TP C#0

Submission

Archive

You must submit a zip file with the following architecture :

```
– rendu-tp-firstname.lastname.zip
        |– rendu-tp-firstname.lastname/
                |– AUTHORS
                |– README
                |– HelloWorld/
                        |– HelloWorld.sln
                        |– HelloWorld/
                                |– Everything except bin/ and obj/
                |– MyCalc/
                        |– MyCalc.sln
                        |– MyCalc/
                                |– Everything except bin/ and obj/
                |– ClickerGame/
                        |– ClickerGame.sln
                        |– ClickerGame/
                                |– Everything except bin/ and obj/
```

- You should obviously replace *firstname.lastname* with your name.
- The code must compile (pressing F5 must display a windows form rather than compilation errors).
- In this practical, you are allowed to use pictures and sounds as bonus, however you must keep the archive's size as light as possible.

AUTHORS

This file should contain the following : a star (*), a space, your login and a newline.
Here is an example (where $ represents the newline) :

```
* firstname.lastname$
```

Please note that the filename is AUTHORS with NO extension.

1 Introduction

1.1 Objectives

During this TP, we will approach new notions such as:

- Discover MonoDevelop
- C# language
- Windows Forms

1.2 Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop programs for Microsoft Windows as well as many other tasks on the Web. It is only available on Windows.

Visual Studio includes a code editor supporting a code completion component and an powerful integrated debugger. Visual Studio supports different languages such as C, C++, C#, VB.NET or F#.

You can download Visual Studio Community for free on *www.visualstudio.com* or Visual Studio Enterprise on Msdnaa which is on *intra-bocal.epitech.eu*.

1.3 MonoDevelop

MonoDevelop is a free IDE available on Windows, OS X and Linux. Like Visual Studio, it includes a code editor which supports auto-completion and has in integrated debugger. C# as well as other .NET languages are supported.

You can download MonoDevelop for free on *www.monodevelop.com*

1.4 The C# language

The C# is a object-oriented programming language created by Microsoft and developed by a team led by the Delphi creator, Anders Hejlsberg.

C# is a language derived from the C and C++ languages resembling the Java language. C# is used for web applications, desktop applications or web services for example.

C# is very different to what you're currently familiar with, you'll get to learn it during the rest of this year. Do not hesitate to go on MSDN.com if you are curious or need to understand it better.

2 Basic Knowledge

2.1 The basics of C#

Here are some basics which you will need to do the following exercises. You will learn more about these notions in the following weeks although you are free to discover the language on your own. You can learn about C# on MSDN. If you ever need help to understand how something work in C#, you should always think of looking for the information you want on MSDN.

- Each instruction ends with a semicolon
- Instructions are contained within blocks delimited with curly brackets
- A variable has a type and it holds a value which you can change using the = operator
- A function has a return type and may take some parameters

For this practical, you only need to use integer variables (type *int*), but you may use other types if you want to (MSDN is your friend). For this practical, you do not need to declare any function yourself but you will work on functions similar to this:

```
private void function_name(type1 arg1, type2 arg2, . . .)
{
    // This is the block of instructions of the function.
    /*
    This function does not
    return a value because its
    return type is void.
    */
}
```

Do not worry about *private* or arguments for now, it is not relevant this week.

In the previous example, what you can see in the block of instructions is the syntax for single-line comments (// comment) and multi-line comments (/* comment */). Except for the syntax, they work in the same way as in Caml.

You can declare an integer variable in the following way:

```
int my_variable_name; // my_variable_name's value is 0
```

```
if (condition) // a condition could be a == b
{
    /* Instructions executed when the condition is true */
}
else
{
    /* Instructions executed when the condition is false */
}
```

To compare 2 variables, you can use the following operators:

- a == b, to check if a is equal to b (same as = in Caml)
- a != b, to check if a is not equal to b (same as <> in Caml)
- >, >=, < and <= are the same as in Caml

Boolean expressions work too; you can learn more about them on MSDN.

2.2 The basics of Windows Forms

Windows Forms are an easy way to make a simple user interface on Windows. They are based on events. This means that when the user interacts with an element, it calls the function associated with that element.

To place an element on your form, you can simply drag and drop it from the Visual Studio's Toolbox. You can then customize your element (text, name, color, picture, font, values, ...) using the properties tab in Visual Studio.

To create the function which will be called when the user clicks on a given element, simply double click that element.

2.3 The basics of MonoDevelop

To open a project :

File -> Open -> Choose a SLN file

You will find the solution explorer on the left edge of your screen. This shows you files and directories of your project.

WindowsFormsApplication solution (1 project)
- WindowsFormsApplication: application.
  - Properties : Project Resources.
  - Reference: Libraries used in the application.
  - Form1.cs: Contains your code and displays the graphical editor.
    - Form1.Designer.cs: code generated by the graphical editor.
    - Form1.resx: Contains your code.
    - Program.cs: Program Entry Point.

You can compile and run your project by pressing F5.

3 Exercise 1: Hello World

Open the "HelloWorld" project. Pressing F5 will display a window with a button and a text field. These are called objects in C#. Objects have attributes; in this case, the button has (among others) a *Text* attribute (the displayed text), *ForeColor* (the color of the font) and *BackColor* (the background color).

Firstly, click on Form1.cs (in the solution explorer). You will write here your first C# code. Don't worry about other files or the rest of the code for now as you will learn more later.

The first function to notice is :

```
1 public Form1()
2 {
3     InitializeComponent();
4         // Write some code here.
5 }
```

It is called when the window is created. You should initialize your attributes and objects here. Do not remove or change InitializeComponent(); or the Windows Form will not work anymore.

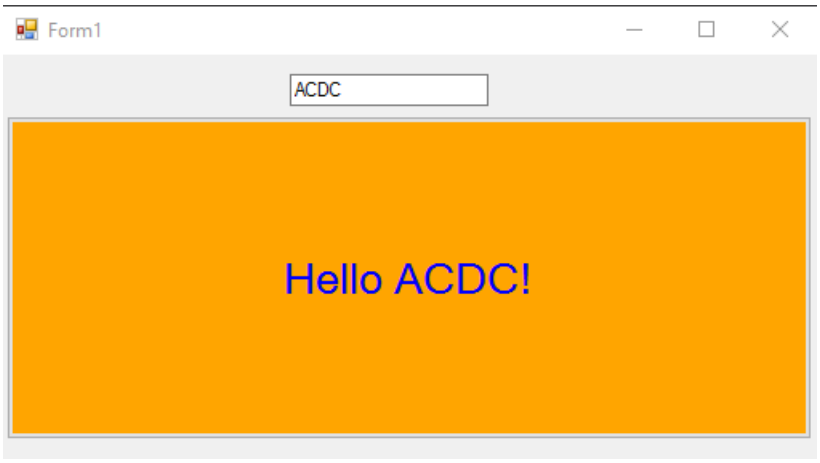
The second function concerns the button's behaviour. It is called each time the button is clicked on.

```
1 private void button1_Click(object sender, EventArgs e)
2 {
3         // Write your code here.
4 }
```

You must initialize the button's text to "Hello World!" with a green background and a red font. When clicking on the button, its text, background color and font color should change (you can choose any colors and text you want as long as they are different). However, when clicking again, the button should go back to its original state.

About objects : The button object's name is button1 and text box's name is textBox1 (useful for the bonus).
For example, you can acces the button's text via *buttton1.Text*
Bonus:

- Create a *Textbox* to display "Hello World!" on the button if the text box is empty and "Hello ACDC!" (for example) if "ACDC" is written in it.
- Be creative (add colors, sounds, pictures, features, ...).



4 Exercise 2: MyCalc

Open the MyCalc project.

In this exercise, you must create a simple calculator. You must implement the following operations : addition, substraction, multiplication, division and the modulo.
Your calculator is composed of 5 objects : 2 *NumericUpDown* objects which hold the operands' values, a *Combobox* to choose the operator, a *Button* to submit the operation and a *Label* to display the result.

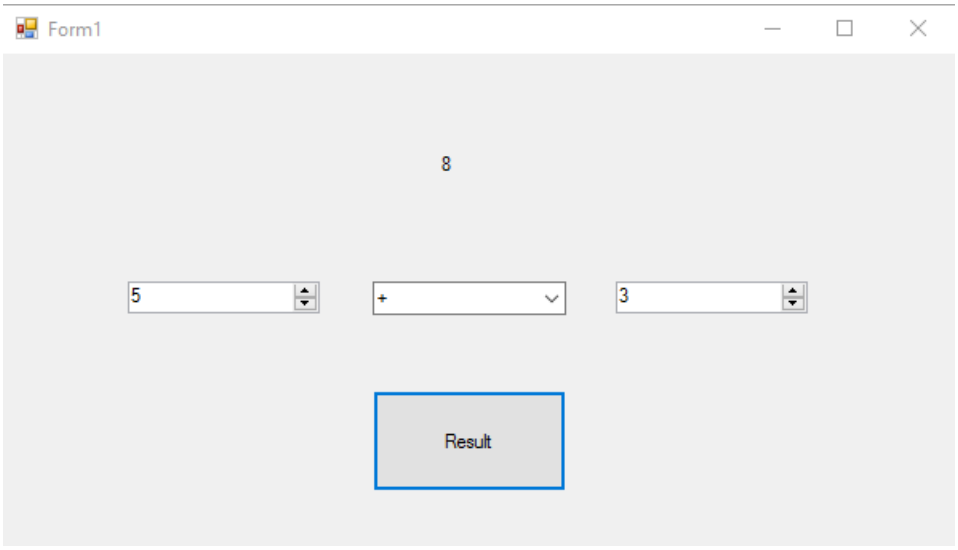
The user will enter a value in each *NumericUpDown* fields and choose an operator. Then clicking on the button should display the operation's result in the label. *NumericUpDown* objects only accept valid numbers as input so you do not have to worry about checking that the input is a number.

Using *if* and *else*, check which operator is selected and put the result in the label whenever the button is clicked.

Do not forget to handle special cases.

About objects : *NumericUpDown* objects are named "nb1" and "nb2", the *Combobox* is "comboBox1", the *Button* is "button1" and the label is *Label* est "label1". The items of the *Combobox* are : "+", "-", "*", "/" and "%" (in that order).

Hint 1: To write the result in the label, you will need to use the "ToString()" method to convert a number into a string.
Hint 2: Do not forget to go on MSDN if you have trouble with the ComboBox and the NumericUpDown objects.



Bonus: You can add some color, sounds and be creative one more time.

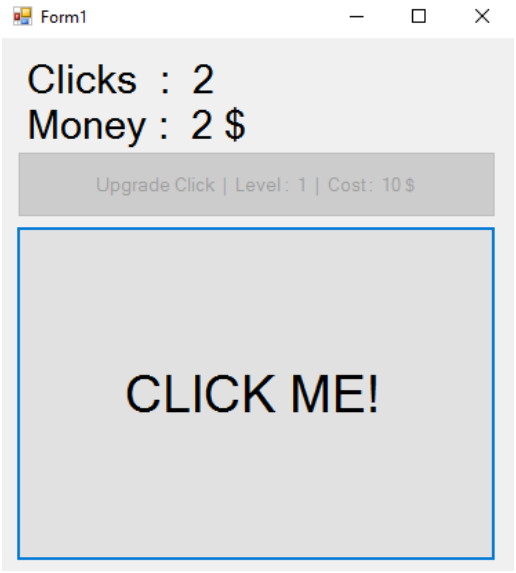
5 Exercise 3: Clicker Game

Open the ClickerGame project.

In this exercise, you will make a simple clicker game (just like Cookie Clicker but you own game). Feel free to have fun once you have finished it: be creative (colors, pictures, ...).

5.1 Part A

By the end of this part, your game will be similar to this:



You will need :

- A main button to click on. : b_clicker
- A label to display the number of clicks done on the main button. : l_clicks
- A label to display the player's money. : l_money
- A button to upgrade the player's click. : b_upgrade

Features:

- Clicking the main button should increase by 1 the count of clicks (at the top of the screen on the example).
- Clicking the main button should increase the player's money.
- The amount of money per click should start at 1 and increase by 1 for each upgrade the player buys. (See *Level* in the example.)
- The price of the upgrade should double each time it is bought. It should start at 10.
- The upgrade button should display the current money/click and the cost to upgrade.
- The upgrade button should only be *Enabled* (check MSDN) when the player has enough money to buy the upgrade.

Hint 1 : to use a variable across different function calls, you should declare it in the class before the *public void Form1()* line.

Hint 2 : To change the default values of these variables, you should change their values below the line *InitializeComponents();* in the same function.

5.2 Part B

You now have a working simple clicker game, although it is still quite boring at the moment with only one kind of upgrade. In this part, we want to add some randomness to the game using a *Random* object. Here is how to use it :

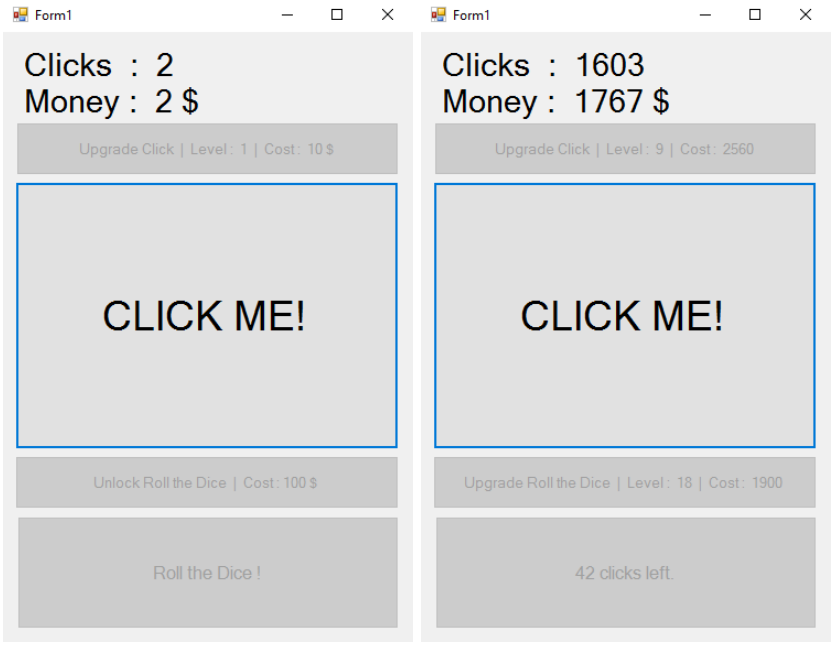
```
Random rand; // declare it

public Form1()
{
    InitializeComponents();
    rand = new Random(); // initialize it
}
/*
You can now use rand.Next() to get a random integer
Go on MSDN to learn more.
*/
```

You will need :

- A button for the new gameplay mechanic "Roll the dice !" : b_rng
- A button to unlock and upgrade this new feature : b_rng_upgrade

After this part, your game will be similar to these examples:



Features:

- The new button should be locked until the new upgrade is bought at least once (initial cost is 100).
- Each time the new upgrade is bought, the cost should increase by 100.
- Once the new feature is unlocked, a countdown should appear on the button.
- After 100 clicks on the main button, the new button becomes enabled.
- Clicking on the "Roll the dice!" button should add a random amount of money among the range [-100;cost] where cost is the current cost for the next upgrade. This means that at the level 1 of the new upgrade, it will add from -100 to 200 to the player's money.
- Clicking on the "Roll the dice!" button also locks it again for another 100 clicks on the main button.
- The player's money should not drop below 0.

Upgrading the new feature means either unlocking it (if it was at level 0) or adding 100 to both the cost for the next upgrade and the maximum amount of money which can be added by rolling the dice.

Once again, the new upgrade button should only be enabled if the player has enough money to buy it.

Don't forget that you can be creative and have fun with what the windows forms can offer you (colors, pictures, sounds, ...).

The code is the law.