# Lab: Unit Testing Arrays and Lists

Tasks for exercise in class and for homework to the course  ["Programming Fundamentals and Unit Testing" @ SoftUni](https://softuni.bg/trainings/4256/programming-fundamentals-and-unit-testing-september-2023).

# Unit Test Array: Reverse Array

Look at the **provided skeleton** and examine the Reverse.cs class that you will test:

A screenshot of a computer

Description automatically generated

The method takes in an **integer array**, **reverses it**, and returns a **string** with the **reversed numbers**,

A close-up of text

Description automatically generated

Then, look at the tests inside the ReverseTests.cs class:

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generatedThe first test is **finished** so you can have a **reference**, the second test is **unfinished**, your job is to finish writing it, and the final test is **empty,** and your task is to finish it. The tests should run when you're finished:

A screenshot of a computer code

Description automatically generated

# Unit Test Array: Even Odd Subtraction

Test a given method which takes in an **array of integers** and finds the **difference** between the **sum** of **even numbers** and the **sum** of **odd numbers**.

The method is found in the EvenOddSubtraction.cs file:

A screenshot of a computer code

Description automatically generated

You are given again a **test** **file** EvenOddSubtractionTests.cs which contains **4 tests**. One of them has been **finished**, one is **finished partially**, and **two** are **empty** for you to finish:

A screenshot of a computer

Description automatically generatedFinish the two tests and using the **names of the other two** finish writing the unit tests.

When you are ready make sure your **tests run:**A screenshot of a computer

Description automatically generated

# Unit Test Array: Average

For the final array problem, test a method which takes in an **array of integers** and finds the **average** of the **sum** of numbers.

The method is found in the Average.cs file:

A screenshot of a computer program

Description automatically generated

Notice the method will **throw an exception** if the array is **empty**.

In the **test** **file** AverageTests.cs you have **5 tests**. Two of them are **unfinished** and three are **empty**. Once again, it's your task to finish the tests and run them:A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

# Unit Test List: Gauss Trick

For the first **list problem**, test a method which takes in a **list** **of integers** and sums each **pair of numbers** from the beginning and end.

The method is found in the GaussTrick.cs file:

A screenshot of a computer code

Description automatically generated

Example: If we have the numbers 1 2 3 4 5, the method will first **sum 1 and 5**, then **2 and 4**, and finally because the list is **odd** it will **append** **3** to the result. The final list will be 6 6 3.

In the **test** **file** GaussTrickTests.cs you are given **6 tests**. One is **finished**, two are **unfinished** and three are **empty**:

A screenshot of a computer

Description automatically generated

Hint: Use CollectionAssert to assert **correctness** when using **lists**. **The first test is an example of it**.

Don't forget to also **assert** if your **tests** **run**:

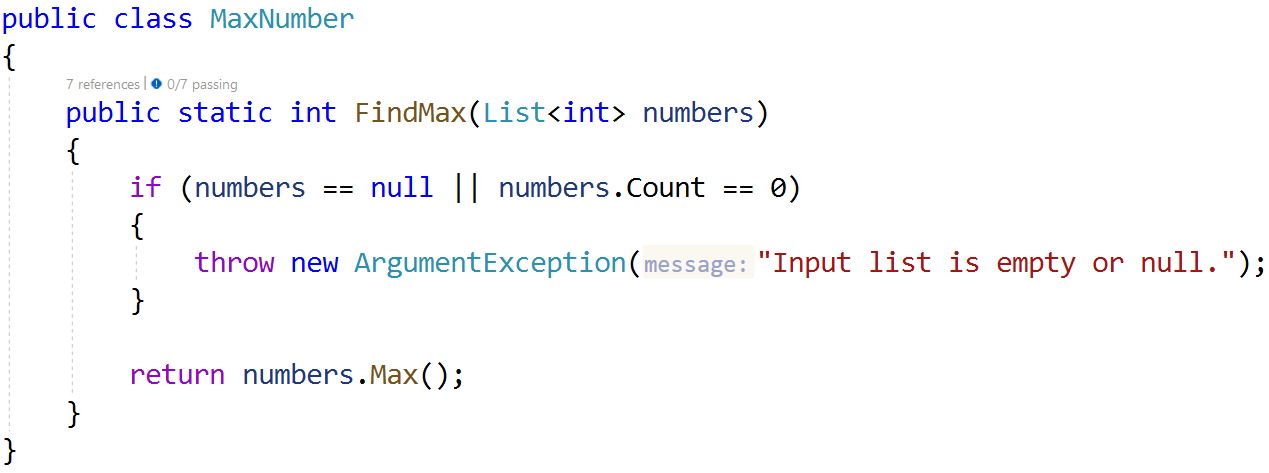
A screenshot of a computer

Description automatically generated

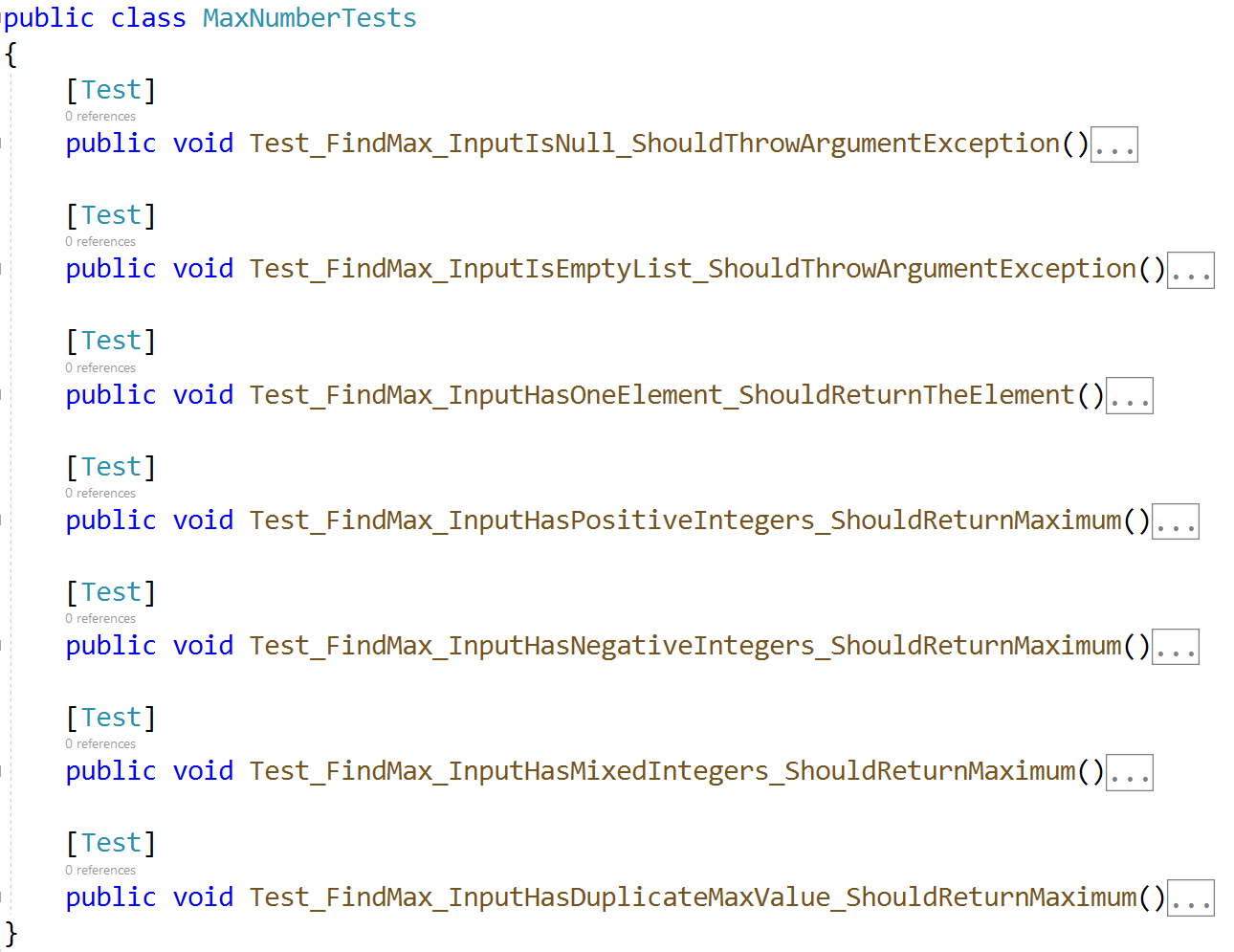
# Unit Test List: Max Number

For the second **list problem**, test a method which takes in a **list** **of integers** and returns the **largest number in the list**.

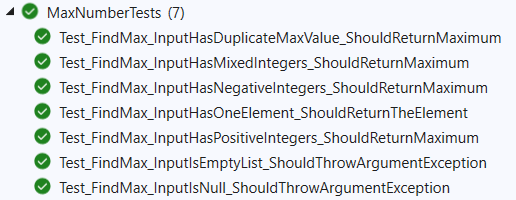
The method is found in the MaxNumber.cs file:



In the **test** **file** MaxNumberTests.cs you are given **7 tests**. Two are **unfinished** and five are **empty**:



As usual check if your tests run correctly:



# Unit Test List: Adjacent Equal

For the last **list problem**, test a method which takes in a **list** **of integers** and **sums each adjacent number if they are equal** and returns the **list as a string**.

The method is found in the AdjacentEqual.cs file:

A computer screen shot of a program

Description automatically generated

In the **test** **file** AdjacentEqualTests.cs you are given **8 tests**. Three are **unfinished** and five are **empty**:

A screenshot of a computer program

Description automatically generated

A close up of a number

Description automatically generated

As usual check if your tests run correctly:

A screenshot of a computer

Description automatically generated

Finally make sure all your tests run:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated