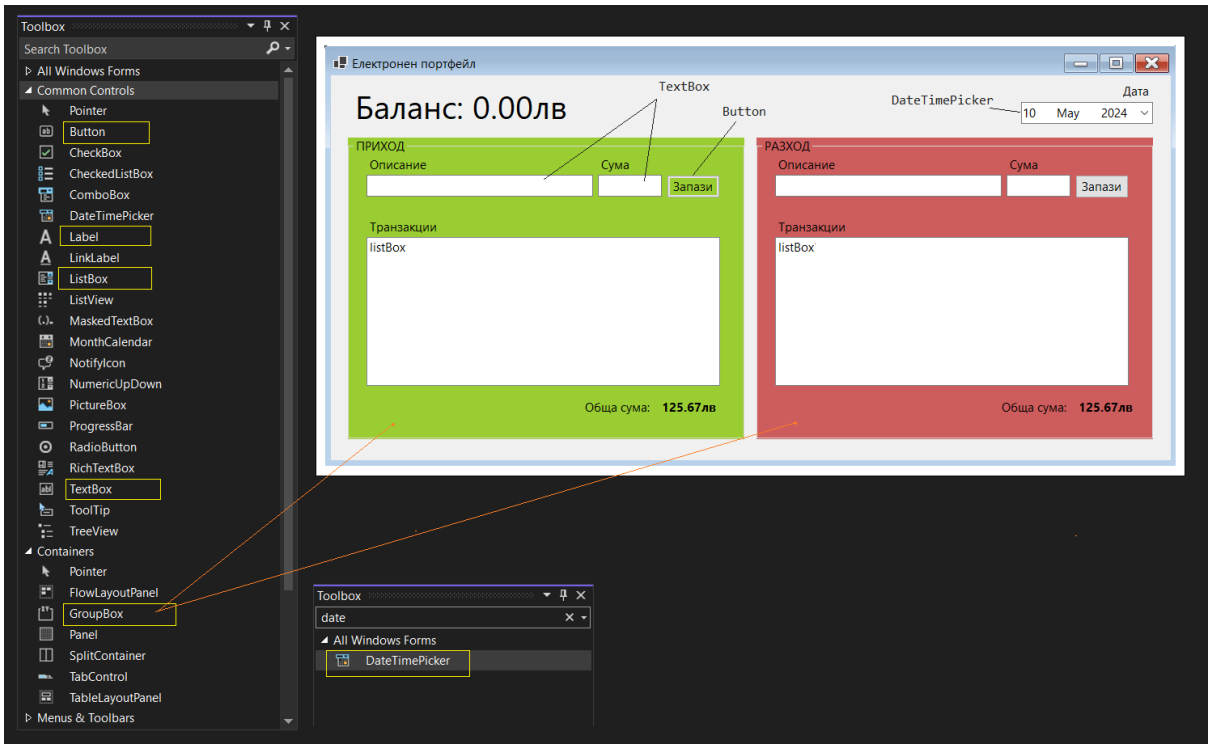
Практически проект

Електронен портфейл

Създайте Windows Forms приложение , което позволява на потребителя да въвежда своите финансови транзакции – приходи и разходи. Данните да се записват във файл за съхранение и анализ. Приложението да позволява визуализация на общия приход и разход (за определен период).

Примерено изображение на проекта



Етапи при изграждането на приложението.

I. Създаване на Windows Forms приложение във Visual Studio

В Visual Studio създайте проект с име {My-First-Name}-E-Wallet(например: Ivan-E-Wallet).

II. Създаване на графичния интерфейс

- Променете заглавието на формата с нов текст: Електронен портфейл

Използвайки панела с контроли на Visual Studio добавете и подредете нужните интерфейс елементи във формата на приложението. Например:

- Добавете *label* за баланс със текст: **Баланс: 0.00лв**
  - Font Size: 24pt
- Добавете *GroupBox* от категория *Containers* на панела с контроли и му задайте текст **ПРИХОД**
- В контейнера за приходи добавете и останалите контроли:
  - *Label* с текст **Описание** и label с текст **Сума**
  - *Button* с текст **Запази**

- *Label* с текст **Транзакции**
- *ListBox*
- *Label* с текст **Обща сума:** и *label* с текст **125.67лв** с удебелен *шрифт*

***Подредете** и **подравнете** контролите в контейнера **Приход** по начина показан в примерното изображение на приложението.*

- За да създадем секцията за **Разход**и:
  - Маркирайте с помощта на мишката **ComboBox Приход**, след което го копирайте и поставете.
  - Издърпайте копието и го позиционирайте в дясно от секцията **Приход**
  - Променете характеристиките в копието така, че те да отговарят на секция **Разход**

Накрая да добавим и контролата за **дата**

- *Label* с текст **Дата**
- Намерете контролата **DateTimePicker**, като напишете в търсачката на панела с инструменти ***date*** и го добавете към формата на приложението

III. Именуване елементите(контролите) на графичния интерфейс

За именуване на контролите ще използваме Унгарската нотация, според която името на променливата има префикс с името на контролата последвано от описателно име присъщо за контролата. **Всяка дума с изключение на първата започва с главна буква.** Например контролата TextBox използвана за въвеждане на името на ученик би могла да има име:

***textBox*Name**, или ***textBox*StudentName**, или ***textBox*FullName** и т.н.

Контрола **TextBox**

Т.к. в приложението има две секции (за приходи и разходи), чиито контроли са сходни, бихме могли да разграничим имената им като добавим и името на секцията към която принадлежат. Например бутона „Запази“ би могъл да се казва

button**ProfitS**ave за приходите и button**ExpenseS**ave за разходите.

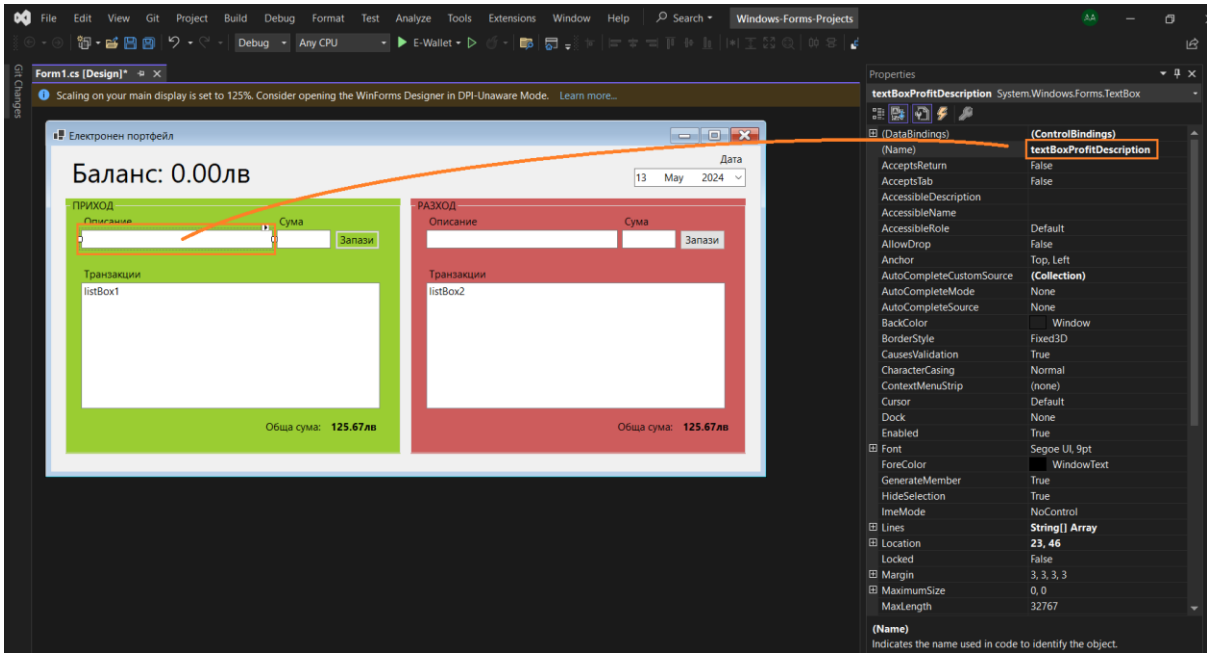
секция **Profit**(Приход)

секция **Expense**(Разход)

Контрола **Button**

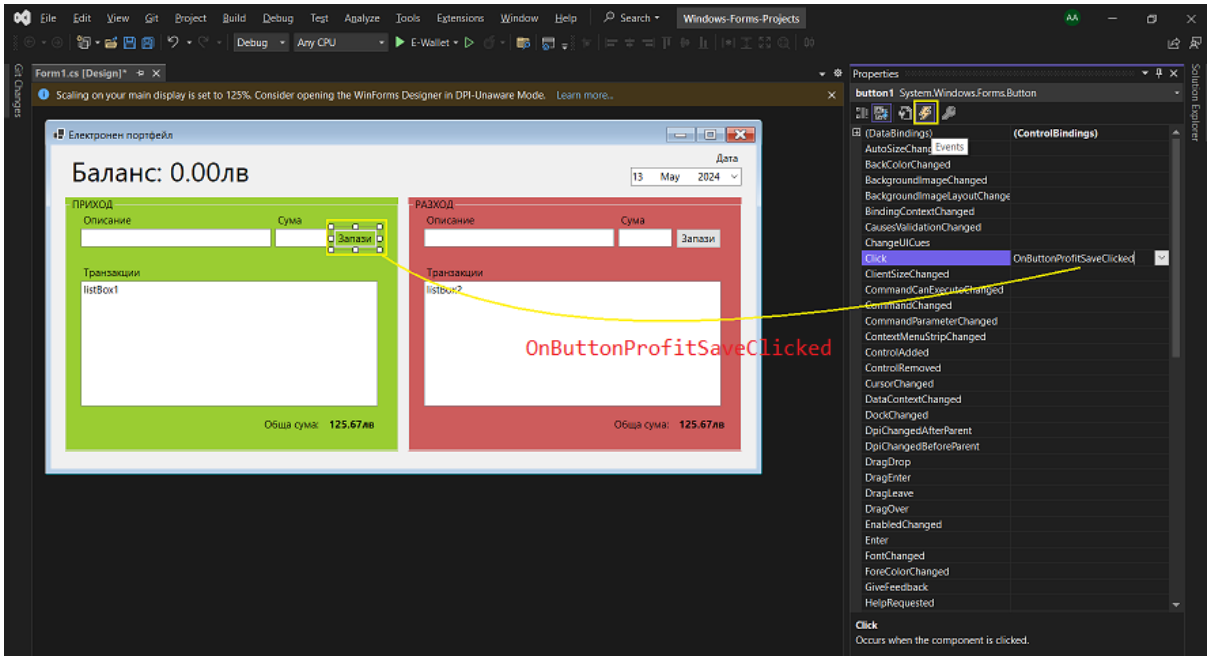
Използвайки описаната по-горе конвенция за именуване задайте подходящи имена на контролите.

Забележка: Част от контролите, предимно етикети(labels), които се използват за визуализиране на текстово съдържани, могат да бъдат пропуснат при именуване, защото те няма да бъдат извиквани никъде от кода на приложението .



IV. Създаване на методи - обработчици на събития

Ред е да създадем код, който да се изпълни, когато бутонът „Запази“ бъде кликнат(отнася се и за двата бутона). За целта е необходимо да маркираме дадения бутон с мишката и от панела с неговите характеристики (**Properties**) да включим бутона отговарящ за събитията (**Event**), да изберем събитието **Click** и да му дадем име – това ще бъде обработчика на събитието, който ще се извика, когато бутонът бъде натиснат с мишката. Дайте описателни имена на обработчиците, като се ръководите от примера.



V. Имплементиране бизнес логиката на приложението.

Нека напишем кода, който трябва да се изпълни, когато бъде натиснат бутонът „Запази“, като използваме създадените в предишния етап обработчици на събития.

Първо ще имплементираме логиката за секция „Приход“. Когато въведем „***Описанието***“ и „***Сумата***“ за конкретен приход в съответните текстови полета и кликнем бутона „***Запази***“ трябва да се случат няколко неща описващи логиката на нашето приложение:

1. да прочетем данните за въведения приход;
2. да запишем прочетените данни в файл „**profit.txt**“ във формат:
    a. *{date}#{description}#{amount}*
3. да актуализираме и визуализираме списъка с всички приходи;
4. да преизчислим и визуализираме общата сума за приход
5. и да занулим текстовите полета за последващо въвеждане на нов приход.

```
private void OnButtonProfitSaveClicked(object sender, EventArgs e)
{
    // Прочитаме дата
    string date = dateTimePicker.Text;

    // 1. Прочитаме входните данни за приход
    string description = textBoxProfitDescription.Text;
        // - за работа с финанси използваме типа decimal
    decimal amount = decimal.Parse(textBoxProfitAmount.Text);

    // 2. Записваме информацията в файл
        // - декларираме името на файла, в който ще пишем
    string filePath = "../../../profit.txt";
        // - създаваме поток за писане във файл
    StreamWriter writer = new StreamWriter(filePath, true, Encoding.Unicode);
    using (writer)
    {
        writer.WriteLine($"{date}#{description}#{amount}");
    }

    // 3. Актуализираме списъка с приходи
    listBoxProfits.Items.Add($"{date}: {description} - {amount}");

    // 4. Обновяваме общия приход
    decimal total = decimal.Parse(labelProfitTotal.Text) + amount;
    labelProfitTotal.Text = total.ToString();

    // 5. Занулуваме полета за въвеждане на нов приход
    textBoxProfitDescription.Text = string.Empty;
    textBoxProfitAmount.Text = string.Empty;
}
```

Добавя данните в края на файла

Използваме Unicode кодираща таблица

VI. Въвеждане и запис на разход – САМОСТОЯТЕЛНО ИЗПЪЛНЕНИЕ

За секцията „Разход“ постъпваме по аналогичен начин – кодът ще се изпълни от обработчика на събитието отговарящо за въведените разходи.

VII. Помощни методи за актуализация на данните при промяна на данните

- Помощен метод за актуализация на **баланса**.
  - Напишете метод **UpdateBalance()**, който актуализира ***БАЛАНСА*** при всяка промяна в приходната или разходната част на приложението.

```
private void UpdateBalance()
{
    // Прочитаме общия приход от контролата labelProfitTotal
    decimal totalProfit = decimal.Parse(labelProfitTotal.Text);
    // Прочитаме общия разход от контролата labelExpenseTotal
    decimal totalExpence = [redacted];

    // Разликата на двете е новия БАЛАНС
    decimal balance = [redacted];
    // Записваме в контролата labelBalance
    labelBalance.Text = $"Баланс: {balance}";
}
```

  - **Извикайте метода UpdateBalance()** при стартиране на програмата и при всяко добавяне на нов приход и разход съответно **при инициализацията на приложението, и от обработчиците на събитието клик за двата бутона „Запази“.**
- **Зареждане на данните** за приходи и разходи от съответните файлове при стартиране на програмата.
  - Напишете помощен метод **LoadProfits()**, който чете всички **приходи** от файла с приходите;

```
private void LoadProfits()
{
    decimal totalAmount = 0;
    // Пътя до файла от който ще четем
    string filePath = "../../../profit.txt";
    // Поток за четене от файл
    StreamReader reader = new StreamReader(filePath, Encoding.Unicode);
    using (reader)
    {
        // Четем от файла ред по ред до края му. Всеки ред в файла е във следния формат:
        // 20 May 2024#на заем от приятел#22.10
        string line;
        while((line = reader.ReadLine()) != null)
        {
            // Разбиваме всеки ред на части по разделител символа #
            string[] profitDetails = line.Split('#');
            string date = profitDetails[0];
            string description = profitDetails[1];
            string amount = profitDetails[2];
            // добавяме детайлите на всеки приход към списъка от приходи
            listBoxProfits.Items.Clear(); // Превантивно зачистваме съдържанието на списъка
            listBoxProfits.Items.Add($"{date}: {description} - {amount}");

            // добавяме парите от всеки прочетен ред към общата сума
            totalAmount += decimal.Parse(amount);
        }
        // След като минем през всички редове на файла актуализираме натрупания в totalAmount приход
        labelProfitTotal.Text = totalAmount.ToString();
    }
}
```

  - Напишете по аналогичен начин помощен метод **LoadExpenses()**, който чете всички **разходи** от файла с разходите;