МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу кафедра інженерії програмного забезпечення

Практична робота № 1

з дисципліни «Архітектура та проектування програмного забезпечення .Net»

*назва дисципліни*

на тему: «РОЗРОБКА КОМПОНЕНТІВ»

Виконав: студент 2 курсу групи 622П напряму підготовки (спеціальності)

121 інженерія програмного забезпечення

(шифр і назва напряму підготовки (спеціальності))

Скицко Р. С.

(прізвище й ініціали студента)

Прийняв: доц. каф. 603, к.т.н., доцент

Лучшев П.О.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: Кількість балів: Оцінка ECTS:

Харків – 2025

**Мета роботи:** Вивчення особливостей створення та використання компонентів(components) та елементів інтерфейсу користувача (UI elements). Закріплення навичок об'єктно-орієнтованого програмування (OOP) з дотриманням принципів SOLID.

**Варіант завдання**

17. Назва та опис функціональності: Лічильник символів – TextBox для введення, Label для відображення кількості введених символів або знаків, що залишилися.

Рекомендовані:

* Властивості: MaxLength, CurrentLength
* Події: TextChanged

**Опис компонента**

Компонент SymbolCounter є візуальним елементом, створеним на базі UserControl, який реалізує функціонал підрахунку символів у текстовому полі. Основна його мета — відстеження кількості введених символів у полі введення textBox1 та оповіщення про зміни за допомогою події.

Основні елементи структури:

* textBox1: Текстове поле для введення даних користувачем.
* MaxTextLength (властивість): Встановлює або повертає максимальну довжину введення.
* CurrentTextLength (властивість): Повертає поточну кількість введених символів. Має приватний setter.
* SymbolTextChanged (подія): Викликається при зміні тексту.
* Метод TextBox1\_TextChanged: Обробник події TextChanged, який оновлює CurrentTextLength та викликає подію SymbolTextChanged.

Використання принципів SOLID:

S — Single Responsibility Principle (Принцип єдиної відповідальності):  
Компонент SymbolCounter відповідає лише за одну функцію — відслідковування кількості символів у текстовому полі. Він не займається ні валідацією даних, ні збереженням, ні логікою зовнішнього вигляду, тобто його відповідальність чітко визначена.

O — Open/Closed Principle (Принцип відкритості/закритості):  
Компонент відкритий для розширення — наприклад, можна додати додаткові події або нову логіку, не змінюючи поточний код. Завдяки події SymbolTextChanged можна підключити обробники ззовні, не змінюючи сам UserControl.

I — Interface Segregation Principle (Принцип розділення інтерфейсу):  
Cтруктура компонента чітко виділяє те, що повинно бути доступне зовні (через публічні властивості та події), не змушуючи користувача залежати від непотрібної реалізації.

L — Liskov Substitution Principle (Принцип підстановки Лісков) та D — Dependency Inversion Principle (Принцип інверсії залежностей) не використовувалися

**Код розробленого компонента (самостійно розроблених частин)**

Зміст файлу SymbolCounter.cs:

using System.ComponentModel;

namespace Library

{

[ToolboxBitmap(typeof(SymbolCounter), "SymbolCounter.ico")]

public partial class SymbolCounter : UserControl

{

public SymbolCounter()

{

InitializeComponent();

textBox1.TextChanged += TextBox1\_TextChanged;

textBox1.MaxLength = 10;

}

private void TextBox1\_TextChanged(object sender, EventArgs e)

{

CurrentTextLength = textBox1.Text.Length;

SymbolTextChanged?.Invoke(this, EventArgs.Empty);

}

[DefaultValue(10)]

public int MaxTextLength

{

get => textBox1.MaxLength;

set => textBox1.MaxLength = value;

}

[DesignerSerializationVisibility(DesignerSerializationVisibility.Visible)]

[DefaultValue(0)]

public int CurrentTextLength

{

get;

private set;

}

public event EventHandler SymbolTextChanged;

}

}

# **Екранні форми тестування та результати.**

Таблиця 1 – результати тестових перевірок

|  |  |  |  |
| --- | --- | --- | --- |
| Тест № | Вхідні дані | Вихідні дані | |
| Очікуваний результат | Отриманий результат |
| 1 | - | Символів введено: 0 Символів залишилося: 10 | Рисунок 1 |
| 2 | 123 | Символів введено: 3 Символів залишилося: 7 | Рисунок 2 |
| 3 | 1234567890 | Символів введено: 10 Символів залишилося: 0 | Рисунок 3 |

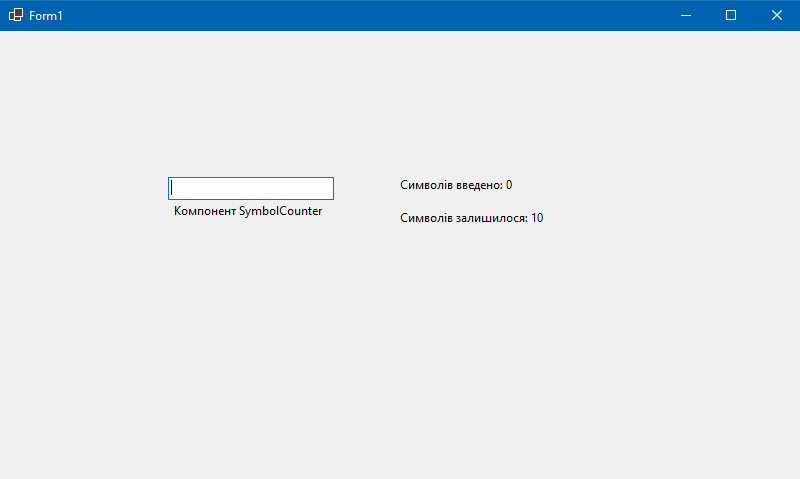


Рисунок 1 – Результат тесту 1

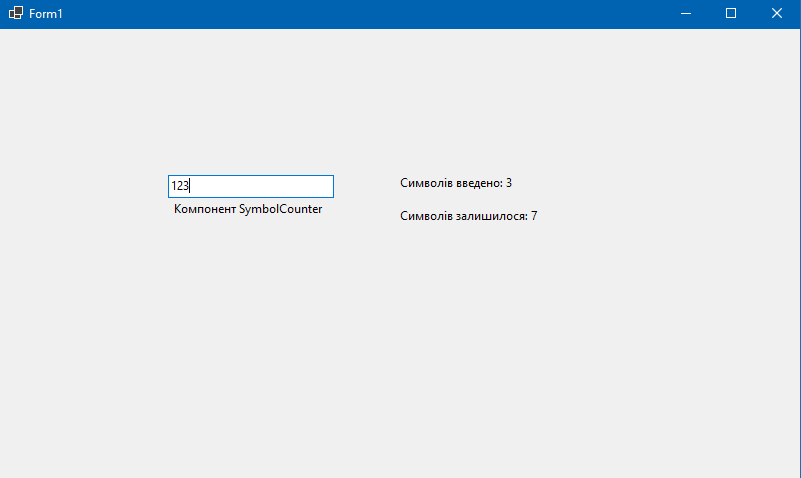


Рисунок 2 – Результат тесту 2

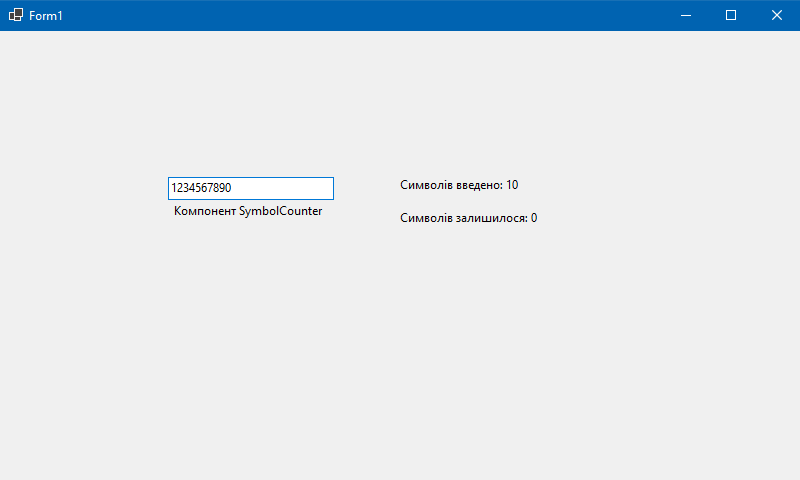


Рисунок 3 – Результат тесту 3

Таблиця 2 – результати тестових перевірок NuGet пакету в окремому рішенні

|  |  |  |  |
| --- | --- | --- | --- |
| Тест № | Вхідні дані | Вихідні дані | |
| Очікуваний результат | Отриманий результат |
| 1 | - | Символів введено: 0 Символів залишилося: 10 | Рисунок 4 |
| 2 | 123 | Символів введено: 3 Символів залишилося: 7 | Рисунок 5 |
| 3 | 1234567890 | Символів введено: 10 Символів залишилося: 0 | Рисунок 6 |

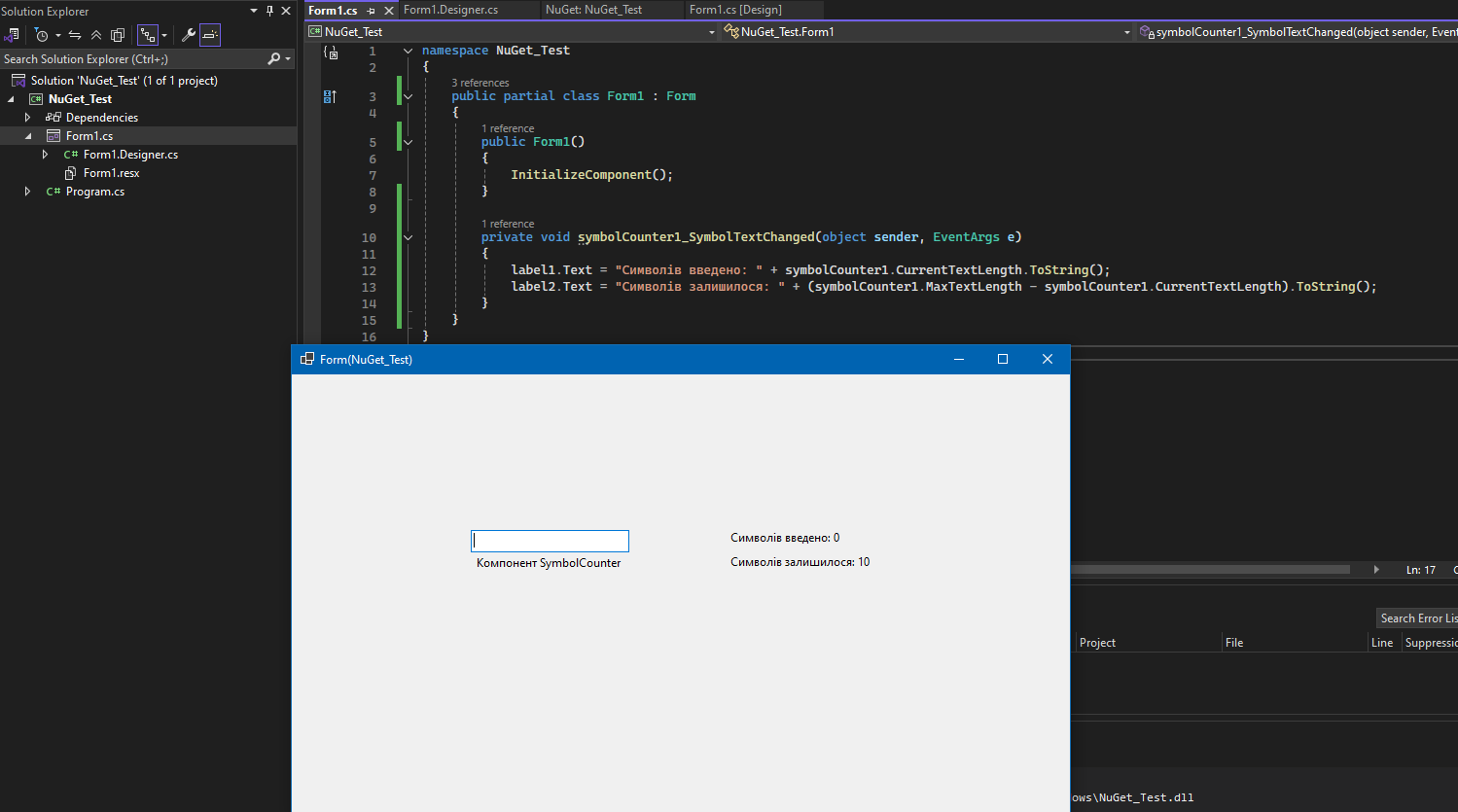


Рисунок 4 – Результат тесту 1

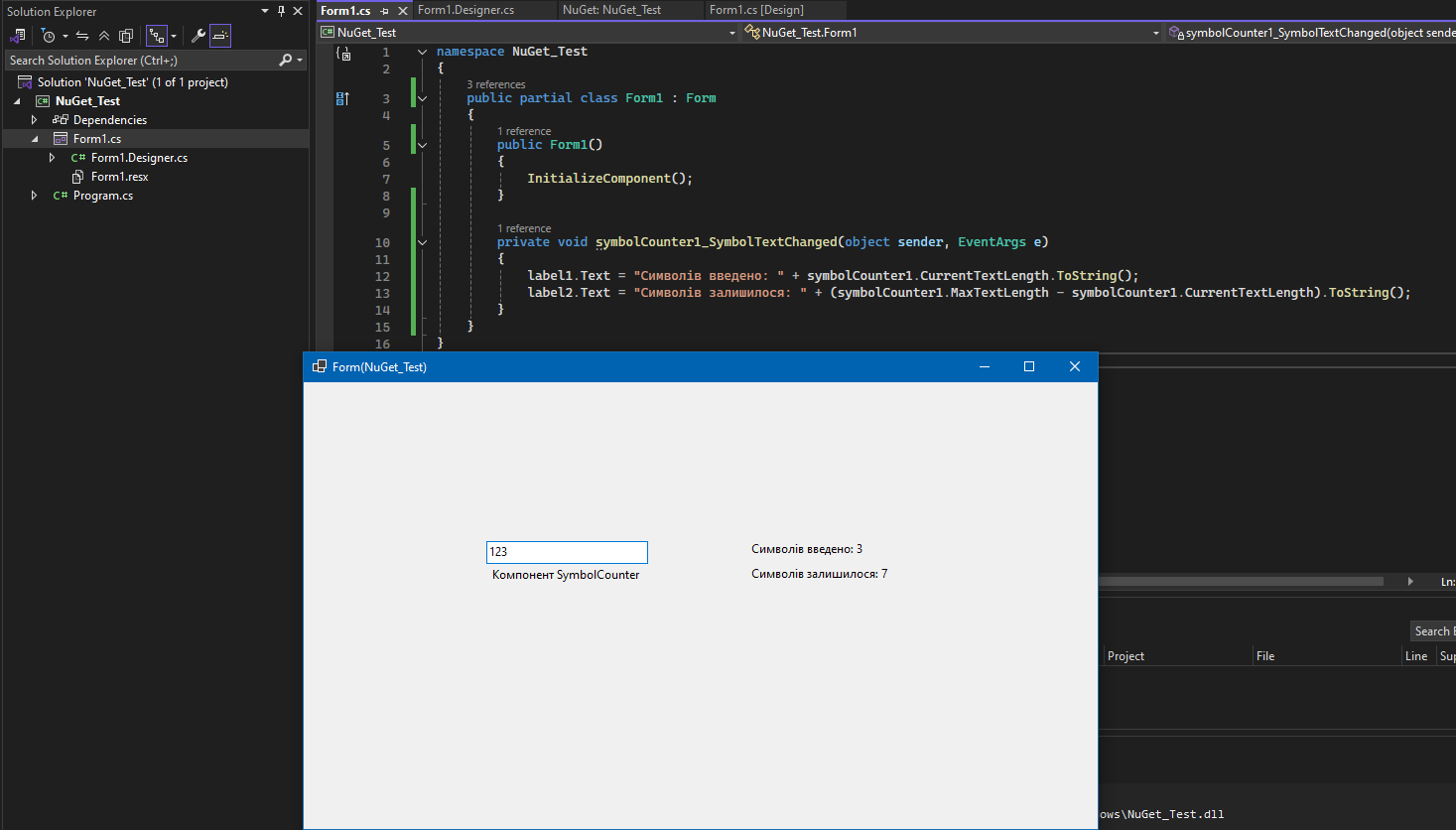


Рисунок 5 – Результат тесту 2

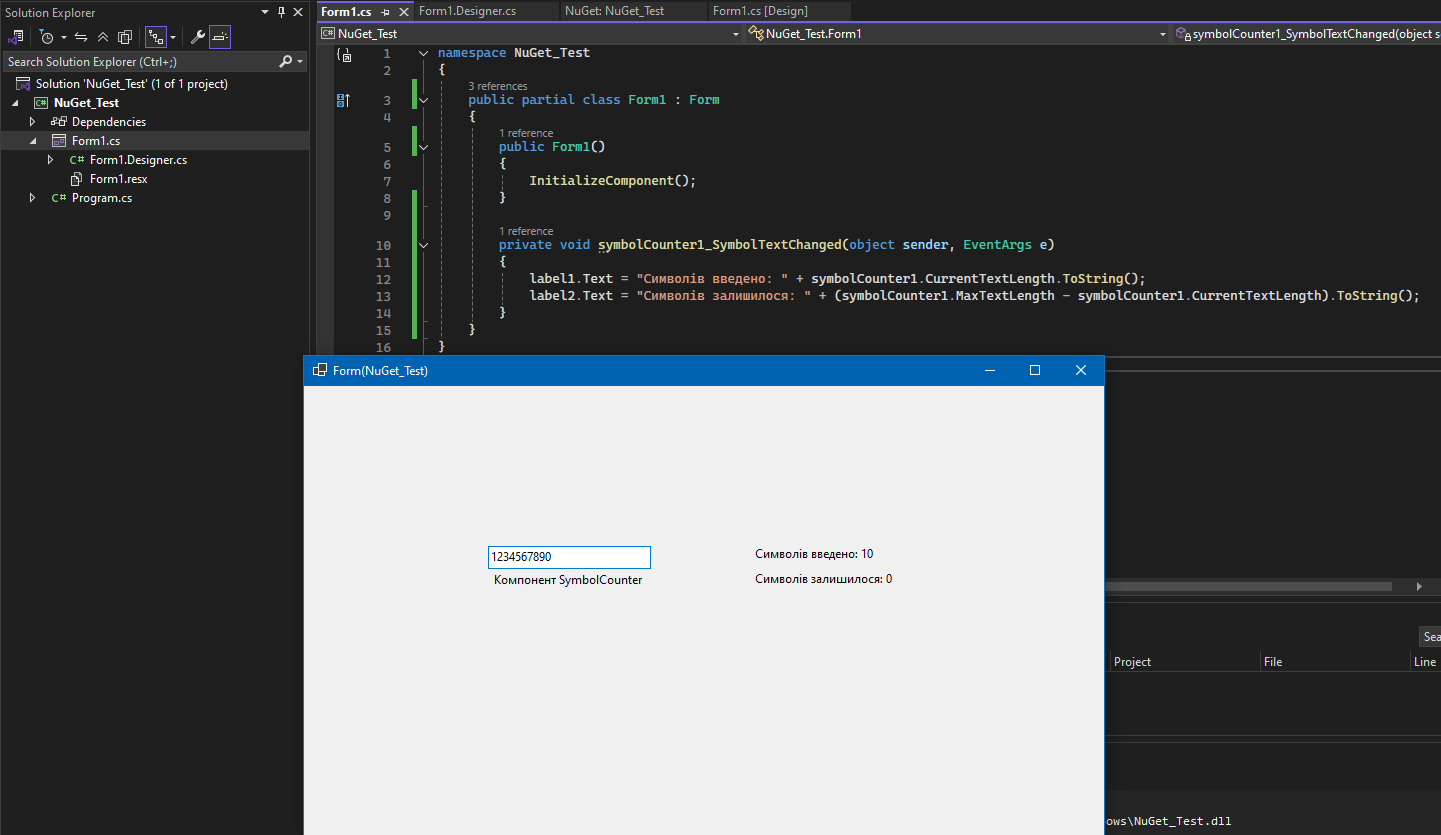


Рисунок 6 – Результат тесту 3

Таблиця 3 – самооцінка виконання вимог до роботи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Вимоги до роботи | | Бали | Відмітка про виконання |
| 1 | Базові | Реалізація компоненту та основної програми окремими проектами в рамках одного рішення (Solution) | 1 | + |
| 2 | Реалізація власних властивостей та подій компонента, що відображаються в редакторі властивостей (Properties) | 1 | + |
| 3 | Реалізація функціональності компонента згідно завдання | 1 | + |
| 4 | Пояснення у звіті по використанню кожного з принципів SOLID при створенні компоненту | 2 | + |
| 5 | Додаткові | Прив'язка піктограми до компонента для відображення на панелі інструментів | 1 | + |
| 6 | Прив'язка меж та пропорційна зміна внутрішніх елементів щодо загального розміру компонента | 1 | + |
| 7 | Створення пакету NuGet з компонентом для повторного використання. Застосування менеджеру пакетів для установки та тестування компонента в окремому рішенні (Solution). | 2 | + |
| 8 | Часу | Своєчасне виконання базових вимог та звітності | 1 | – |