Application Development Orienting

Assignments week 7

Quiz questions, practical assignments and

answers to quiz questions

**Version:** 0.2 **Last updated:** 04 oktober 2019

Contents

[1 Quiz 3](#_Toc21269284)

[1.1 Question 1 3](#_Toc21269285)

[1.2 Question 2 3](#_Toc21269286)

[1.3 Question 3 3](#_Toc21269287)

[1.4 Question 4 3](#_Toc21269288)

[1.5 Question 5 4](#_Toc21269289)

[2 Practical assignments 5](#_Toc21269290)

[2.1 Programming Assignment 1: Money Box 5](#_Toc21269291)

[2.1.1 Case description 5](#_Toc21269292)

[2.1.2 Explanation 5](#_Toc21269293)

[2.1.3 Screenshots 6](#_Toc21269294)

[2.2 Programming Assignment 2: Student & Course Management 7](#_Toc21269295)

[2.2.1 Case description 7](#_Toc21269296)

[2.2.2 Screenshots 7](#_Toc21269297)

[2.2.3 Explanation 7](#_Toc21269298)

[2.2.4 Additional features 8](#_Toc21269299)

[2.3 Programming Assignment 3: Back to the automated teller machine! 9](#_Toc21269300)

[2.3.1 Case description 9](#_Toc21269301)

[2.3.2 Explanation 9](#_Toc21269302)

[2.3.3 Screenshots 10](#_Toc21269303)

[2.3.4 Additional features 11](#_Toc21269304)

[2.4 Programming Assignment 4: Smart Lamp Control Application 12](#_Toc21269305)

[2.4.1 Case description 12](#_Toc21269306)

[2.4.2 Screenshots 12](#_Toc21269307)

[2.4.3 Explanation 12](#_Toc21269308)

[2.5 Programming Assignment 5: Tic Tac Toe (optional assignment, uses 2 dimensional arrays) 13](#_Toc21269309)

[2.5.1 Case description 13](#_Toc21269310)

[2.5.2 Screenshots 13](#_Toc21269311)

[2.5.3 Explanation 13](#_Toc21269312)

[2.5.4 Additional features 14](#_Toc21269313)

[3 Quiz answers 15](#_Toc21269314)

# Quiz

Difficulty: C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-3_5.png. Estimated time: 30 minutes.

Answers to the quiz-questions can be found in the last section of this chapter.

## Question 1

In a solution, we have the following method definition:

private void ButtonShowArrayNumbers\_Click(object sender, EventArgs e)

{

int[] integerArray = new int[7] {0, 0, 0, 0, 0, 0, 0};

for (int i = 0; i < integerArray.Length; i++)

{

integerArray[i] = 17 \* i + 9;

listBoxArrayNumbers.Items.Add(integerArray[i]);

}

}

Assume this method is called when pushing the button ShowArrayNumbers and correctly executed, what will be shown by the listBox?

## Question 2

In a solution, we have the following method definition.

private void ButtonShowArrayNumbers\_Click(object sender, EventArgs e)

{

int[] integerArray = new int[7];

integerArray[0] = 1;

integerArray[4] = 2;

integerArray[integerArray.Length - 1] = 5;

for (int i = 0; i < integerArray.Length; i++)

{

listBoxArrayNumbers.Items.Add(integerArray[i] + integerArray[4]);

}

}

Assume this method is called when pushing the button ShowArrayNumbers and correctly executed, what will be shown by the listBox?

## Question 3

The code below is given in a working program, will this code compile and if it does not, what is the error?

int[] myArray = { 1, 2, 3, 3.75 };

## Question 4

The code snippet below was found in a program, will the code below compile and if so what would the output be? How can we improve the code?

int[] myArray = { 1, 2, 3, 4, 5 };

for (int i = myArray.Length; i >= 0; i--)

{

listBoxArrayNumbers.Items.Add(myArray[i]);

}

## Question 5

Look at the following code. The code is written correctly and executes, what will be shown by the list box?

listBoxArrayNumbers.Items.Clear();

int[] myArray = { 1, 2, 3, 4, 5, 6, 7 };

int i = 0;

int j = myArray.Length - 1;

while (i < j)

{

myArray[i] = myArray[i] + myArray[j];

myArray[j] = myArray[i] - myArray[j];

myArray[i] = myArray[i] - myArray[j];

listBoxArrayNumbers.Items.Add(myArray[i]);

listBoxArrayNumbers.Items.Add(myArray[j]);

i++;

j--;

}

# Practical assignments

## Programming Assignment 1: Money Box

Difficulty: C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-3_5.png . Estimated time: 75 minutes.

The assignment covers the following learning goals:

* Understanding the use of arrays and creating a simple application that relies solely on the array.

### Case description

You’re going to implement your very own money box. A money box that used to be given as a present when opening of a Dutch bank account (for children) perfectly represents what arrays are useful for.

### Explanation

Start a new Windows Form Application project and make it look something like the screenshot attached. Use six buttons and change the text of the buttons to €0.05, €0.10, €0.20, €0.50, €1.00, €2.00. Add a list box to the form, and two label’s, these will be used to give an overview of the coins and the totals.

Create two arrays inside your form, one array is needed to keep track of the number of specific coins that are inserted into the money box, this one is of integer type. The other one is needed to keep track of the value of the coins in the same respective position (0.05, 0.10, …) and is for example of type double. Both arrays should have 6 elements total.

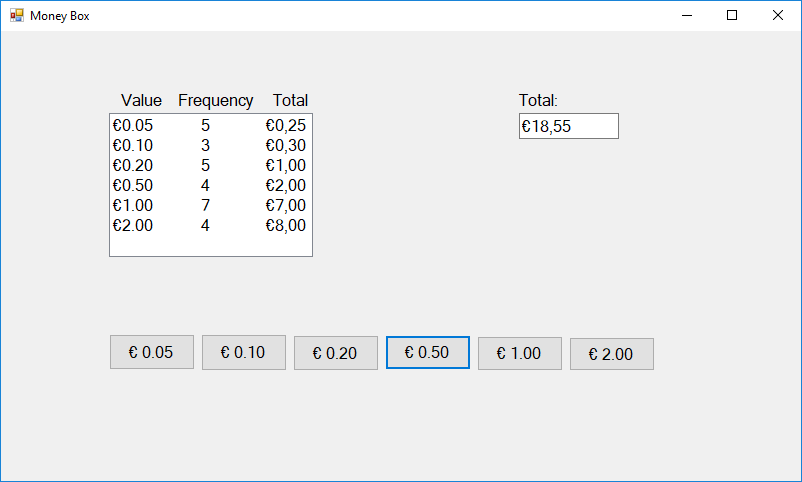
To create functionality, click on all the buttons on the form and inside their click handler, increase the frequency of that coin occurring by 1, thus update the array at the respective index (based on which button is clicked).

Now to display the information in the listbox, create a typed method something like private string CoinInfoString(int index), that will handle the formatting and return a string with information as: the value of the coin, the frequency and the total for this coin (at index location index in the array).

To calculate the total for that coin at location index, also create a typed method, something as private double CalculateTotal(int index) or private void CalculateTotal(int index).

And finally create a void method inside the form that will handle displaying all the coin information in your form. This can be a method that will clear the list box, populate it with all the coin values, frequencies and totals for that coin. And also add a total amount that you stored inside your money box, thus a total of the totals per coin.

### Screenshots



## Programming Assignment 2: Student & Course Management

Difficulty: . Estimated time: 120 minutes.

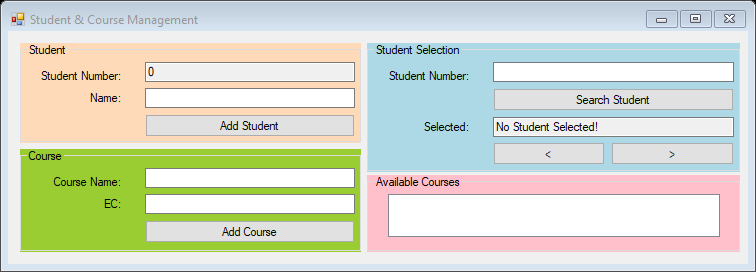
The assignment covers the following learning goals:

* Implement a management application with arrays as data collection. This assignment will learn you to work with arrays together with the use of different form controls and applying all the things you learned so far.

### Case description

Your client wants to manage keep track of students enrolled and the courses that the academy provides. Since you have learned about arrays, you can now start to store related data. The client wants to be able to store a student’s name and have the student be identified by a unique number. He also wants to have a simple overview of the courses and ECs that the university provides.

### Screenshots



Create a form similar to the one in the screenshot using the different controls, i.e. groupbox, button, textbox, label and listbox.

### Explanation

Start of by creating one variable inside your form like private int studentNumber, and initialize this variable with the value 0.

Next, create several arrays. You should have 2 string arrays for the student names and courses and one int array for the EC’s of the courses. Initialize the arrays with for example 4 places for the student names, 5 for the number of courses.

Let us start with adding functionality to add a student to the array. When the user enters a student name and clicks the add button, the student name should be inserted in the string array at position studentNumber. Provide feedback to the user by using a MessageBox to display that the user was added successfully.

Do something similar for adding courses to the system. Create a loop to find the first empty spot for a course and add it in that index. Also add the number of ECs in the same index in the array of ECs. Make sure that after adding a course, the coursesListBox is updated by displaying all the current courses and their related ECs. Prevent the application from crashing when you start exceeding the maximum number of courses you entered earlier.

Up next, let’s make the student selection functional. With a click on search student we want to look up that student number (index) in the array and if we find a name display it and otherwise display a message in the TextBox that we have no student selected.

For the arrow buttons, we want to make sure that if we have no selected student, whichever Button is pressed, the first index is displayed. To get this working, add a private int searchIndex to your form.

If the user presses either arrow for the first time, move one position forward or backward and display if there is a student for that number in the selected TextBox.

In case you forgot, to wrap around the array you can make use of the % modulus operator.

### Additional features

Create some course removal functionality for the courses. For example create a button that will remove the selected course based on the selection from the ListBox. This should nicely interact with your already implemented functionality as removing a course should lead to being able to re-add a new course again.

## Programming Assignment 3: Back to the automated teller machine!

Difficulty: C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-3_5.png. Estimated time: 120 minutes.

The assignment covers the following learning goals:

* Understanding updating an existing application by extending it with new functionality, and in particular functionality involving arrays.

### Case description

You’re going to modify the application you worked on in week 3 when you covered functions. Since we are now learning about arrays we can provide some additional complexity. For this application we will add a few arrays to the form that allow us to represent different bank accounts!

### Explanation

Open your previous version of the ATM application. As you can still see, several functions were implemented to facilitate a properly functioning application. Now we are up to modifying some of the existing functionalities to suit our new goal!

Start by adding an extra Panel and two extra RadioButton objects to the Form. The radio buttons will be used to swap between different accounts. To keep it simple, set the Checked property of one of the two radio buttons to true.

Add 1 integer variable to the form, the integer will be an index to keep track of which bank account is selected. Now add one array of type double. The array will be used to track the bank account balances.

Once you start your application, you want to obtain the index of the bank account that is selected. Create a typed method to obtain the index of the selected bank account number and assign this to the bank account index variable you have created earlier.

Let us now continue with addressing the errors that we see. First lets modify the function that shows the bank balance. In the previous version we made use of one variable that kept track of one bank account balance. We now need to display the right bank balance based on its position in the array. Make it work by using the variable you created earlier.

Nice, now we only need to change a few things to update the right bank balance. Try to figure these out yourself as an exercise! In case you really get stuck, ask your teacher or fellow student for some help.

If you got everything working and launch your application, do you notice anything when you select a different bank account with the radio button? Maybe it may be that you forgot to implement the CheckedChanged event that you can use to change between bank accounts. Try to implement it such that whenever you change the radio button, the right balance is displayed and updated!

### Screenshots

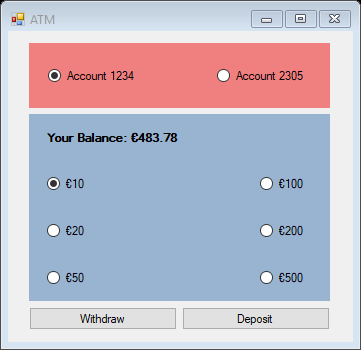


Figure 1: Possible GUI

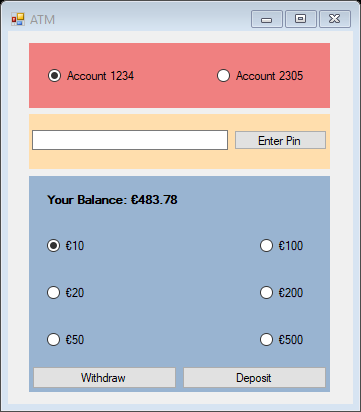


Figure 2: Possible additional feature GUI

### Additional features

In case you want a bit extra of a challenge and functionality, try to create a ‘login’ form by having the user enter a pin code before the user can access the bank account.

Add another panel for example between the one with the account numbers and the balances as in the screenshot.

Hide the panel with the bank balance and the buttons by navigating to the properties of the panel and set the Visible property to false.

Up next we need to create a new array in the form that will represent the pin codes of respectively account 1 and account 2. It does not really matter whether the array is of type string or int as long as they hold a value that represents a pin code.

Since pretty much all is set, add code to the button click event handler that will verify if the pin code matches the one of the selected account. If it does match, make the hidden panel visible again and hide the panel with the pin code. Now be careful, since if you change the radio button selection you have access to the other account as well. Fix this, by applying the right logic.

Right, our ATM application is taking shape, but to protect our clients’ bank accounts a bit better, we will want to keep track of the number of times an incorrect pin was entered. For this, you might’ve guessed it, we will again use an array. Add this array of type int to your form with initial values of 0.

So whenever the user enters an incorrect pin, you want to update the number of times an incorrect pin was entered. Now it is up to you to choose and add a check that once a user entered 3 or 5 times or however many times an incorrect pin, the account will be blocked.

And that’s pretty much it. You should now have a somewhat safer ATM application that allows you to use multiple accounts!

## Programming Assignment 4: Smart Lamp Control Application

Difficulty: 1 star. Estimated time: 45 minutes.

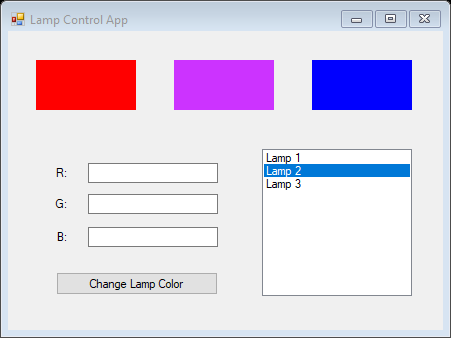
The assignment covers the following learning goals:

* Understanding and applying arrays as data structure, enabling you to work with different states for similar types of data.

### Case description

For a client you are going to make a simple smart lamp control application. This will consist of three picture boxes representing the lamps, with for the user three input text boxes to change the color of the three smart lamps.

### Screenshots



### Explanation

Start of by making your form look somewhat like the form provided in the screenshot. Add three PictureBoxes, three TextBoxes, Three Labels, a ListBox and a Button to the Form.

Inside the code of your form, use three array variables, for each lamp one. Each array should hold three integer values that determine the color to be shown. For instance three times value 255 will result in ‘black’ and three times value 0 result in ‘white’.

You can change the colors of the picturebox by changing the BackColor property of the PictureBox and assign Color.FromArgb(r, g, b) to it.

Implement the code for the Change Lamp Color click event handler. Make sure that whenever this button is clicked, the input RGB values are assigned to the selected lamp (from the ListBox).

## Programming Assignment 5: Tic Tac Toe (optional assignment, uses 2 dimensional arrays)

Difficulty: C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-3_5.png. Estimated time: 90 minutes.

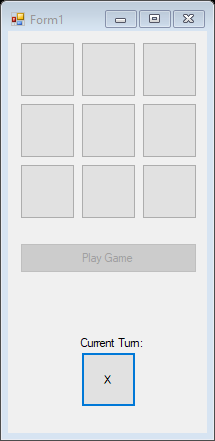
This exercise is only for students that want to do an extra challenge. The required knowledge goes beyond the theory taught in the lecture of week 6. The assignment covers the following learning goals:

* Applying multi-dimensional arrays with objects as data structure, allowing you to keep track of a game’s state and manage it accordingly.

### Case description

For this assignment you need to create a simple GUI as shown in the screenshot. This GUI provides the base for a tic tac toe game which you will be implementing. In case you are not familiar with the game, check the internet for an explanation. To keep track of the game’s state we will be making use of arrays!

### Screenshots



### Explanation

To get things set up let’s start off with creating two arrays, one will be a two dimensional array of Button objects, define something as private Button [,] ticTacToeButtons. Up next we will also keep track of our player symbols by using an array, create another array like so private char[] symbols.

Up next we want to initialize the Button array as a 3 by 3 array and assign in every available index one of the 9 Button objects from the Form. Initialize it inside the Form’s constructor and make sure you keep the order matching the way the buttons are laid out on the Form. This is important since we will be checking the game’s state after every turn, to test if there is a winner.

Make sure you also initialize the array with symbols (characters X and O) to use throughout the game.

Much like in the screenshot, pressing the Button with Play Game as text should result in this Button being disabled. The 9 Button objects representing the play field should be enabled and the current player’s symbol should be displayed as X.

Implement the button click events for all 9 Button objects, all 9 will need similar logic. You should make the following things working. A current player symbol is assigned to the clicked Button’s text property. You can program this by making use of a simple if statement or you can make use of the modulus operator (the modulus would be good practice). This button must then be disabled such that another player cannot override it. And the game state should be checked to see if after that turn, the game is won by a player.

If the game is won, deactivate the game by implementing some simple logic, disabling all buttons and displaying an appropriate message.

Otherwise, update the state allowing for the next player to take a turn.

So for a game to be won by a player, a symbol must occur three times in a row. This can be in a row, in a column or in one of the two diagonals.

Create a function for each of those, for example private bool CheckRow(int rowIndex), private bool CheckColumn(int columnIndex) and both diagonals separately, which could be something as private bool CheckDiagonalOne() and for the other diagonal likewise.

There are many ways to implement a clever check that is more efficient than doing it in this way, but this is good practice none the less.

So for the check row function, an integer is entered, and it is up to you to implement a check that will determine if all 3 symbols in that row are equal, and thus we have a winner. In case there is a winner in a row combination, return true, if not, return false. Remember to do this for all the rows.

Similarly, create the check for the column. This is almost done in identical fashion as the one for the rows.

For the diagonals you can create a similar kind of loop, remember that for that a cell is part of the main diagonal if the indices are identical (0, 0), (1, 1), (2, 2). And for the anti-diagonal the cells are in locations (0, 2), (1, 1), (2, 0).

In case you find it too hard to implement the checks by using a loop, you can always fall back on statically (hard coded) checks. But you are of course encouraged as a practice to do it by making use of looping mechanisms and optimize it if you have extra time left.

### Additional features

Since a user can get quite frustrated when s/he has to close and re-open your application every time a game is played, implement functionality that resets the game’s state and allows to play another game. Additionally, keep track by the number of times a player won the game (X or O) respectively by creating another array.

# Quiz answers

|  |  |
| --- | --- |
| Question | Answer |
| 1 | 9  26  43  60  77  94  111 |
| 2 | 3  2  2  2  4  2  7  Note that the C# compiler implicitly initializes each array element to a default value depending on the array type. For an int array all elements are initialized to 0. |
| 3 | It will not compile, a decimal number is assigned in an integer array. |
| 4 | It does compile as it is written correctly. It does however crash during run time. Since it tries to access an index 5 which is not existent (last index is 4 since we start counting at index 0). |
| 5 | 7  1  6  2  5  3  Since every element at index i is swapped with the element at index j as long as the index i is smaller than the index j. With j decrementing with 1 every iteration and i incrementing by 1. |