**KIỂM THỬ PHẦN MỀM BUỔI 1 NHÓM 2**

**NỘI DUNG: MSUNIT và NUNIT**

STT:80

MSSV:2251052137

Họ tên: Nguyễn Ngọc Tưởng

Mục lục

[1. Khởi tạo project 1](#_Toc191373174)

[Bước 1: Tạo project (windows form app .net framework) 1](#_Toc191373175)

[Bước 2: Tạo tên project 2](#_Toc191373176)

[Bước 3: Chỉnh các thuộc tính(Properties) có trong form 2](#_Toc191373177)

[2. Thiết kế giao diện 3](#_Toc191373178)

[Bước 1: Thiết kế và các control vào form bằng Toolbox 3](#_Toc191373179)

[Bước 2: Thiết lập các thuộc tính của control bằng Properties 4](#_Toc191373180)

[3. Thêm class caculation phục vụ cho tính toán 4](#_Toc191373181)

[Add class 4](#_Toc191373182)

[Note: Mục cần chú ý trong Class Caculation: 5](#_Toc191373183)

[Nội dung trong class Caculation: 5](#_Toc191373184)

[4. Thêm các sự kiện có trong form (Form1.cs) 6](#_Toc191373185)

[5. Tạo project kiểm thử 8](#_Toc191373186)

[Bước 1: Click chuột phải Solution > Add > New Projects... 8](#_Toc191373187)

[Bước 2: Đặt tên project 9](#_Toc191373188)

[Bước 3: Tại project Unit Test, thực hiện Add Reference để tham chiếu đến project cần thực hiện Unit Test 10](#_Toc191373189)

[Code trong file UnitTest.cs 10](#_Toc191373190)

[Kết quả sau khi kiểm thử: 14](#_Toc191373191)

# 1. Khởi tạo project

## Bước 1: Tạo project (windows form app .net framework)

A screenshot of a computer

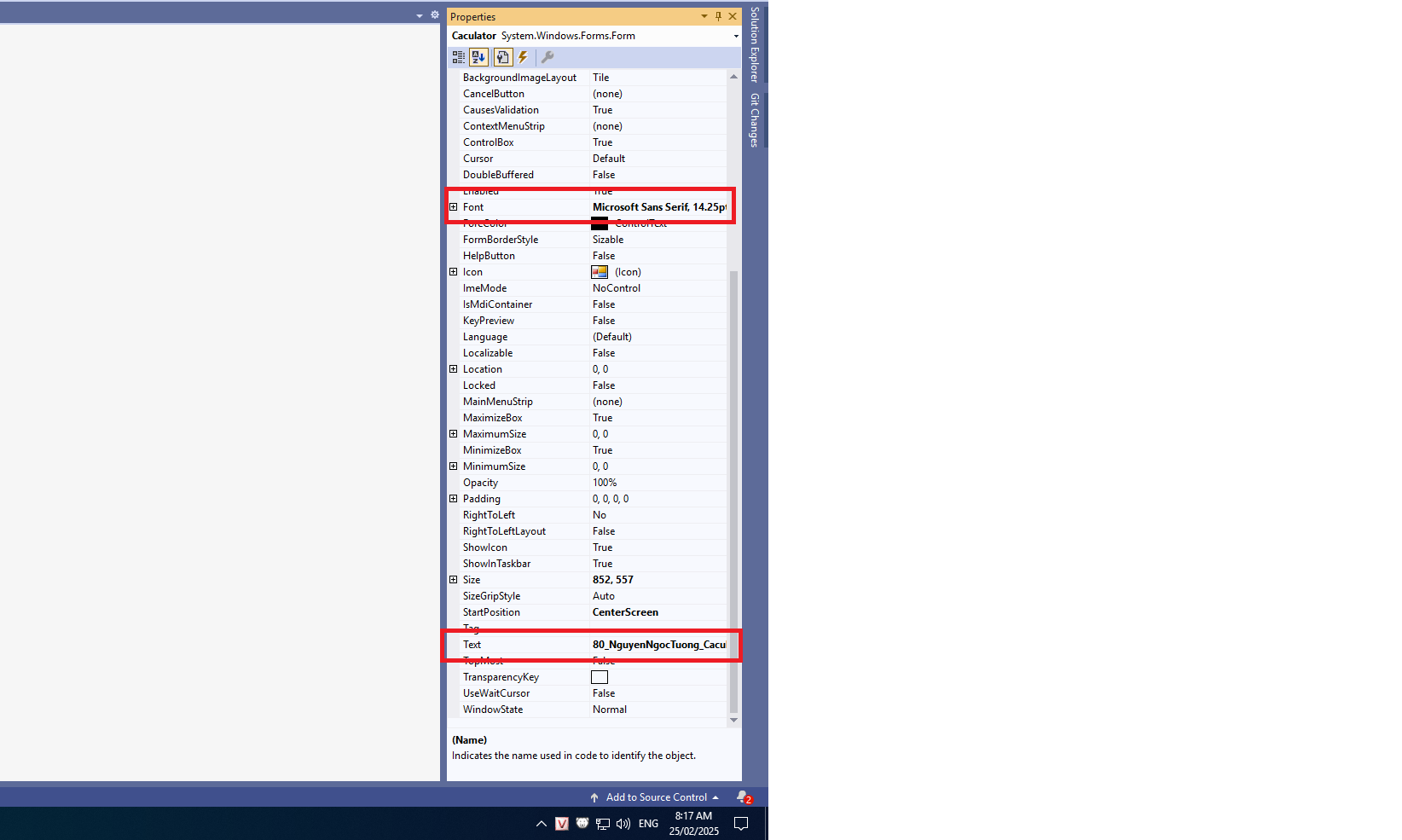
AI-generated content may be incorrect.

## Bước 2: Tạo tên project

A screenshot of a computer

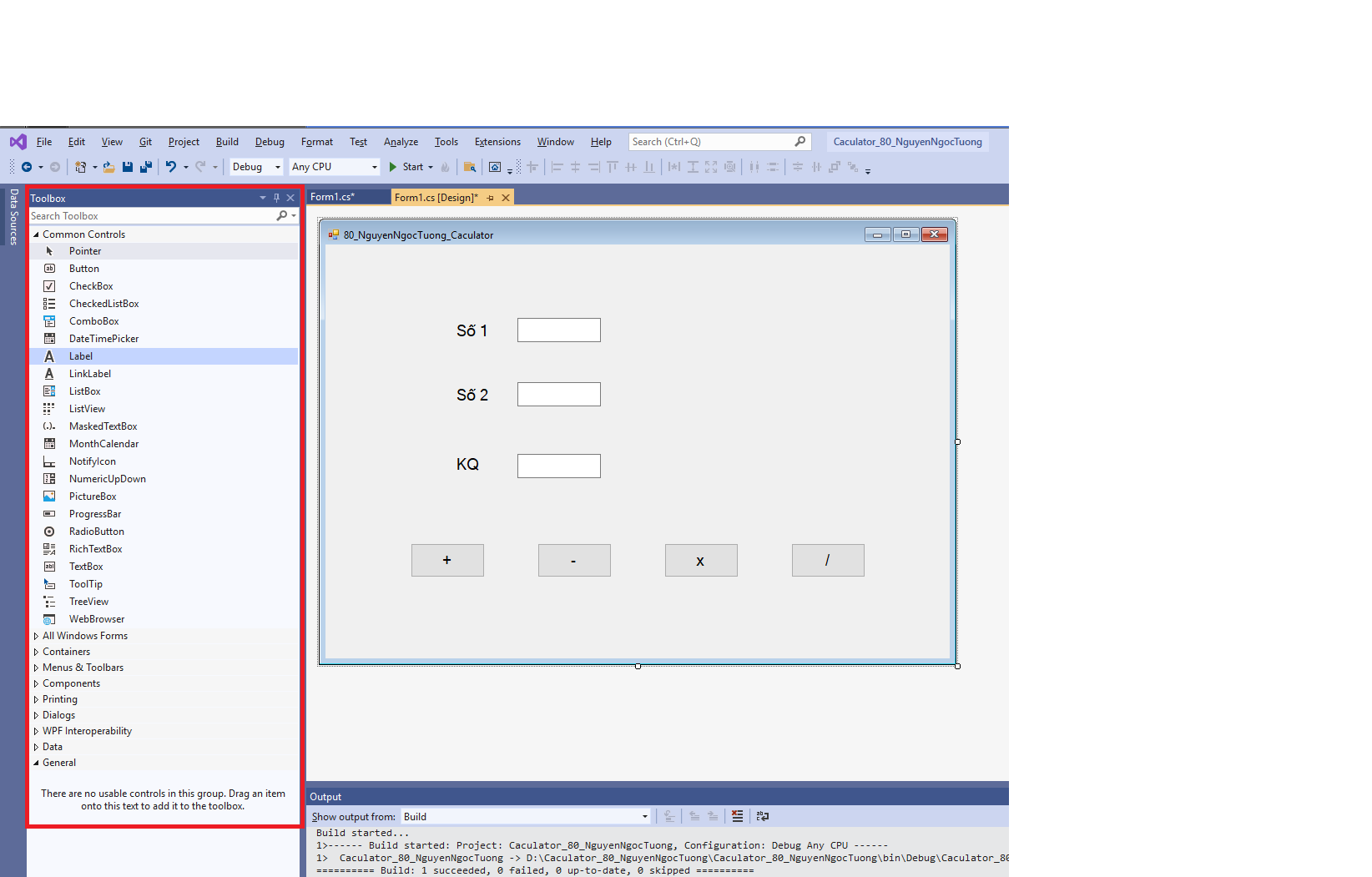
AI-generated content may be incorrect.

### Bước 3: Chỉnh các thuộc tính(Properties) có trong form

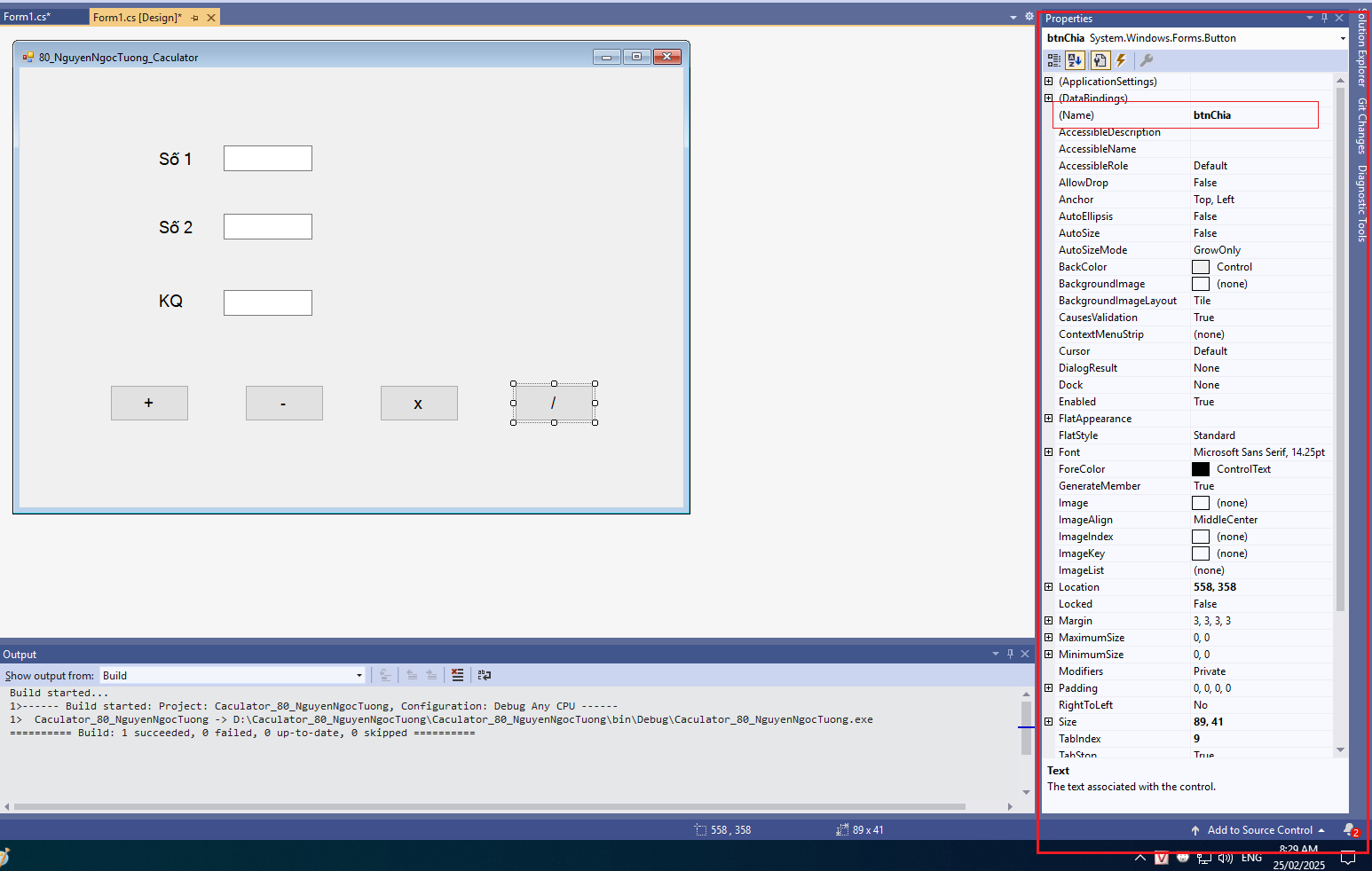


# 2. Thiết kế giao diện

## Bước 1: Thiết kế và các control vào form bằng Toolbox

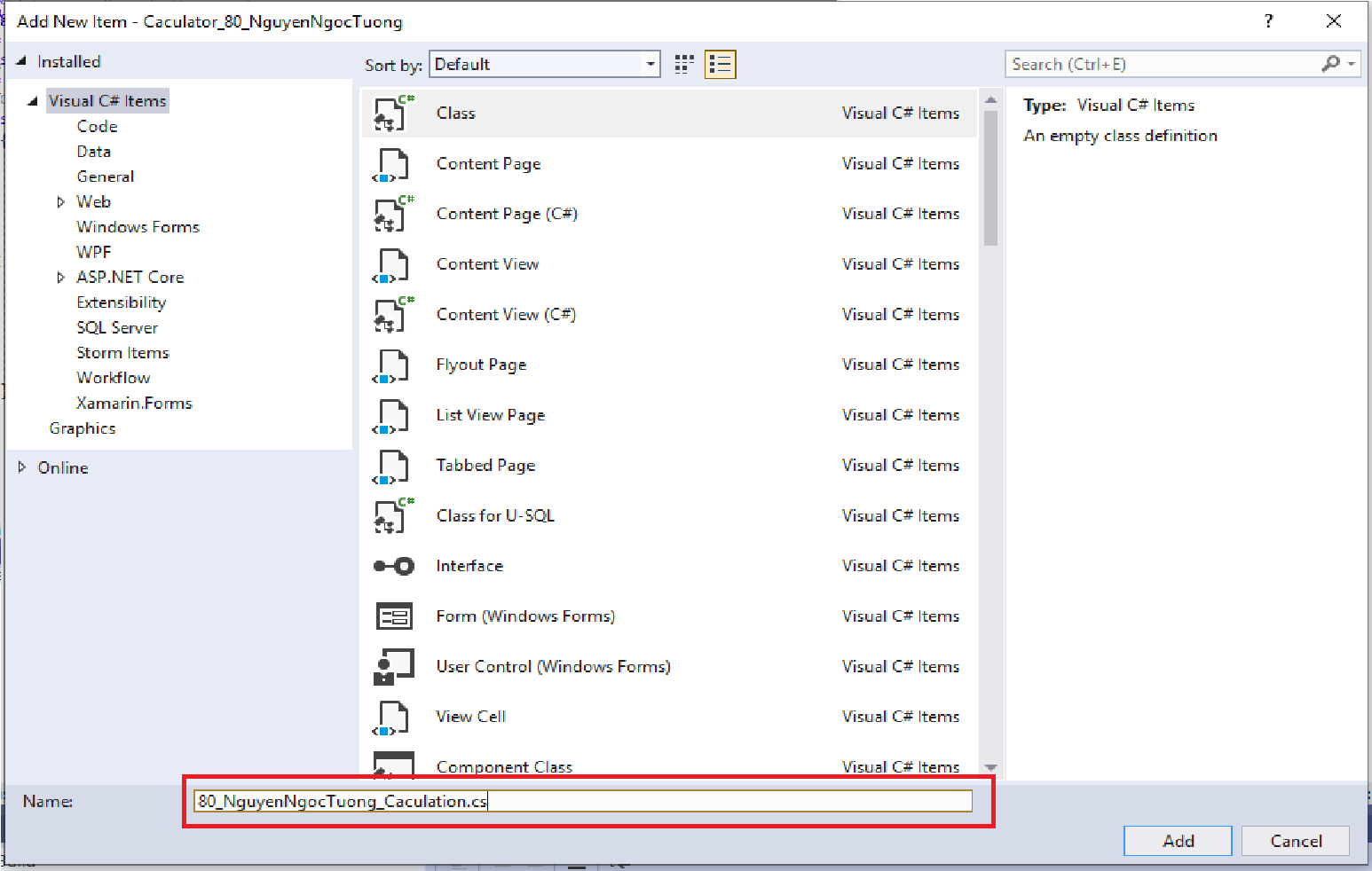


## Bước 2: Thiết lập các thuộc tính của control bằng Properties



# 3. Thêm class caculation phục vụ cho tính toán

## Add class



## Note: Mục cần chú ý trong Class Caculation:

1. Thêm từ khóa public để sử dụng lớp này trong lớp khác
2. Hàm khởi tạo trong caculation để tạo đối tượng và khởi tạo giá trị.
3. Phương thức Execute: tính toán dựa theo giá trị khởi tạo

## Nội dung trong class Caculation:

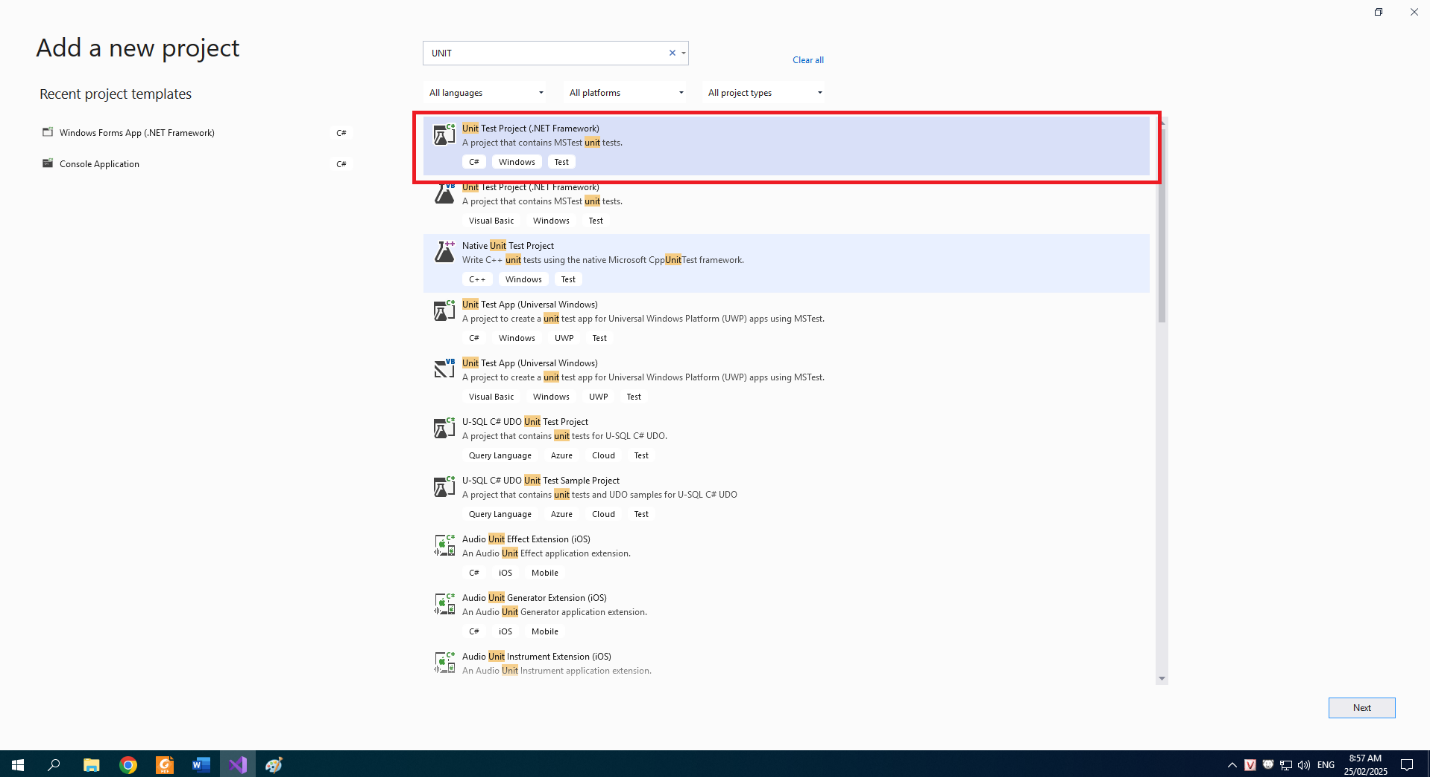
1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. namespace Caculator\_80\_NguyenNgocTuong
7. {
8. public class Caculation\_80\_NguyenNgocTuong // thêm từ khóa public để sử dụng trong lớp khác
9. {
10. private int a\_Tuong\_80, b\_Tuong\_80;
11. public Caculation\_80\_NguyenNgocTuong(int a\_Tuong\_80, int b\_Tuong\_80) // phương thức khởi tạo gán a và b tương ứng với 2 tham số được truyền vào
12. {
13. this.a\_Tuong\_80 = a\_Tuong\_80;
14. this.b\_Tuong\_80 = b\_Tuong\_80;
15. }
16. public int Execute\_80\_NguyenNgocTuong(String operation)// phương thức tính toán dựa vào toán tử được truyền vào
17. {
18. int result = 0;
19. switch (operation)//kiểm tra giá trị của toán tử để xác định phép toán
20. {
21. case "+": // thực hiện phép cộng
22. result = this.a\_Tuong\_80 + this.b\_Tuong\_80;
23. break;
24. case "-"://thực hiện phép trừ
25. result = this.a\_Tuong\_80 - this.b\_Tuong\_80;
26. break;
27. case "\*"://thực hiện phép nhân
28. result = this.a\_Tuong\_80 \* this.b\_Tuong\_80;
29. break;
30. case "/"://thực hiện phép chia
31. result = this.a\_Tuong\_80 / this.b\_Tuong\_80;
32. break;
33. }
34. return result;
35. }
36. }
37. }

# 4. Thêm các sự kiện có trong form (Form1.cs)

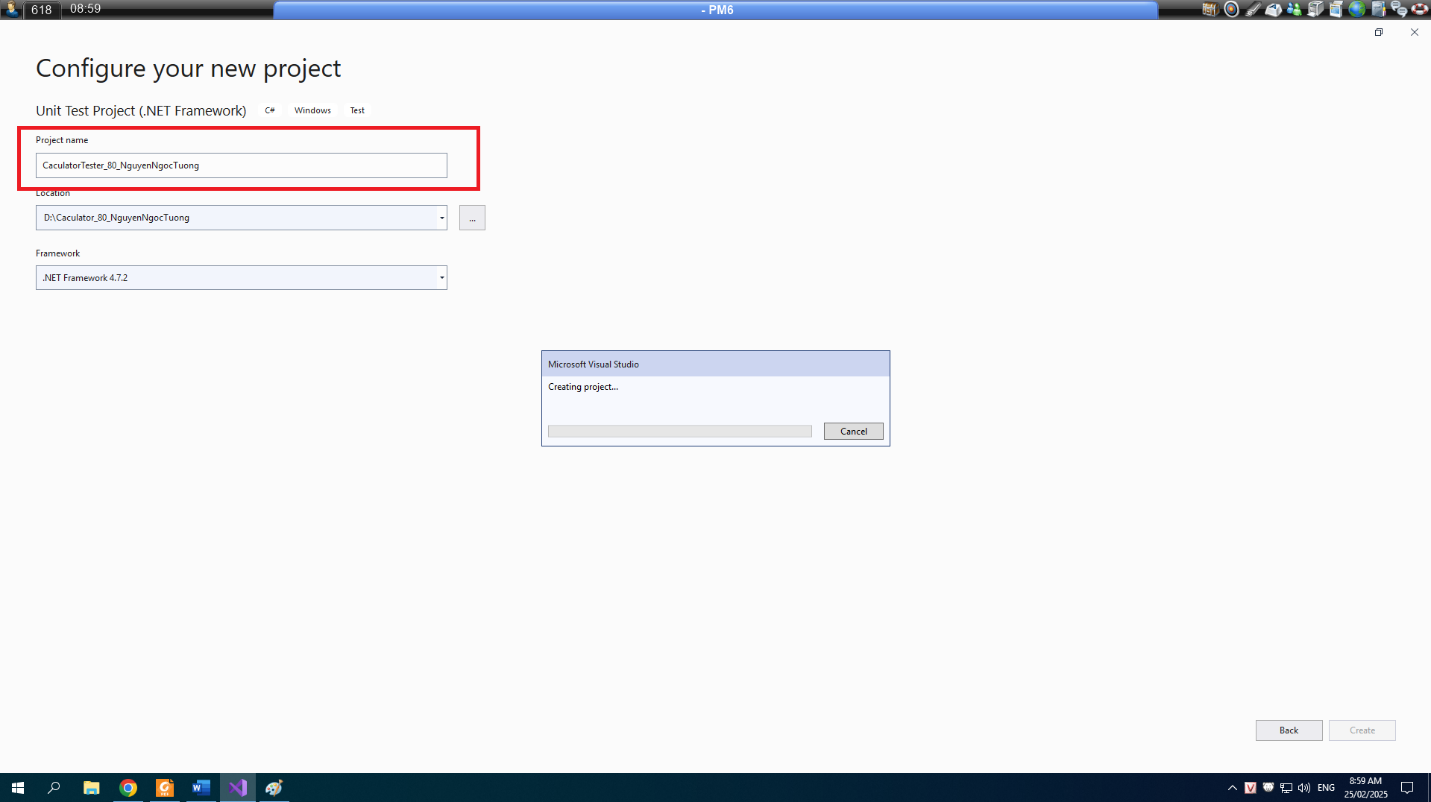
1. using System;
2. using System.Collections.Generic;
3. using System.ComponentModel;
4. using System.Data;
5. using System.Drawing;
6. using System.Linq;
7. using System.Text;
8. using System.Threading.Tasks;
9. using System.Windows.Forms;
10. namespace Caculator\_80\_NguyenNgocTuong
11. {
12. public partial class Caculator : Form
13. {
14. public Caculator()
15. {
16. InitializeComponent();
17. }
18. private void btnCong\_Tuong\_80\_Click(object sender, EventArgs e)
19. {
20. int a\_Tuong\_80, b\_Tuong\_80, relsut\_Tuong\_80;
21. a\_Tuong\_80 = int.Parse(txtSo1\_Tuong\_80.Text); // gán giá trị của a từ ô txtSo1
22. b\_Tuong\_80 = int.Parse(txtSo2\_Tuong\_80.Text);// gán giá trị của b từ ô txtSo2
23. Caculation\_80\_NguyenNgocTuong caculation = new Caculation\_80\_NguyenNgocTuong(a\_Tuong\_80, b\_Tuong\_80);// khởi tạo caculation bằng 2 tham số được truyền vào là a và b
24. relsut\_Tuong\_80 = caculation.Execute\_80\_NguyenNgocTuong("+");// kết quả được tính toán bằng phương thức Execute và toán tử truyền vào là dấu +
25. txtKQ\_Tuong\_80.Text = relsut\_Tuong\_80.ToString();//gán và hiển thị thuộc tính text bằng kết quả
26. }
27. private void btnTru\_Tuong\_80\_Click(object sender, EventArgs e)
28. {
29. int a\_Tuong\_80, b\_Tuong\_80, relsut\_Tuong\_80;
30. a\_Tuong\_80 = int.Parse(txtSo1\_Tuong\_80.Text); // gán giá trị của a từ ô txtSo1
31. b\_Tuong\_80 = int.Parse(txtSo2\_Tuong\_80.Text);// gán giá trị của b từ ô txtSo2
32. Caculation\_80\_NguyenNgocTuong caculation = new Caculation\_80\_NguyenNgocTuong(a\_Tuong\_80, b\_Tuong\_80);// khởi tạo caculation bằng 2 tham số được truyền vào là a và b
33. relsut\_Tuong\_80 = caculation.Execute\_80\_NguyenNgocTuong("-");// kết quả được tính toán bằng phương thức Execute và toán tử truyền vào là dấu -
34. txtKQ\_Tuong\_80.Text = relsut\_Tuong\_80.ToString();//gán và hiển thị thuộc tính text bằng kết quả
35. }
36. private void btnNhan\_Tuong\_80\_Click(object sender, EventArgs e)
37. {
38. int a\_Tuong\_80, b\_Tuong\_80, relsut\_Tuong\_80;
39. a\_Tuong\_80 = int.Parse(txtSo1\_Tuong\_80.Text); // gán giá trị của a từ ô txtSo1
40. b\_Tuong\_80 = int.Parse(txtSo2\_Tuong\_80.Text);// gán giá trị của b từ ô txtSo2
41. Caculation\_80\_NguyenNgocTuong caculation = new Caculation\_80\_NguyenNgocTuong(a\_Tuong\_80, b\_Tuong\_80);// khởi tạo caculation bằng 2 tham số được truyền vào là a và b
42. relsut\_Tuong\_80 = caculation.Execute\_80\_NguyenNgocTuong("\*");// kết quả được tính toán bằng phương thức Execute và toán tử truyền vào là dấu \*
43. txtKQ\_Tuong\_80.Text = relsut\_Tuong\_80.ToString();//gán và hiển thị thuộc tính text bằng kết quả
44. }
45. private void btnChia\_Tuong\_80\_Click(object sender, EventArgs e)
46. {
47. int a\_Tuong\_80, b\_Tuong\_80, relsut;
48. a\_Tuong\_80 = int.Parse(txtSo1\_Tuong\_80.Text); // gán giá trị của a từ ô txtSo1
49. b\_Tuong\_80 = int.Parse(txtSo2\_Tuong\_80.Text);// gán giá trị của b từ ô txtSo2
50. Caculation\_80\_NguyenNgocTuong caculation = new Caculation\_80\_NguyenNgocTuong(a\_Tuong\_80, b\_Tuong\_80);// khởi tạo caculation bằng 2 tham số được truyền vào là a và b
51. relsut = caculation.Execute\_80\_NguyenNgocTuong("/");// kết quả được tính toán bằng phương thức Execute và toán tử truyền vào là dấu /
52. txtKQ\_Tuong\_80.Text = relsut.ToString();//gán và hiển thị thuộc tính text bằng kết quả
53. }
54. }
55. }

# 5. Tạo project kiểm thử

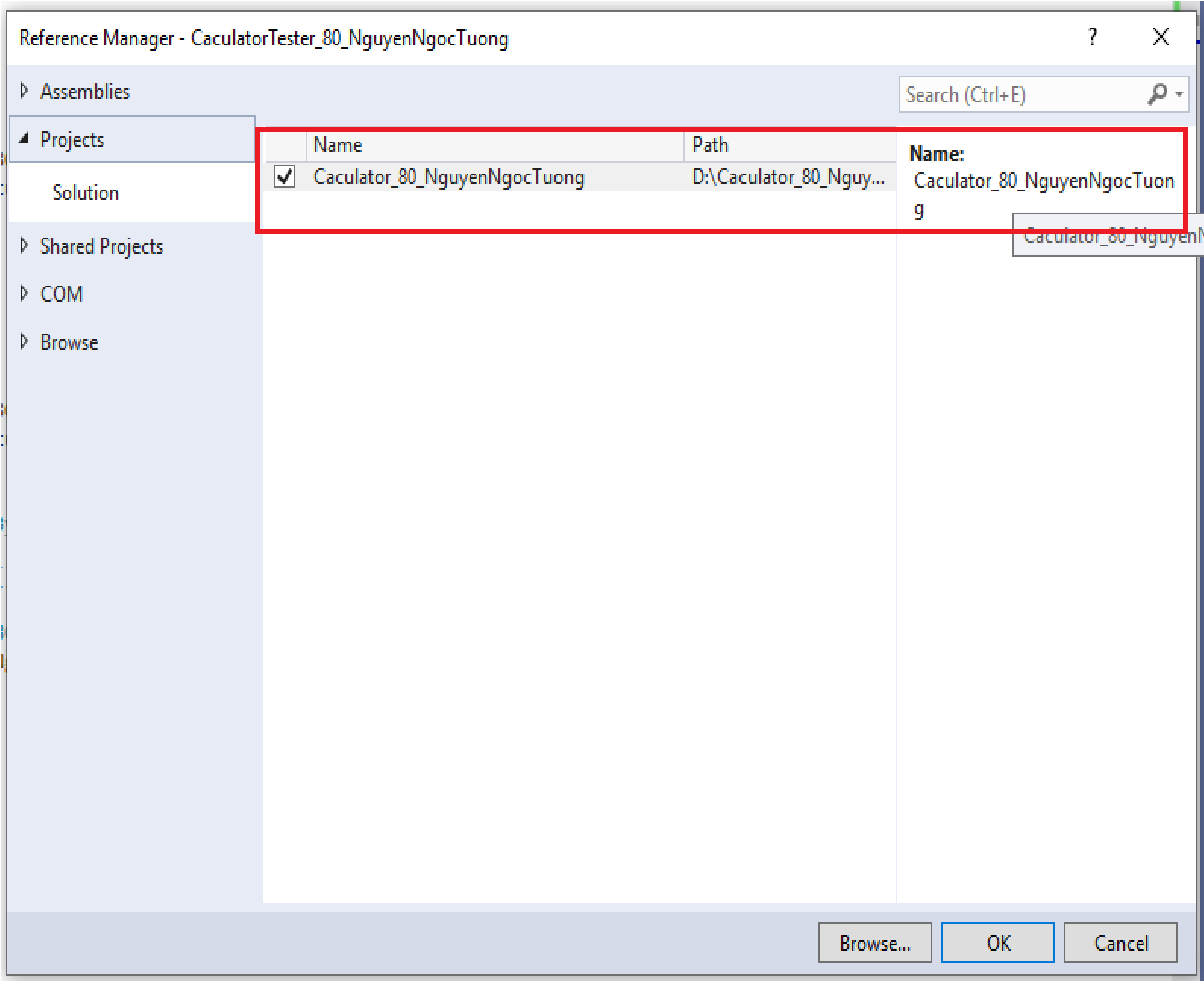
## Bước 1: Click chuột phải Solution > Add > New Projects...



## Bước 2: Đặt tên project



## Bước 3: Tại project Unit Test, thực hiện Add Reference để tham chiếu đến project cần thực hiện Unit Test



## Code trong file UnitTest.cs

1. using Microsoft.VisualStudio.TestTools.UnitTesting;
2. using System;
3. using Caculator\_80\_NguyenNgocTuong;
4. namespace CaculatorTester\_80\_NguyenNgocTuong
5. {
6. [TestClass]
7. public class UnitTest1
8. {
9. private Caculation\_80\_NguyenNgocTuong caculation\_Tuong\_80;
10. [TestMethod]
11. [TestInitialize] // thiết lập thẻ dùng chung cho test case
12. public void setup()
13. {
14. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(12, 6);
15. }
16. [TestMethod] // test case 1: a = 12, b = 6, expected =18, actual = 18 --> true
17. public void TestCase1\_AddOperator\_Tuong\_80()
18. {
19. int expected\_Tuong\_80, actual\_Tuong\_80;
20. expected\_Tuong\_80 = 18;
21. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
22. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
23. }
24. [TestMethod] // test case 2: a = 12, b = -6, expected =6, actual = 6 --> true
25. public void TestCase2\_AddOperator\_Tuong\_80()
26. {
27. int expected\_Tuong\_80, actual\_Tuong\_80;
28. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(12, -6);
29. expected\_Tuong\_80 = 6;
30. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
31. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
32. }
33. [TestMethod] // test case 3: a = -12, b = -6, expected =-18, actual = -18 --> true
34. public void TestCase3\_AddOperator\_Tuong\_80()
35. {
36. int expected\_Tuong\_80, actual\_Tuong\_80;
37. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(-12, -6);
38. expected\_Tuong\_80 = -18;
39. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
40. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
41. }
42. [TestMethod] // test case 4: a = -1, b = 0, expected =-1, actual = -1 --> true
43. public void TestCase4\_AddOperator\_Tuong\_80()
44. {
45. int expected\_Tuong\_80, actual\_Tuong\_80;
46. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(-1, 0);
47. expected\_Tuong\_80 = -1;
48. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
49. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
50. }
51. [TestMethod] // test case 5: a = 5, b = 5, expected = 7, actual = 10
52. public void TestCase5\_AddOperator\_Tuong\_80()
53. {
54. int expected\_Tuong\_80, actual\_Tuong\_80;
55. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(5, 5);
56. expected\_Tuong\_80 = 7;
57. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
58. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
59. }
60. [TestMethod] // test case 6: a = -5, b = 5, expected = -1, actual = 0
61. public void TestCase6\_AddOperator\_Tuong\_80()
62. {
63. int expected\_Tuong\_80, actual\_Tuong\_80;
64. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(-5, 5);
65. expected\_Tuong\_80 = -1;
66. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
67. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
68. }
69. [TestMethod] // test case 7: a = 0, b = -5, expected = 1, actual = -5
70. public void TestCase7\_AddOperator\_Tuong\_80()
71. {
72. int expected\_Tuong\_80, actual\_Tuong\_80;
73. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(0, -5);
74. expected\_Tuong\_80 = 1;
75. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
76. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
77. }
78. [TestMethod] // test case 8: a = 0, b = 0, expected = -5, actual = 0
79. public void TestCase8\_AddOperator\_Tuong\_80()
80. {
81. int expected\_Tuong\_80, actual\_Tuong\_80;
82. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(0, 0);
83. expected\_Tuong\_80 = -5;
84. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
85. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
86. }
87. [TestMethod] // test case 9: a = -9, b = -9, expected = -5, actual = -18
88. public void TestCase9\_AddOperator\_Tuong\_80()
89. {
90. int expected\_Tuong\_80, actual\_Tuong\_80;
91. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(-9, -9);
92. expected\_Tuong\_80 = -5;
93. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("+");
94. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
95. }
96. [TestMethod]//test case tru: a = 12, b = 6, expected = 6, actual = 6 --> true
97. public void TestSubOperator\_Tuong\_80()
98. {
99. int expected\_Tuong\_80, actual\_Tuong\_80;
100. expected\_Tuong\_80 = 6;
101. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("-");
102. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
103. }
104. [TestMethod]//test case nhan: a = 12, b = 6, expected = 72, actual = 72 --> true
105. public void TestMulOperator\_Tuong\_80()
106. {
107. int expected\_Tuong\_80, actual\_Tuong\_80;
108. expected\_Tuong\_80 = 72;
109. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("\*");
110. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
111. }
112. [TestMethod]//test case chia: a = 12, b = 6, expected = 2, actual = 2 --> true
113. public void TestDivOperato\_Tuong\_80()
114. {
115. int expected\_Tuong\_80, actual\_Tuong\_80;
116. expected\_Tuong\_80 = 2;
117. actual\_Tuong\_80 = caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("/");
118. Assert.AreEqual(expected\_Tuong\_80, actual\_Tuong\_80);
119. }
120. [TestMethod]//test case chia voi so 0
121. [ExpectedException(typeof(DivideByZeroException))]
122. // Chỉ định rằng kiểm thử này kỳ vọng sẽ ném ra một ngoại lệ loại DivideByZeroException
123. public void TestDivOperatorByZero\_Tuong\_80()
124. {
125. caculation\_Tuong\_80 = new Caculation\_80\_NguyenNgocTuong(14, 0);
126. // Gọi phương thức Execute\_80\_NguyenNgocTuong với tham số "/" để thực hiện phép chia.
127. // Vì b = 0, phương thức này sẽ ném ra ngoại lệ DivideByZeroException.
128. caculation\_Tuong\_80.Execute\_80\_NguyenNgocTuong("/");
129. }
130. }
131. }

## Kết quả sau khi kiểm thử:

A screenshot of a computer

AI-generated content may be incorrect.