## Zadanie: Analiza Tablic i Testy Jednostkowe NUnit

Twoim zadaniem jest stworzenie biblioteki klas realizującej obliczenia statystyczne na zbiorach liczb całkowitych oraz przygotowanie projektu testowego w celu weryfikacji poprawności działania kodu.

### 1. Specyfikacja klasy ArrayAnalyzer

Utwórz klasę ArrayAnalyzer w projekcie typu *Class Library* (.NET), która zawierać będzie następujące metody:

* **ObliczSrednia(int[] tablica)**:
  + Przyjmuje jako parametr tablicę liczb całkowitych.
  + Zwraca wartość średnią (typ double) wszystkich elementów.
  + **Warunek**: Jeśli przekazana tablica jest pusta (Length == 0) lub ma wartość null, metoda musi rzucić wyjątek ArgumentException z komunikatem "Tablica nie może być pusta".
* **LiczbaWiekszychNizSrednia(int[] tablica)**:
  + Przyjmuje jako parametr tablicę liczb całkowitych.
  + Wykorzystuje metodę ObliczSrednia do wyznaczenia progu.
  + Zwraca liczbę elementów (typ int), których wartość jest ściśle większa od obliczonej średniej.
  + Jeśli tablica jest pusta, metoda powinna zwrócić 0 (lub obsłużyć wyjątek z poprzedniej metody).

### 2. Projekt Testowy (NUnit)

Utwórz projekt testowy typu **NUnit Test Project**. W klasie testowej zaimplementuj następujące przypadki testowe:

* **Test standardowy**: Sprawdzenie średniej dla zestawu danych, gdzie wynik jest liczbą całkowitą.
* **Test zmiennoprzecinkowy**: Sprawdzenie średniej dla danych dających wynik po przecinku.
* **Test asercji wyjątku**: Weryfikacja, czy podanie pustej tablicy skutkuje rzuceniem błędu ArgumentException.
* **Testy parametryzowane**: Użycie zestawów danych do sprawdzenia metody liczącej elementy większe od średniej.

## Rozwiązanie

### Kod Logiki (ArrayAnalyzer.cs)

C#

using System;  
  
namespace StatystykiLiczbowe  
{  
 public class ArrayAnalyzer  
 {  
 public double ObliczSrednia(int[] tablica)  
 {  
 if (tablica == null || tablica.Length == 0)  
 {  
 throw new ArgumentException("Tablica nie może być pusta");  
 }  
  
 double suma = 0;  
 foreach (int liczba in tablica)  
 {  
 suma += liczba;  
 }  
 return suma / tablica.Length;  
 }  
  
 public int LiczbaWiekszychNizSrednia(int[] tablica)  
 {  
 if (tablica == null || tablica.Length == 0) return 0;  
  
 double srednia = ObliczSrednia(tablica);  
 int licznik = 0;  
  
 foreach (int liczba in tablica)  
 {  
 if (liczba > srednia)  
 {  
 licznik++;  
 }  
 }  
 return licznik;  
 }  
 }  
}

### Wykonaj testy jednostkowe

### Kluczowe elementy NUnit do zapamiętania:

1. **[TestFixture]**: Oznacza klasę zawierającą testy.
2. **[Test]**: Oznacza pojedynczą metodę testową.
3. **[SetUp]**: Metoda, w której przygotowujesz środowisko (np. tworzysz instancję klasy \_analyzer), aby nie powtarzać kodu w każdym teście.
4. **Assert.Throws<T>(() => kod)**: Kluczowa konstrukcja w NUnit do sprawdzania, czy program poprawnie reaguje na błędy (np. pustą tablicę).
5. **[TestCase]**: Pozwala na uruchomienie tej samej metody testowej dla wielu różnych zestawów danych, co oszczędza mnóstwo czasu na egzaminie.