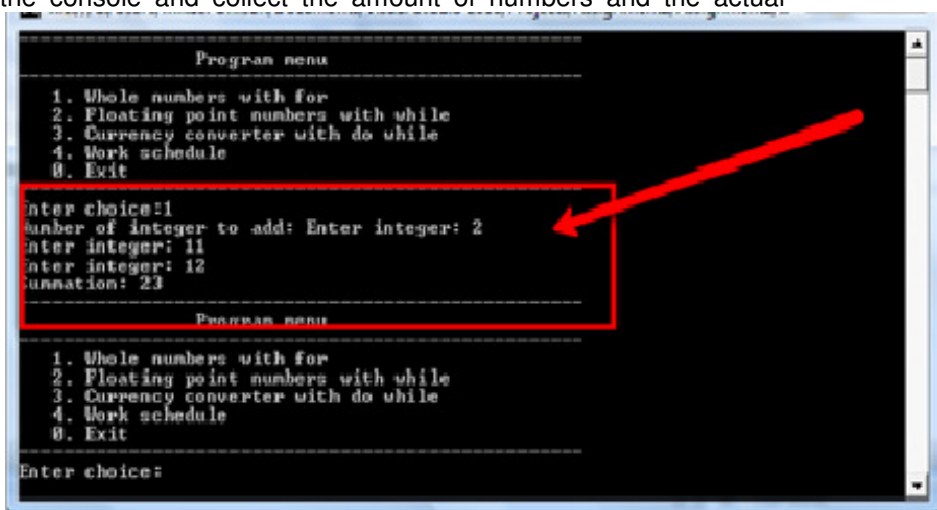
In this part we are going to work with the menu choice number 1 and get it to work. The class is going to summarize a couple of whole numbers that is collected from the console window. The class first asks for how many numbers to summarize and then reads that many integers from the console.

1. Create a new class with the name **WholeNumbersForAdd** and save it. The class will be responsible for:

Read the user input from the console and collect the amount of numbers and the actual numbers.

Calculate the summation.

Display the result.



2. The code that follows further down can be used as a template for this class and the following classes.

- Copy the code and complement where indicated in the code. It is a requirement for this class to use a **For-loop**. When you know how many iterations is needed on beforehand the For-loop is the most suitable loop to use.

```
Public Class WholeNumbersForAdd
    'Variable declaration (aka fields, instance variable, class variable or attribute)
    Private numberOfInputs As Integer 'num of values to add
    Private summation As Integer 'the result of the summation

    ''' <summary>
    ''' Starts the WholeNumbersForAdd
    '''
    ''' Public sub-method that performs the whole process
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub start()
        WriteProgramInfo()
        ReadInput()
        SumNumbers()
        WriteResult()
    End Sub

    ''' <summary>
    ''' Writes the welcome text to the console
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub WriteProgramInfo()
        Console.WriteLine("***** Summation of whole numbers *****")
        Console.WriteLine("                using a for statement")
        Console.WriteLine()
        Console.WriteLine()
        Console.Write("Number of integer to add: ")
    End Sub

    ''' <summary>
    ''' Reads the amount of numbers to add
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub ReadInput()
        numberOfInputs = Input.ReadIntegerConsole()
    End Sub

    ''' <summary>
    ''' Sub-method that sums up the numbers as they are read
    ''' and the result is stored in the instance variable sum
    ''' </summary>
    ''' <remarks></remarks>
    Public Sub SumNumbers()
        Dim index As Integer 'counter variable
        Dim num As Integer 'stores the value that the user gives

        'for-statement that iterates
        For index = 0 To numberOfInputs - 1 Step 1
            'ADD YOUR CODE HERE
        Next
    End Sub

    ''' <summary>
    ''' Writes the result
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub WriteResult()
        'ADD YOUR CODE HERE
    End Sub
End Class
```

- When you have completed the code above you need to change the Menu class to initiate an object of this class and then run the Start method, as it is written in the example code above.

### The FloatingNumbersWhileAdd Class (while-statement)

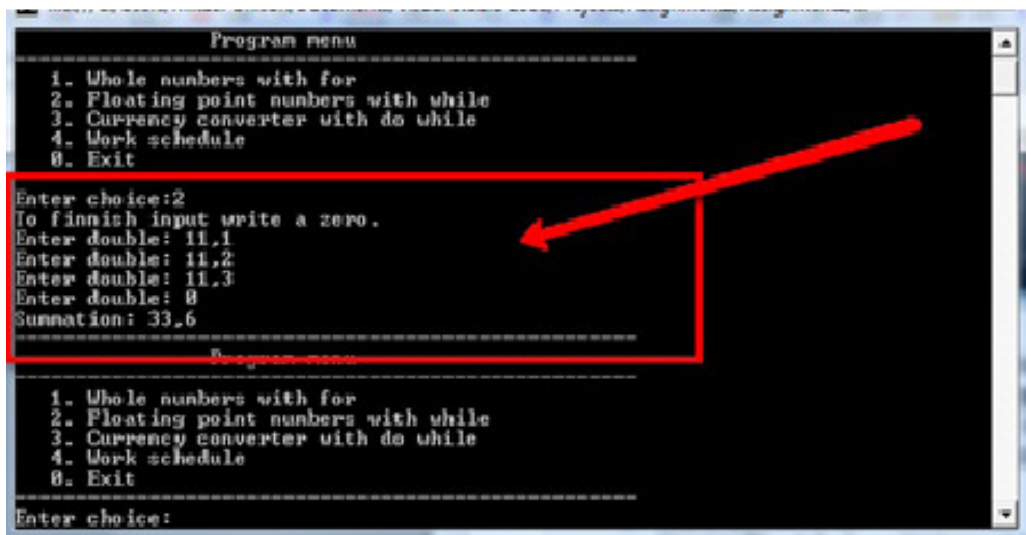
In this section, the program lets the user feed in values that can be real numbers or integers. The program does not require the user to specify the number of values to be read. It stops reading when the user writes a zero value as shown in the sample program below. A while statement is the best choice when the number of iterations is not known.

1. There are two important requirements in this part of the assignment:

Use double data type for the numeric values.

Use a while statement to perform the iteration.

2. Create a new class FloatNumbersWhileAdd.vb.



3. **Hint:** When you compare two floating point numbers, it would be a good idea to compare the round-off values. In the example below, comparison is done to a 7 decimal positions.

```
. number = ReadInput()
```

```
If ((Math.Round( number, 7) == 0.0))  
Then  
    'code  
End If
```

```
Public Class FloatingPointsNumberWhileAdd
    Private summation As Double
    ''' <summary>
    ''' Starts the WholeNumbersForAdd
    ''' </summary>
    Public Sub start()
        WriteProgramInfo()
        ReadValuesAndSumNumbers()
        WriteResult()
    End Sub

    ''' <summary>
    ''' Writes the welcome text to the console
    ''' </summary>
    Private Sub WriteProgramInfo()
        'ADD YOUR CODE HERE
    End Sub

    ''' <summary>
    ''' Reads the data and
    ''' </summary>
    Private Sub ReadValuesAndSumNumbers()
        Dim done As Boolean = False
        summation = 0
        While (Not done)
            'ADD YOUR CODE HERE
            'Read a number from the console (Input.ReadDoubleConsole)
            'Check if the number is 0
            'If yes done=true (end the iteration)
            'otherwise add it to the summation
        End While
    End Sub

    ''' <summary>
    ''' Displays the result of the summation
    ''' </summary>
    Private Sub WriteResult()
        'ADD YOUR CODE HERE
    End Sub
End Class
```

Math is a class from the System namespace. In the example above, it is assumed that a number that is like 0.0000004 is practically zero.

4. Write the code above and complete the missing parts.
5. Return to the menu class and add functionality for the 2<sup>nd</sup> menu option.

### The CurrencyConverter Class (do-while statement)

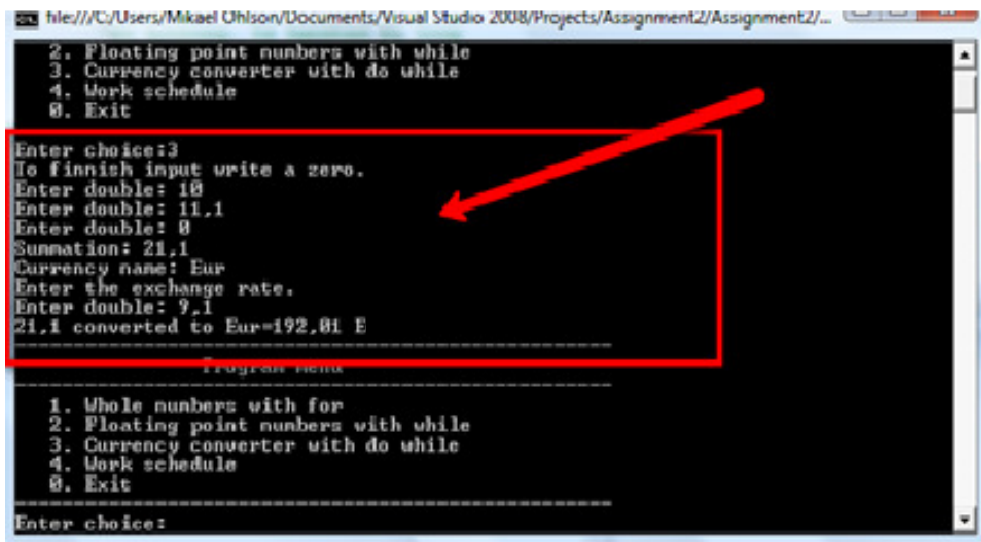
In this part we first sum up a number of values as in the previous part. The difference is that we now deal with values representing a currency. We also introduce a new facility that is to convert the accumulated amount in the local currency to a given foreign currency. The name of the currency and the exchange rate are chosen by the user.

1. There are two important requirements in this part of the assignment:

Use a [Decimal](#) data type for currency values. Use a do-while statement for the iteration.



2. Add a new class to your project and save it as **CurrencyConverter.vb**. You can now apply the copy-paste method by copying code from the previous part and paste it to this class. Change the code such that you use a do-while loop instead of while. A do-while statement is very appropriate for loops where at least one iteration is necessary.



```

file:///C:/Users/Mikael Ohlson/Documents/Visual Studio 2008/Projects/Assignment2/Assignment2/...
2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
0. Exit

Enter choice: 3
To Finnish input write a zero.
Enter double: 10
Enter double: 11.1
Enter double: 0
Summation: 21.1
Currency name: Eur
Enter the exchange rate.
Enter double: 7.1
21.1 converted to Eur=192.01 E

-----
Program menu
-----
1. Whole numbers with for
2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
0. Exit

Enter choice:
  
```

3. No code is given for this part.
4. Return to the menu class and add functionality for the 3<sup>rd</sup> menu option.

## The Time Tracker WorkingSchedule Class

With the knowledge you have gathered so far you are now going to write this class without any guidance.

The worker has to work every third weekend with start week number six and every sixth week he works night with start on week one. You must make a program that shows him when he is supposed to work weekends and then when he is supposed to work night. This should be divided into two parts but in one class (one for weekends and one for nights). See the illustration below. This part of the program will have its own menu. To exit to the main menu a zero is given as input.

Write a class that will present the user with the year schedule when the user chooses the 4<sup>th</sup> menu option. When the user types 0 he will be returned to the main menu otherwise the working schedule menu should loop over and over again.

You are free to choose what kind of loop to use ([For](#), [While](#), [Do-while](#)) but you must motivate why you chose that loop in a comment in the class.

### Remember:

A For-loop is used when the number of iterations is known on forehand.

A while loop is used when the number of iterations is not known. The loop will continue until one or more expressions are valid. The expression(s) is tested before the loop runs.

A do-while loop is used during the same conditions as a while loop. The difference is that a do-while loop is executed at least once. The expression(s) are validated after the first iteration.



```

file:///C:/Users/Mikael Olsson/Documents/visual studio 2008/Projects/Assignment2/Assi
-----
Program menu
-----
1. Whole numbers with for
2. Floating point numbers with while
3. Currency converter with do while
4. Work schedule
0. Exit
-----
Enter choice:4
-----
Work schedule
-----
1. Weekends to work
2. Nights to work
0. Exit
-----
Enter choice:1
6      9      12      15      18      21
24     27     30     33     36     39
42     45     48     51
-----
Work schedule
-----
1. Weekends to work
2. Nights to work
0. Exit
-----
Enter choice:2
1      7      13      19      25      31
37     43     49
-----
Work schedule
-----
1. Weekends to work
2. Nights to work
0. Exit
-----
Enter choice:

```

## Some hints

A special way of formatting can be used with both Console.Write and Console.WriteLine which gives a lot of possibilities.

```
Console.WriteLine("{0:C} can be exchanged to {1:f2} {2} with the exchange rate {3:C}/{4}.", _
    sum, foreingAmount, currencyName, Rate, currencyName)
```

The expression {0:C} works like this: The compiler removes the brackets {}, gets the value from the variable 0 in the list and replaces the 0 with it. C tells that the variable is a currency. The runtime compiler adds the symbol for the local currency defined in Windows Regional Settings (Control Panel), f2 tells the compiler that the number is a floating point number and that it should be rounded off to two decimal places.

In Visual Basic, you can only write one statement on a line. If you want to write a part of the statement on the next line you must write the character '\_' (with a space before) at the end of the line as in the example above. If you wish to have two statements on the same line (**not** recommended), then you should use a ":" character between the statements.

Some more hints:

number += 1 is the same as: number = number + 1

strText = "James " & "Bond" will give the result "James Bond". The character '&' concatenates strings. You can also use it as follows:

```
strText = "James "
```

```
strText &= "Bond"
```

and the result will be the same as above "James Bond".

## Good Luck!

*Programming is fun. Never give up. Ask for help!*

*Farid Naisan*

Instructor and Course Responsible