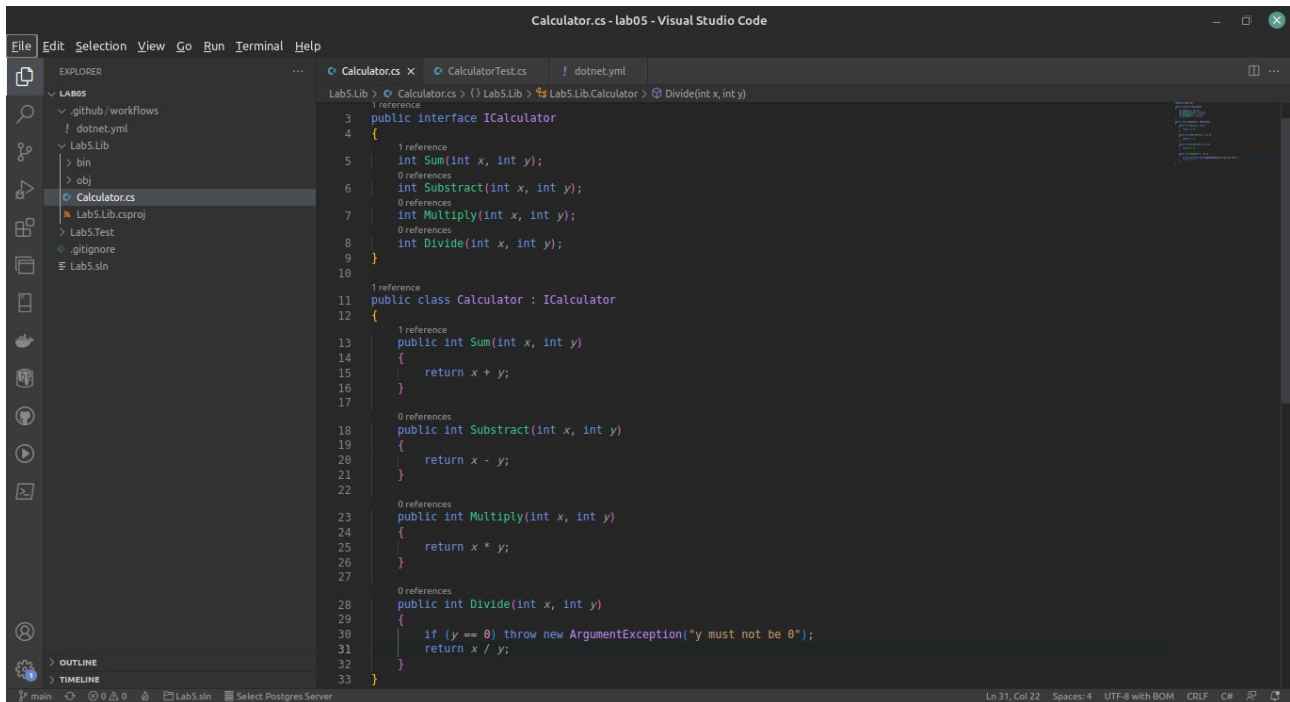


# Pipelines

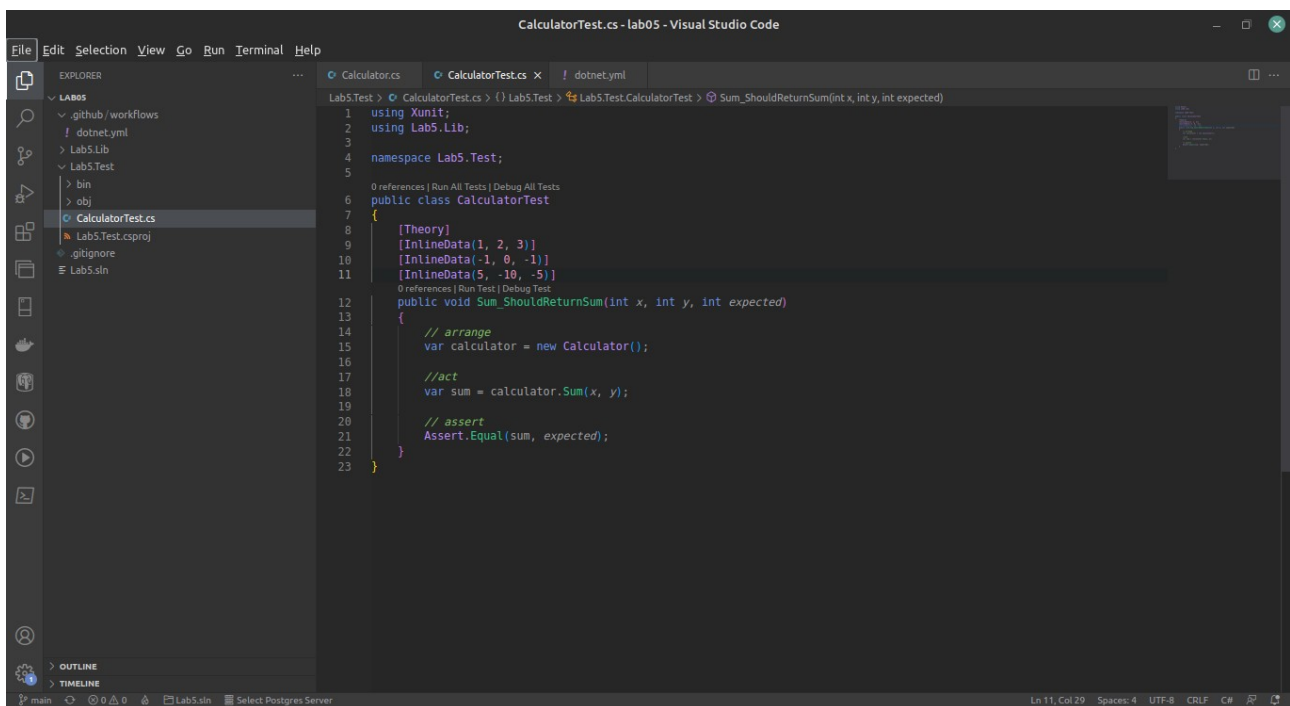
1. Utworzyłem projekt w środowisku .NET zawierający prostą klasę reprezentującą kalkulator.



The screenshot shows the Visual Studio Code interface with the 'Calculator.cs' file open in the editor. The file is part of the 'Lab5.Lib' project. The code defines an interface 'ICalculator' with methods 'Sum', 'Subtract', 'Multiply', and 'Divide'. A class 'Calculator' implements these methods. The 'Divide' method includes a check for division by zero, throwing an 'ArgumentException' if the divisor is zero.

```
3 public interface ICalculator
4 {
5     1 reference
6     int Sum(int x, int y);
7     0 references
8     int Subtract(int x, int y);
9     0 references
10    int Multiply(int x, int y);
11    0 references
12    int Divide(int x, int y);
13 }
14
15 1 reference
16 public class Calculator : ICalculator
17 {
18     1 reference
19     public int Sum(int x, int y)
20     {
21         return x + y;
22     }
23
24     0 references
25     public int Subtract(int x, int y)
26     {
27         return x - y;
28     }
29
30     0 references
31     public int Multiply(int x, int y)
32     {
33         return x * y;
34     }
35
36     0 references
37     public int Divide(int x, int y)
38     {
39         if (y == 0) throw new ArgumentException("y must not be 0");
40         return x / y;
41     }
42 }
```

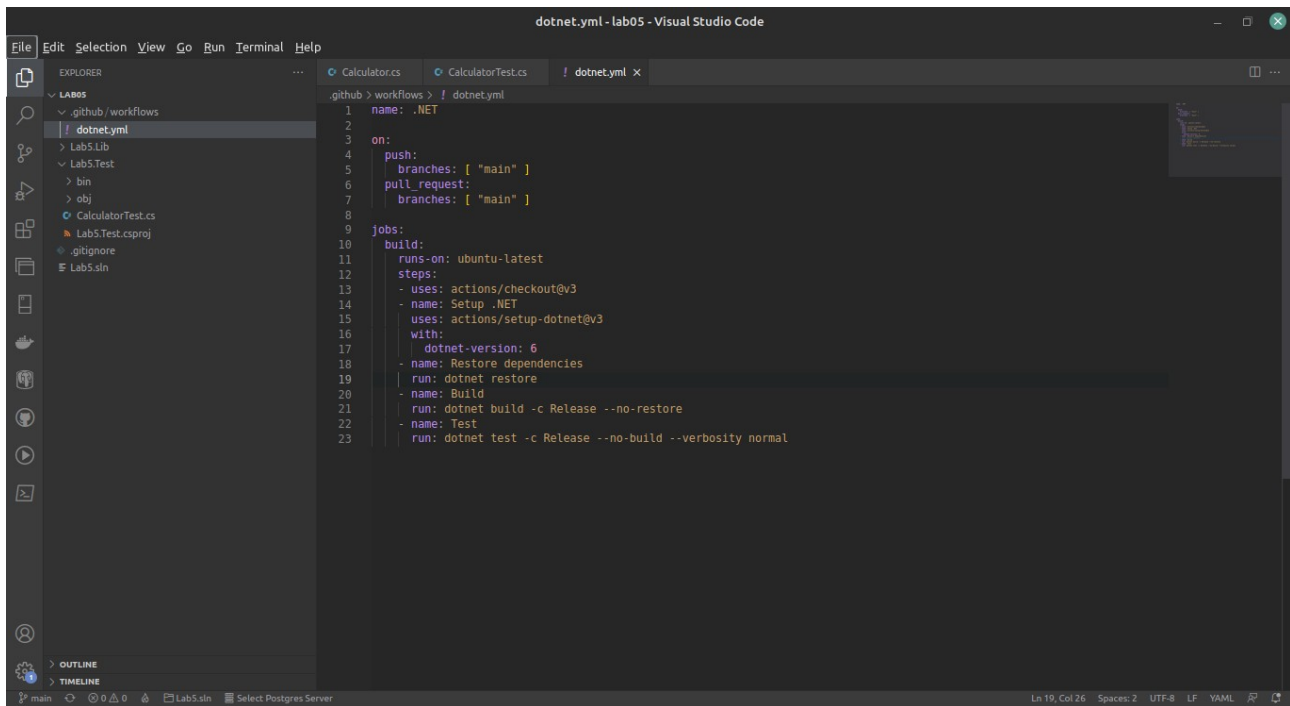
2. Do klasy z biblioteki klas utworzyłem klasę testującą.



The screenshot shows the Visual Studio Code interface with the 'CalculatorTest.cs' file open in the editor. The file is part of the 'Lab5.Test' project. The code uses 'Xunit' for testing. It defines a test class 'CalculatorTest' with a theory 'Sum\_ShouldReturnSum' that tests the 'Sum' method of the 'Calculator' class with various inputs and expected results.

```
1 using Xunit;
2 using Lab5.Lib;
3
4 namespace Lab5.Test;
5
6 0 references | Run All Tests | Debug All Tests
7 public class CalculatorTest
8 {
9     [Theory]
10    [InlineData(1, 2, 3)]
11    [InlineData(-1, 0, -1)]
12    [InlineData(5, -10, -5)]
13    0 references | Run Test | Debug Test
14    public void Sum_ShouldReturnSum(int x, int y, int expected)
15    {
16        // arrange
17        var calculator = new Calculator();
18
19        // act
20        var sum = calculator.Sum(x, y);
21
22        // assert
23        Assert.Equal(sum, expected);
24    }
25 }
```

3. Następnie do projektu stworzyłem pipeline CI korzystający z GitHub Actions, który buduje i testuje moje rozwiązanie.



```
1 name: .NET
2
3 on:
4   push:
5     branches: [ "main" ]
6   pull_request:
7     branches: [ "main" ]
8
9 jobs:
10  build:
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v3
14      - name: Setup .NET
15        uses: actions/setup-dotnet@v3
16        with:
17          dotnet-version: 6
18      - name: Restore dependencies
19        run: dotnet restore
20      - name: Build
21        run: dotnet build -c Release --no-restore
22      - name: Test
23        run: dotnet test -c Release --no-build --verbosity normal
```