



MALMÖ HÖGSKOLA

Programming in C# I

Assignment 3

Windows Forms Application

Calculators

Mandatory

[Farid Naisan](#)

University Lecturer
Department of Computer Science
Malmö University, Sweden

Assignment 3: Calculators

1. Introduction:

So far, we have been working with Console applications to learn and test the basic syntax and structures of the C# language. It is now time to move over to more practical cases and create applications having graphical user interface (GUI). From now on, all our applications, examples, exercises and assignments will be GUI-based. Using Visual Studio (VS), we are going to create a Windows Forms Application.

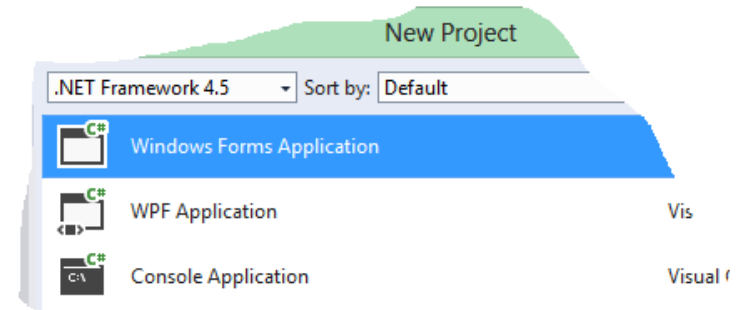
In our Console applications, we had classes which also took care of communications with the user because we were not in a position to make the problem more complicated. A more object-oriented way is to let a class, a user interface class (UI) be responsible for all user interactions. However, the time is mature now and as we begin creating a form object (**MainForm**) that would make our main user-interface, we will decouple all other classes from interacting with the user and let the form class be solely responsible for that.

Before starting with this assignment, it is important to note the following:

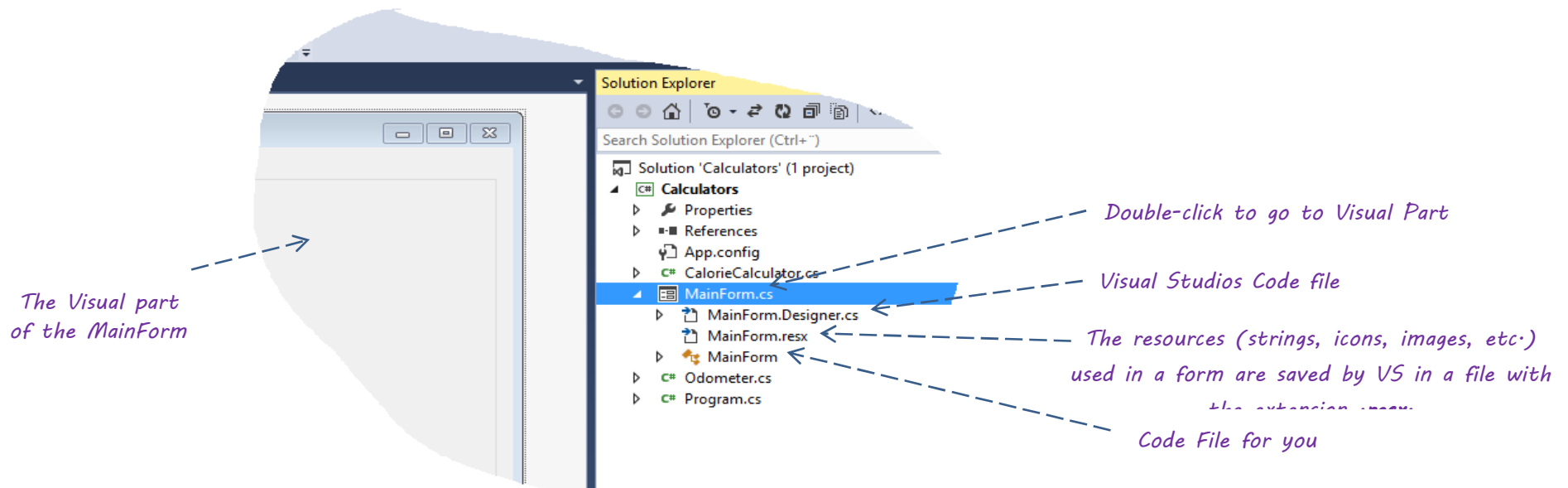
- a. Analogous to routines in an organization where not every staff-member is (or should be) having dialogs with customers, we let only one or more Form objects (GUI) act as a link between the user (one who runs the program) and the rest of the programs (objects of other classes). Classes other than forms should be designed such that their methods serve the forms. The communication between them should take place through methods. In this assignment we will be using at least one Form (**MainForm**) class as both the starting object and the user-interface to handle all interactions with the user.
- b. Remember that a class should normally not know its clients, i.e. other classes that want to use the services (methods) of this class. As an example, if you have a class **Product** with data and methods, this class should be programmed such that it should be independent of the classes that may be using it. It should be able to be used by **MainForm**, as well as by another class (**ProductManager**, if you had one such class) which are in need of services that the Product class makes available. The Product class in turn can make use of other classes to accomplish its tasks.
- c. In our Console applications, we had a class with the Main method through which our program started at run time. This is still true for Windows Form applications too. However, working in VS, you normally do not need to do any programming in this class. VS generates this class and

also all the programming code necessary in this method. You begin your work directly with the start-up form which VS creates (Form1) for you when you create a Windows Form Application.

- d. When you create a GUI application in VS, select Windows Forms Application (instead of Console). VS prepares the project and also creates a start-up form (Form1.cs). Change the name of this file to MainForm.cs (right-click on the file name in Project Explorer in VS and select **Rename**). VS asks you if you wish to rename also the related class (Form1); accept the offer and begin working with this file. VS creates also a file named Program.cs. It is will be this file in which VS codes the Main method.



- e. Every form has a visual part and a code part. You use the visual part to design your GUI, using the **Toolbox**, the **Properties** box and other graphical tools available in VS, and the related code file to program for the GUI.



- f. VS creates a file xxx.Designer.cs, where xxx is the name of the related Form (MainForm.Designer.cs). VS uses this file to do its initializations and write (generate) all code behind the visual design that you do on the form.

2. Objectives

A programmer should be ready to program any type of calculations provided a specification is available. In your programming life, you will undoubtedly come across problems that require arithmetic and mathematical calculations. In this assignment we will work with a number of simple arithmetic calculations. The main objectives of this assignment are:

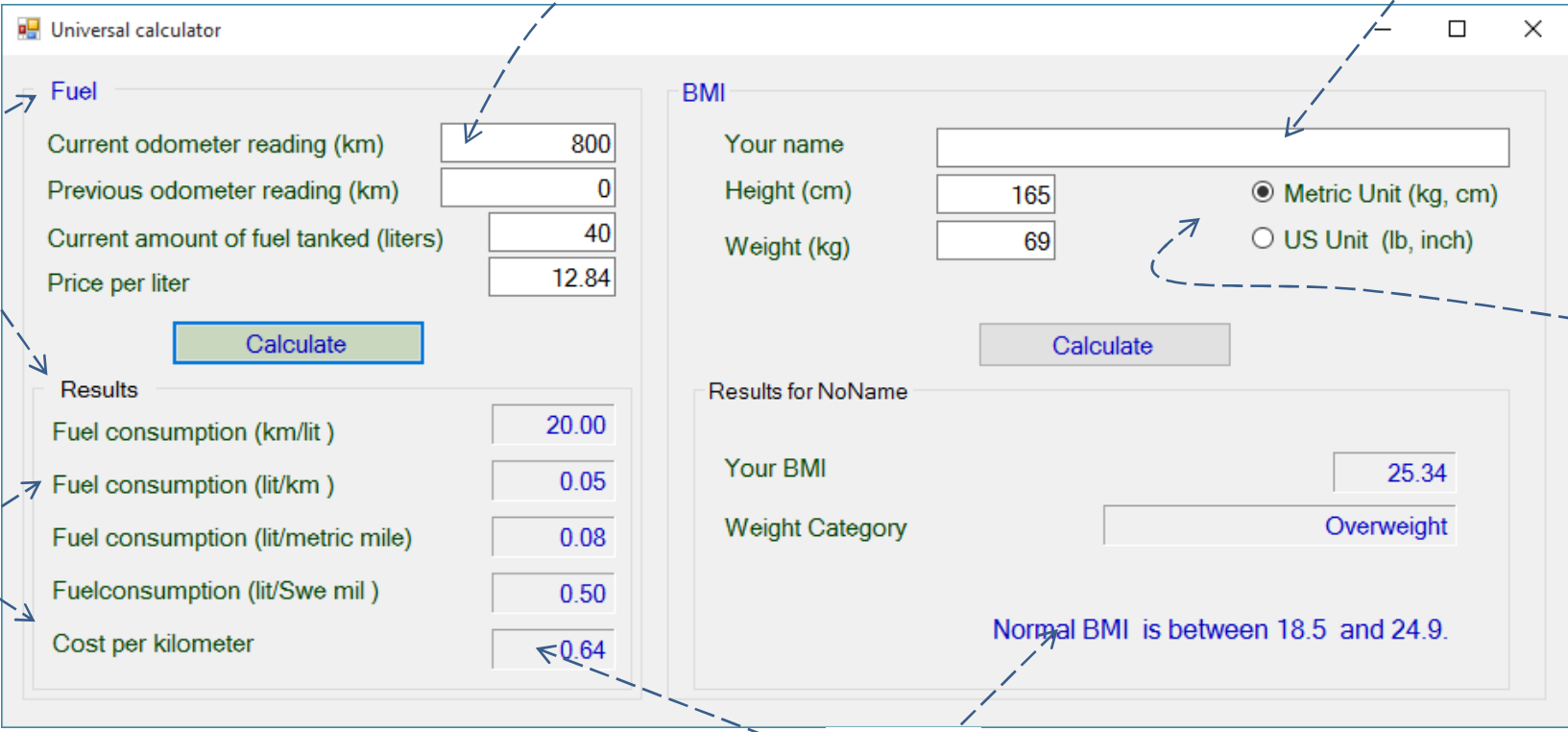
- To begin writing Windows Forms based desktop applications with graphical user interface (GUI) using some of the most common Windows Form controls.
- To get input from different Windows Form controls and save them in instance variables.
- Perform input validation.
- Use getter and setter methods as well parameterized methods and methods with return value to establish communication between objects of classes.

From this point on, we will be considering separation of concerns as discussed earlier. More precisely, we will try to separate the presentation of data from the logics that manipulate input data and process output. Further details are given under requirements later in this document.

3. Description

In our daily life, we need to have calculated data to make decisions. There are many areas that a simple calculator facilitates the computation of values greatly. Examples of calculators are, geometric calculator, date and time calculator, age calculator, marriage calculator and hundreds of other types. In this assignment we create an application with two calculators, **Fuel Calculator**, **BodyMassIndex (BMI) Calculator** and **Calorie-Calculator (BMR, VG part)**. You can then add more calculators to your application in the future (using **TabControl** for more space).

With today's high fuel prices, the economy of driving a car affects the economy of the family considerably. There are certainly a lot of people who don't know how to calculate the fuel consumption for their cars. It would therefore be helpful to have a Fuel Calculator program that takes necessary input from the user and calculates some useful data. In addition most people should think of their weight and the calories they take in daily to keep themselves healthy. Our BMI and BMR Calculators will help them control their weight.



Fuel

Current odometer reading (km)

Previous odometer reading (km)

Current amount of fuel tanked (liters)

Price per liter

Calculate

Results

Fuel consumption (km/lit)

Fuel consumption (lit/km)

Fuel consumption (lit/metric mile)

Fuelconsumption (lit/Swe mil)

Cost per kilometer

BMI

Your name

Height (cm)

Weight (kg)

☒ Metric Unit (kg, cm)

☐ US Unit (lb, inch)

Calculate

Results for NoName

Your BMI

Weight Category

Normal BMI is between 18.5 and 24.9.

This is a screen dump of a program execution session. You can also see which Form Controls are used in this GUI.

4. Formulas for fuel consumption calculator

To make use of the calculator, the value should be noted from two successive full tanks. When tanking the first time, tank the car fully and then read the kilometers on the odometer of the car. This becomes the previous reading. Most cars have a separate odometer that can be set to zero to measure certain driving distances, in which case the previous reading is 0. When you tank the car next time, tank the car fully again, read the odometer and note the amount of fuel that filled the tank. Make also note of the price per liter if you wish to calculate the cost of fuel per kilometer.

Use the following formulas to calculate results shown on the GUI (above figure) for the fuel calculator: See also <http://www.calculator.net/gas-mileage-calculator.html#> for checking your calculations.

4.1 Consumption liter per km: **litPerKm**: How many liters the car burns for each km?

Distance: **km** = **current reading** – **previous reading** (*)
litPerKm = **fuel amount** / **km**;

4.2 Consumption: **km per liter: kmPerLit**: (this value is used in other formulas): How many kilometers the car be driven for a liter?

kmPerLit = **km**/fuel amount
where **km** calculated in (*) above.

4.3 Consumption **litPerMetricMile** - Values in metric mile:

const double **kmToMileFactor** = 0.621371192;
litPerMetricMile = **litPerKm** (as above) / **kmToMileFactor**

4.4 Consumption per Swedish mil – Values in Swedish mile (mil):

literPerSwedMil = **litPerKm** * 10;

4.5 Cost per km: Cost per km = **litPerKm** * unit price (price can be in any currency)

5. Formulas for the Body-Mass Index (BMI) Calculator

Body mass index (BMI) is a measure of body fat based on height and weight that applies to adult men and women over 18. It does not measure the fat but gives a good estimation of a healthy body weight. You can check the results of your calculation using the calculator at http://www.nhlbi.nih.gov/health/educational/lose_wt/BMI/bmicalc.htm.

5.1 BMI values:

$$\begin{aligned} \text{BMI} &= \text{weight in kg} / \text{height}^2 \text{ (in m}^2\text{)} && \text{(Metric Units)} \\ \text{BMI} &= 703.0 * \text{weight (in lb)} / \text{height}^2 \text{ (in inch}^2\text{)} && \text{(U.S. Units)} \end{aligned}$$

where $\text{height}^2 = \text{height} * \text{height}$

5.2 The above formulas are standard for adults not taking the age into consideration. The World Health Organization's (WHO) has recommended a body weight based on BMI values for adults, as summarized in the following table. It is used for both men and women, age 18 or older. For more information see: http://apps.who.int/bmi/index.jsp?introPage=intro_3.html.

BMI	Nutritional status
Below 18.5	Underweight
18.5–24.9	Normal weight
25.0–29.9	Overweight (Pre-obesity)
30.0–34.9	Obesity class I
35.0–39.9	Obesity class II
Above 40	Obesity class III

6. Features and requirements for a Pass (G) grade

This course is for beginners and does not require any programming background. This and all other assignments are structured for beginners (although they might sometimes get a little hard). The requirements specified below are minimum requirements addressing the beginner students but we are open to more advanced solutions in case you have better ideas that provide better solutions with well-structured and object-oriented code. Keeping this in mind, you do not have to follow the requirements step by step. As a recommendation, if you are already familiar with Windows Forms and wish to try something new (and more challenging), you may try creating a Windows Presentation Foundation (WPF).

7. Functional Requirements

- 7.1 At program start, the Form should be clear of all design-time texts such as Label1, etc. (Form1 as the title of the form is not accepted). All input boxes (textboxes) and output controls (labels) should be empty at program start. Input boxes may have default values (for example previous odometer reading can be set to zero).
 - 7.2 The application should **control the user input** so it does not crash or give unexpected output for invalid input. As soon as an input value is not valid, do not continue with the calculation (in the code) and let the user give valid data.
 - 7.3 The user is to be given a notification when the input data is invalid. Use a **MessageBox**.
 - 7.4 The previous reading should be equal or greater than zero (not negative).
 - 7.5 Current odometer reading must be greater than the value of the previous reading.
 - 7.6 The price should not be negative (can be zero for free gas). Note: If you don't specify any currency (as in the GUI example above), your program will work for any type of currency.
 - 7.7 Weight and height should be floating-point values greater than zero. Assign a default value for the name ("No name") if the user does not provide one.
 - 7.8 All results should be correct. There are several sites in the Internet that you can check. Use the ones given above.
-

8. Structural Requirements

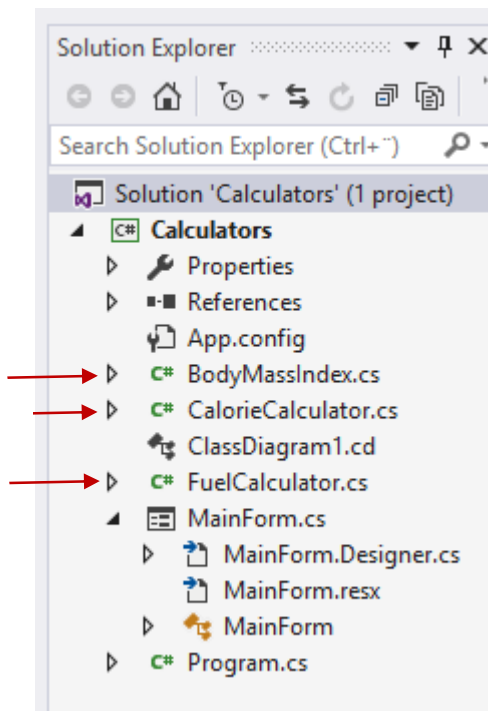
8.1 General:

- 8.1.1 Textboxes should not be used for output. Use Labels instead for output and other read-only information. Do not make a textbox work as a label by disabling the control.
- 8.1.2 All instance variables should be declared as **private**. Public instance variables are strictly forbidden.
- 8.1.3 Every class and every method in the class should contain a brief but informative documentation in form of comments. Shortage of time is not an acceptable excuse.
- 8.1.4 Use appropriate names for all identifiers (variables, methods). Short names such as x, y Fnc are not a part of good programming style and should be avoided. Make sure to rename all VS's default names such as Application1, Project1, Label1, Form1, etc. and change them to proper names.
- 8.1.5 Methods should not be long. You can always break down a long method into smaller ones.
Each method should perform only one task (not two or more different tasks).

For more details of quality requirements and recommendations, see the document Quality Standards and Guidelines, available on Its L.

8.2 Calculator Classes

- 8.2.1 Write one class for each calculator.
- 8.2.2 Every calculation should be coded as a separate method.
- 8.2.3 Wherever possible, call other methods in the same class instead of writing same code twice.



- 8.2.4 The calculator classes should **only** have fields (instance variables) for saving **input**. No output variables are to be used; see the class diagrams at the end of the document.
- 8.2.5 All output should be provided through the return values (or method parameters).
- 8.2.6 Write getter and setter methods in the calculator classes to give read and write access to instance variables (e.g. input values).

MainForm Class.

- 8.2.7 **MainForm** should **only** use an instance of each calculator class and a field for the name (as it does not belong to any calculator class) as its fields. No other instance variables should be used (see the class diagram later in this document).
- 8.2.8 Only and Only **MainForm** should work with the user interface, the form itself and all the controls on it.
- 8.2.9 Other classes (the calculator classes) should not have anything to do with the **MainForm** or its components. Instead, the **MainForm** will be using the calculator classes:
 - MainForm declares and creates an instance of each calculator.
 - MainForm uses the setter methods of the instance to set (save) values from the input controls (textboxes, option buttons, etc.) in the instance's private fields (instance variables).
 - MainForm calls the instance's methods to calculate and provide an output.
 - MainForm updates the GUI by displaying the output using the dedicated controls.
- 8.2.10 Write a method **InitializeGUI** and use it for clearing input/output controls and setting default values (default RadioButton).
- 8.2.11 It is OK to replace **RadioButtons** used for activity levels by a **ComboBox**. (ComboBoxes are included in the next module).

Note: Each set of **RadioButtons** must be encapsulated inside a container component such as a **GroupBox**. Otherwise different **RadioButtons** sets contained in the same component will be considered as one set. To separate the Female-Male buttons from the Activity Level buttons, you need to put at least one of the sets inside a separate **GroupBox** control – see the GUI image given earlier.

```
public partial class MainForm : Form
{
    //Declare and create an instance of the Fuel Calculator
    private FuelCalculator carMilage = new FuelCalculator();

    //Declare and create an instance of the BMI Calculator
    private BodyMassIndex bmiCalc = new BodyMassIndex ( );

    private string name = String.Empty;

    //(Only for V6) Declare and create an instance of the CalorieCalculator
    private CalorieCalculator bmrCalc = new CalorieCalculator();

    //Constructor - same name as the class (MainForm) and no return type, and
    //is executed when the object (of MainForm) is created. Used to do all initializations.
    //
    1 reference
    public MainForm()
    {
        InitializeComponent();
        InitializeGUI();
    }
    1 reference
    public void InitializeGUI()
    {
        rbtnFemale.Checked = true;
        rbtn1.Checked = true;

        //other initializations.
    }
}
```

8.2.12 Use **double.TryParse** and **int.TryParse** to convert text to numerical values.

```
private bool ReadInputFuel()
{
    double value = 0;
    if (double.TryParse(txtCurrReading.Text, out value))
    {
        carMilage.SetCurrentReading(value);
    }
    else
        return false;
}
```

8.2.13 Instead of reading all input in one method, organize readings in separate methods.

```
private void ReadName()
{
    txtName.Text.Trim ( ); //Delete spaces at beginning and end of the string
    if (string.IsNullOrEmpty ( txtName.Text ))
        name = "NoName";
    else
        name = txtName.Text;
}
```

In much the same way, write methods to show output:

```
private void UpdateGUIFuel ( )  
private void UpdateGUIBMR()
```

9. Formulas for Calorie Calculator (BMR) – only for VG Grade

You can skip this part if you do not going for a VG Grade.

BMR (Basal Metabolic Rate) is an equation for estimating the number of calories you need to consume each day. The calculation is done according to the Mifflin - St Jeor equation. You can use the following formulas to solve the problem:

9.1 BMR values:

$$\text{BMR} = 10 * \text{weight (kg)} + 6.25 * \text{height (cm)} - 5 * \text{age (y)}$$

$$\text{BMR Female} = \text{BMR} - 161$$

$$\text{BMR Male} = \text{BMR} + 5$$

9.2 Activity Level and multiplier factor

The values are to be taken from the table.

9.3 Calories to keep your current weight

$\text{maintainWeightCalories} = \text{BMR} * \text{activity level factor}$ (last column in above table). (BMR calculated in above, 5.1)

9.4 Lose or gain weight:

Group	Level Name	Description	Factor
0	Sedentary	Little or no exercise	1.2
1	Lightly active	Exercise 1 to 3 times a week	1.375
2	Moderately active	Exercises 4 to 5 times a week	1.550
3	Very active	Exercises 6 to 7 times a week.	1.725
4	Extra active	Hard exercise or physical job	1.9

To lose 0.5 (500 g), you need to cut off 500 and to lose 1 kg, 1000 calories from daily intake.

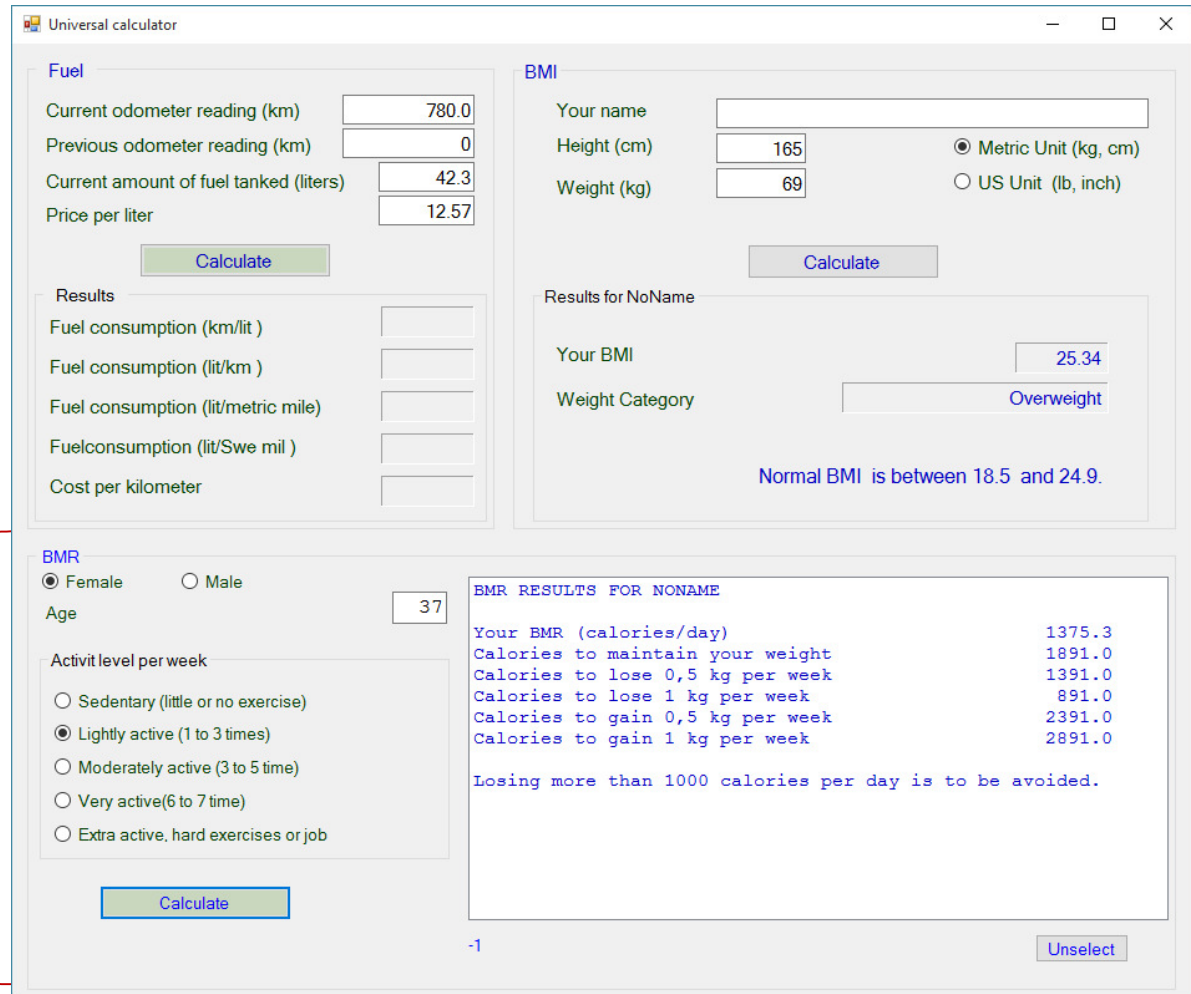
To lose 500 gr (0.5 kg) a week =
maintainWeightCalories – 500

To lose 1000 gr (1 kg) a week =
maintainWeightCalories –1000

To add 500 gr (0.5 kg) a week =
maintainWeightCalories + 500

To add 1000 gr (1 kg) a week =
maintainWeightCalories +1000

*Additional
for VG grade*



The screenshot shows a 'Universal calculator' application with three main sections: Fuel, BMI, and BMR.

Fuel Calculator:

- Current odometer reading (km): 780.0
- Previous odometer reading (km): 0
- Current amount of fuel tanked (liters): 42.3
- Price per liter: 12.57
- Calculate button
- Results:
 - Fuel consumption (km/lit):
 - Fuel consumption (lit/km):
 - Fuel consumption (lit/metric mile):
 - Fuelconsumption (lit/Swe mil):
 - Cost per kilometer:

BMI Calculator:

- Your name:
- Height (cm): 165
- Weight (kg): 69
- Unit selection: ☒ Metric Unit (kg, cm) and ☐ US Unit (lb, inch)
- Calculate button
- Results for NoName:
 - Your BMI: 25.34
 - Weight Category: Overweight
 - Normal BMI is between 18.5 and 24.9.

BMR Calculator:

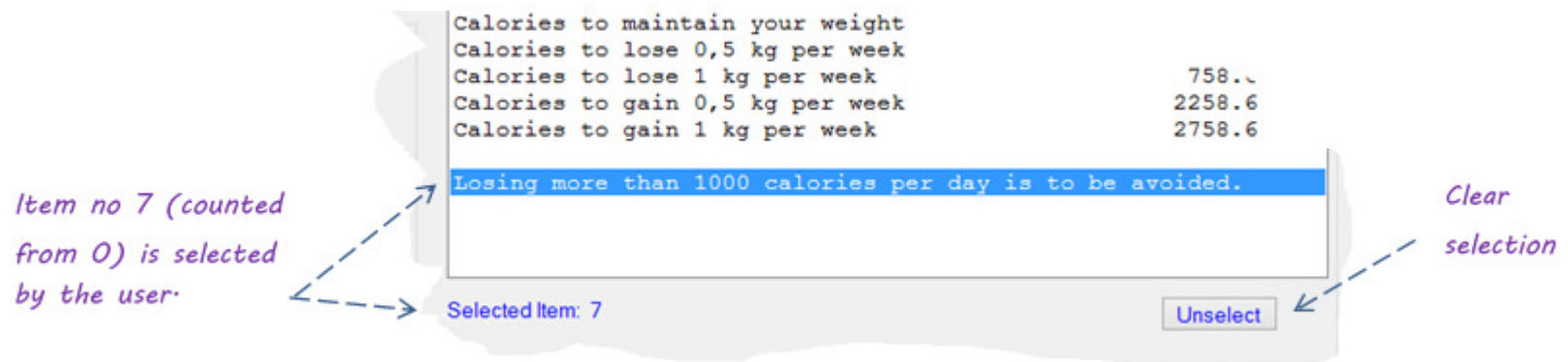
- Gender: ☒ Female, ☐ Male
- Age: 37
- Activit level per week:
 - ☐ Sedentary (little or no exercise)
 - ☒ Lightly active (1 to 3 times)
 - ☐ Moderately active (3 to 5 time)
 - ☐ Very active(6 to 7 time)
 - ☐ Extra active, hard exercises or job
- Calculate button
- BMR RESULTS FOR NONAME:

Your BMR (calories/day)	1375.3
Calories to maintain your weight	1891.0
Calories to lose 0,5 kg per week	1391.0
Calories to lose 1 kg per week	891.0
Calories to gain 0,5 kg per week	2391.0
Calories to gain 1 kg per week	2891.0

Losing more than 1000 calories per day is to be avoided.

Additional text at the bottom right: -1 and Unselect button.

- 9.4.1 For a small exercise with ListBoxes, display the item selected on the GUI, and provide a button that unselects the list box as shown below.

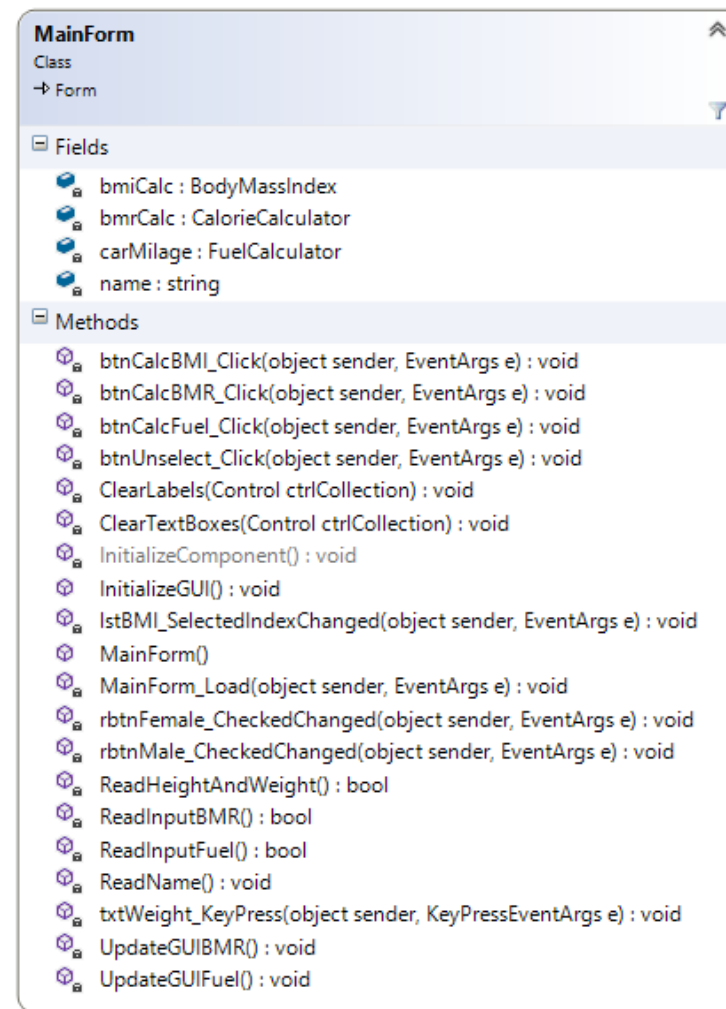
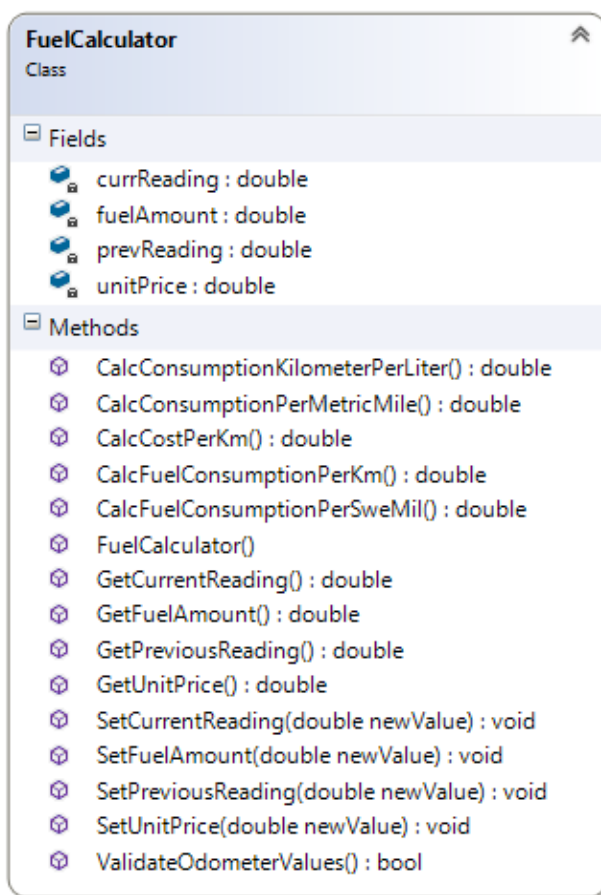


10. Help and Submission

- 10.1 Do the exercises before starting with this assignment. Use also the forum to ask questions. Test your application well before submission. **Projects that cannot be compiled or are poorly tested will be returned immediately for resubmission.**
- 10.2 If you do the VG part and your project does not meet the required quality levels, you will be given a chance to do complementary work and resubmit.
- 10.3 The class diagrams below should give you a hint on how you can construct your classes. As said before, you may write your own methods and you give them other names.

Compress all your files and folders (particularly the folder Properties) that are part of your project into a ZIP or RAR file. Upload the compressed file via the same page where you downloaded the assignment.

Class diagrams



Programming is fun. Never give up!

Good Luck!

Farid Naisan,

Course Responsible and Instructor