Practice Assignments Lists, Comboboxes and TabControl

Quiz questions, practical assignments and

answers to quiz questions

**Version:** 1.0 **Last updated:** 10 oktober 2023

Contents

[1 Practical assignments: Windows Forms App 3](#_Toc147863895)

[1.1 Programming Assignment 1: Advanced controls 3](#_Toc147863896)

[1.1.1 Case description 3](#_Toc147863897)

[1.1.2 Possible GUI 3](#_Toc147863898)

[1.2 Programming Assignment 2: Extending assignment 1 of Intro Objects And Classes 4](#_Toc147863899)

[1.2.1 Case description 4](#_Toc147863900)

[1.2.2 Screenshots 5](#_Toc147863901)

[1.2.3 Additional features 5](#_Toc147863902)

[1.3 Programming Assignment 3: Finances 6](#_Toc147863903)

[1.3.1 Case description 6](#_Toc147863904)

[1.3.2 Screenshots 6](#_Toc147863905)

[1.3.3 Provided material 6](#_Toc147863906)

[1.4 Programming Assignment 4: Moving a car 8](#_Toc147863907)

[1.4.1 Case description 8](#_Toc147863908)

[1.4.2 Possible GUI 8](#_Toc147863909)

# Practical assignments: Windows Forms App

## Programming Assignment 1: Advanced controls

Difficulty:

The assignment covers the following learning goals:

* Working with the advanced controls

### Case description

You’re going to create simple application to allow you to experiment with the radio/check/list/combo boxes.   
Create a form such as depicted in Figure 3: POSSIBLE GUI (leave the ListBox empty); the combobox contains different string in lower case (e.g. *red*, *green*, *blue*, etc.)

Then with this GUI implement the following:

* When ‘*Add to ListBox’* is clicked, the selected item of the combobox is added to the ListBox on the right. Note that when ‘*To UPPERCASE’* is checked the string should be added in all caps; this can achieved with the .ToUpper() methods (e.g. myStringVariable.ToUpper(), "green".ToUpper(), etc.)
* When one of the radiobuttons is checked, the fruit they represent should be added to the ListBox on the right.
* When ‘*Clear ListBox’* is clicked, the ListBox on the right should be empties (i.e. no items in it).

### Possible GUI

Graphical user interface, text, application

Description automatically generated

Figure : POSSIBLE GUI

## Programming Assignment 2: Extending assignment 1 of Intro Objects And Classes

IDE: Visual Studio  
Difficulty:

The assignment covers the following learning goals:

* Create an object of a class-type.
* Working with radio buttons

### Case description

You’re going to make use of the class you created for assignment 1 to create an application representing a checking and saving bank account. A user must be able to deposit, withdraw and view the balance of which either of the bank accounts.

First open the Visual Project you created for assignment 1 and create a GUI to allow a user to select an account, deposit money, withdraw money and see the balance of the checking and savings account (see Figure 1 for an example).

Now you have to declare two variables *checkingAccount* & *savingsAccount* of the type *BankAccount*. HINT: Try to determine at which scope the variables must be declared in the code-behind of Form1.

You still need to assign the *BankAccount*-objects to the variables. Include the statements to create and assign the objects in a ‘special’ method called *public Form1* (we will explain what this is in a later lesson)*:*

public Form1()

{

InitializeComponent();

// Create and assign the BankAccount-objects below this line

}

The next step is to connect the controls of your GUI to the created objects:

* When the *Deposit*-button is clicked, the amount specified in the TextBox must be deposited in the BankAccount-object representing the checked RadioButton. For example: when the *Checking account*-RadioButton is selected an amount of 50.50 must be deposited to the object stored in the *checkingAccount* variable. HINT: You will have to call the appropriate method to deposit the amount.  
  After the amount is deposited you will have to update the *balance*-label with the new balance. HINT: You will have to call the appropriate method to get the balance.
* Implement *Withdraw*-button the same way, but now you will have to withdraw the amount.

Remember that the *WithdrawMoney*-method returns a Boolean value. Can you refactor (change) the statements of the *Withdraw*-button to show a MessageBox when the user does not have enough balance to withdraw the inputted amount?

### Screenshots

Graphical user interface, text, application

Description automatically generated

Figure : POSSIBLE GUI

### Additional features

Can you include an extra button which, when clicked, calls the method *GetInfo* and show the string in a MessageBox? You will notice that the *client* and *accountNo* are empty and 0, respectively. Do you know what statements you will have to include and where it should be put to assign a value to them?

## Programming Assignment 3: Finances

Difficulty:

The assignment covers the following learning goals:

* Define a class with its members
* Create an object of a class-type
* Declare and use the fields necessary to store data
* Define the methods to operate on the objects
* Working with ListBoxes

### Case description

Your user wants to manage his finances. There is a transaction log that belongs to a person and we would like to have information about the expenses. For simplicity, we assume that an expense has a name and the amount of money.

The user should be able to add an expense to the transaction log, show all the transactions and show transaction(s) with an amount bigger than a certain value.

### Screenshots

Graphical user interface, text, application

Description automatically generated

Figure : The GUI

### Provided material

First create the form as shown in Figure 4 and add a class *Expense*. Try to determine what members the class should have:

* Which two private instance variables should you include? (HINT: see the GUI and determine which controls is about an expense).
* What statement(s) should the method *public void InitializeExpense(string name, double amount)* have?
* What statement(s) should the method *public string GetInfo()* have? (HINT: see the GUI and determine which controls is showing the expenses).

|  |  |
| --- | --- |
|  | Starting from this point the assignment requires you to implement statements in the form because we want to keep the complexity to only one class; this is your first time working with classes &. Unfortunately, this is violating a good practice when considering object orienting programming. NEVER do this in a project!  In an upcoming lesson we will cover a more proper way of doing this and discuss *Separation of Concerns*! |

Let’s implement the form by doing the following:

* Declare a variable *transactionLog* of type List<Expense>; what scope do you think this variable must have?
* Determine where you want to write the statement to create and assign the List<Expense> to *transactionLog*.
* Implement the *Add expense to transaction log*-button: when this button is clicked an *Expense*-object must be created, initialized and finally added to *transactionLog* (i.e. the list with Expense-object(s)).
* Implement the *Show all expenses*-button: when this button is clicked the ListBox must be cleared and then the info of all the *Expense*-object(s) in *transactionLog* must be shown.   
  HINT: What statement can you use to iterate through the List and what method must you call to add to the ListBox?
* Implement the *Show expenses with same amount* -button: when this button is clicked the ListBox must be cleared and then only the info of *Expense*-object(s), if any, in *transactionLog* with the same amount as the NumericUpDown must be shown.

## Programming Assignment 4: Moving a car

Difficulty:

The assignment covers the following learning goals:

* Working with a Timer
* Working with a PictureBox

### Case description

You’re going to create simple application to move a PictureBox at set intervals. This can be achieved by adding a Button, Timer and a PictureBox to the form (see Figure 6: POSSIBLE GUI).

The idea is to change the location of the PictureBox control to simulate a moving car. For example, move a car-image/PictureBox by 10 pixels to the right every half second. Changing the location can be done as follows:

<control>.Location = new Point(<new x coordinate>, <new y coordinate>);

The <grey> parts of the code snippet should be replaced with the actual values (e.g. pbxCar.Location = new Point(50, 0);).

Implement the required code to start moving the image/PictureBox after clicking Start. Can you also stop the movement after the right edge of the form has been reached? What about bouncing it between the edges?

### Possible GUI

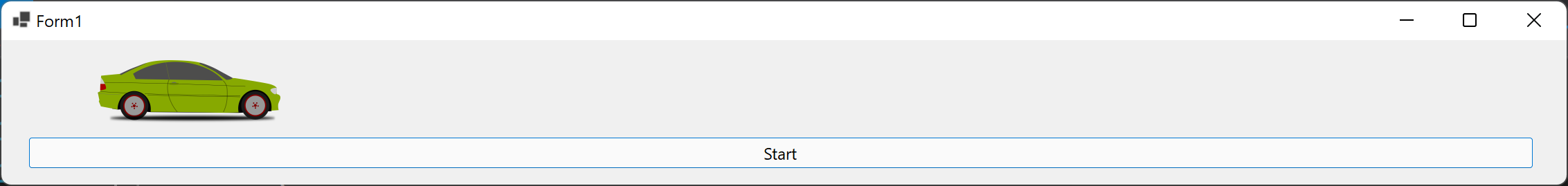


Figure : POSSIBLE GUI