Mahir Efe KAYA
2016400195

CMPE 436
Assignment 2

JUnit Testing

Problem Description:

We are asked to create a test for a faulty code, and then to debug it. And afterwards creating a corrected version of it. In bonus 1, we are asked to suggest better implementation for the other faulty parts that has not been given in the requirements. In bonus 2, we are asked to implemented an automated rounding algorithm.

Tests:

For the requirements, I have separated them as questions such as:

Q1. For the requirement "Unless explicitly stated otherwise, a new RunningAverage must start with a population size of zero (0) and thereby the current average must also be zero",I have created two tests as for:
* a default constructor
*and a explicit constructor with given parameters.

Q2. For the requirement "addElements(List) and removeElements(List) functions", I created two tests, one for adding elements and one for removing elements. As for the subelements of this requirement, I managed to test them at once.

To control if adding or removing works, I added an exception catcher if it does give an error and check the current average afterwards, if it indeed did the task. For the list's preservation, I copied the list to a another variable and compared it afterwards. Then checked the return value, if it does give the same result as the getcurrentAverage() function gives and an calculated value. I controlled the other two elements as it is suggested in requirements.

Q3.For the requirement " You may assume that the initial population size is NOT negative.", I called the constructor with a negative value for size to check. It did not throw any exception, so I corrected it so that it would throw an InvalidValueException.

Q4. For the requirement ""The population size can never be negative.", I removed several elements from a 1-sized Running Average to check.
Q5. "The combine(final RunningAverage, final RunningAverage) function", check if the exception is thrown, if the size has changed, and if the value has been updated properly to the right value.

Bonus 1(Suggestion): The values that has been added shall be recorded in list, so that the values that are to be deleted can be checked. In cases of non-existant values removed, the size value can be dropped either to a negative value or to zero while having its average value non-zero could be a huge problem in the current implementation. With the list, if a non-existant value is wanted to be removed, an exception can be thrown.

The Corrected Faults:

- Default Constructor initializes with the size as 1. (Corrected it to be 0)
- When one removes an empty list program is returning zero, not the current average. (Corrected)
- One can initialize an Running Average object with size negative. (An exception is thrown)
- One can attain a Running Average object by removing more than its size. (An exception is thrown)
- Combine function lacks one pair of parantheses in calculation. (Added them)