Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki	
Technologie obiektowe - projekt	
Temat: Porównanie rozwiązań związanych z testowaniem	Przemysław Pyk 1ID21B Krzysztof Siczek 1ID12B
Numer projektu: 41	

1. Wstęp

Celem projektu było stworzenie porównania rozwiązań związanych z testowaniem. Projekt daje mozliwość testowania wzorca projektowego Buildera w języku Java oraz C#. Tematem Buildera są raporty w firmie zajmującej się sprzedażą aut. Składa się on z 13 klas takich jak na przykład Salesman, Vehicle, Cars, Employee, Report, ReportPattern. Do testów wykorzystano AssertJ EasyMock, AssertJ Mockito w Javie. W C# NUnit JustMock, NUnit Moq oraz Xunit JustMock, Xunit Moq. Wstępnie testy były pisanie w JustMocku oraz Typemocku, lecz zrezygnowaliśmy z Typemocka, ponieważ wymagał on płatności, a darmowa wersja nie działała w odpowiedni sposób.

2. Użyte technologie

NUnit vs xUnit.net

Nunit został wprowadzony w roku 2002 i nadal jest szeroko stosowany, bardzo dobrze udokumentowany i posiada duże community użytkowników, gdzie xUnit.net jest bardziej nowoczesny, bardziej rozszczerzalny i zyskuje na popularności w .NET Core.

Główna różnica polega na sposobie w jaki xUnit.net testuje metody.

W NUnit mamy klasę testową i zbiór metod testowych w niej. NUnit tworzy nową instancję klasy testowej, a następnie uruchamia wszystkie metody testowe w tym samym momencie.

Gdzie xUnit.net tworzy nową instancję klasy testowej dla każdej z metod testowych. W związku z tym nie można używać pól ani właściwości do udostępniania danych między metodami testowymi, co jest złą praktyką, ponieważ metody testowe byłyby od siebie zależne, co jest niedopuszczalne w TDD (Test Driven Development). Dlatego używając Xunit.net trzeba być pewnym, że wszystkie metody są kompletnie odizolowane. Jednakże, jezeli chcemy udostępnić jakieś dane miedzy naszymi metodami testowymi to xUnit nam na to zezwoli.

JUnit – W JUnit, test jednostkowy, który pozwala sprawdzic czy dana funkcjonalność działa wzamierzony sposób. Używanie JUnitów znacznie przyśpiesza pracę nad projektem. JUnity powinno się dzielić na wybrane testy metody lub funkcjonalności, idąc od najprostszych do bardziej zaawansowanych. JUnit jest przykładem architektury xUnity dla frameworków testów jednostkowych.

Junit vs xUnit.net

Porównanie:

	Junit	xUnit.net
Język programowania	Java	.NET
Strona klienta	JUnit umożliwia testowanie front-endowych komponentów takich jak indywidualne klasy oraz funkcje, które wchodzą w skład front-endu.	Możliwość niezależnego testowania równych komponetnów frontendowych, ponieważ jest to framework do testów jednostowych.
Strona serwera	Możliwość testów klas oraz funkcji, które składają się na back-end takie jak np. połączenia bazy danych.	Możliwość niezależnych testowów różnych komponentów back-endu, ponieważ jest to framework do testów jednostkowych
Stany danych (Fixtures)	JUnit zawiera metodę setUp(), która jest uruchamiana przed każdym wywołaniem testu, oraz metodę tearDown(), która jest uruchamiana po każdej metodzie testowej.	Posiada klasy, które są konfirurowane po przetestowaniu klasy.
Grupowanie stanów danych (Group fixtures)	Możliwość używania wbudowanych funkcji setUp() i tearDown() jako grupowania określonych stanów danych.	xUnit.net zawiera narzędzia do zbierania danych, które pozwalają na współdzielenie kontekstu między wieloma testami.

AssertJ powstał jako opensource'owa biblioteka, dzięki której możemy pisać w uproszczony sposób. AssertJ znaczenie upraszcza weryfikację wyjątków nawet w porównaniu z JUnit 5. AssertJ zdecydowanie wygrywa czytelniejszym API, które ułatwia zapoznawanie się z napisanymi testami. Obie te biblioteki czyli AssertJ i JUnit 5 możemy łaczyć, aby użyskać ciekawy rezultat. Metoda statyczna assertAll pochodzi z JUnit, natomiast dobrze znany assertThat z AssertJ. Jak widać można wyciągnąć z każdej biblioteki co najlepsze i połączyć to w czytelne testy jednostkowe.

Java

Mockito to biblioteka udostępniająca API do tworzenia mokowalnych obiektów w Javie. Obiekt mokowany to atrapa implementacji obiektu.
Po utworzeniu takiego obiektu można zdefiniować atrybuty obiektu bez powoływania jego instancji wraz z uzupełnianiem wszystkich właściwości.

EasyMock to framework podobny do Mockito. Pozwala na utworzenia własnych mockowych obiektów, w których można zasymulować dany obiekt,
ustalić sposób zachowania oraz sprawdzić czy funkcjonalności działają jak przewidujemy.

Porównanie

Mockito	EasyMock
działa na licencji Mockito	działa na licencji Apache
pozwala na tworzenie mocków oraz szpiegów(szpiedzy działają na realnych obiektach, jeśli nie zrobi się mockowej wersji metody wywoływana jest prawdziwa)	pozwala na stworzenie mocków
wywoływanie mocków Mockito.when(mock.method(args)).thenReturn(value)	wywoływanie mocków EasyMock.expect(mock.method(args)).andReturn(Value)
weryfikowanie mocków Mockito.verify(mock).method(args)	weryfikowanie mocków EasyMock.verify(mock)
przechwytywanie wyjątków .thenThrow(ExceptionClass.class)	przechwytywanie wyjątków .andThrow(new ExceptionClass())
możliwość konfiguracji za pomocą adnotacji	możliwość konfiguracji za pomocją adnotacji
brak potrzeby wywoływania powtórki tworzenia mocku	konieczność wywoływania @replay do ponownego wykorzystania mocku
możliwość weryfikacji nieoczekiwanych inwokacji, niepotrzebnych inwokacji oraz weryfikacji kolejności	możliwość weryfikacji nieoczekiwanych inwokacji, niepotrzebnych inwokacji oraz weryfikacji kolejności
lepsza czytelność kodu verify(), when()	gorsza czytelność kodu expect(mock.foo()), mock.foo()
weryfikacja jest wyraźna, wskazanie miejsca błędu	weryfikacja nie jest wyraźna, brak miejsca błędu
weryfikacja w kolejności jest elastyczna Nie wymaga weryfikacji każdej pojedynczej interakcji	weryfikacja w kolejności nie jest elastyczna

JustMock jest open source'owym rozwiązaniem pozwalającym na łatwe wykonywanie testów jednostkowych, najlepiej dla projektów w metodologii SOLID.
Pozwala na stworzenie obiektów mokowanych, w których uzupłniamy podstawowe istotne wartości do celów testowych.

TypeMock to płatne rozwiązanie, pozwalające na tworzenie obiektów mokowanych, działające wspólnie z frameworkiem .NET.
Pozwala na podłączenie udających zależności w jednym podejścią oraz ostrzega o wewnętrznych zależnościach w testach.

Porównanie

JustMock	TypeMock
działa z frameworkiem .NET	działa z frameworkiem .NET
nie pozwala na mockowanie obiektów na produkcji	pozwala na mockowanie obiektów, do których nie można dotrzeć z poziomu testu np. zainicjowane na produkcji

3. Przykładowe testy

Fragment klasy VehicleList:

```
13      public void initList(){
14          Car car = new Car();
15          Car car1 = new Car();
16          Car car2 = new Car();
17          Car car3 = new Car();
18          Car car4 = new Car();
19          Car car5 = new Car();
20          Car car6 = new Car();
21          Car car7 = new Car();
22          Car car8 = new Car();
23          Car car9 = new Car();
24
25
26          car.setCar("YV1MK76F2A2182012", "WR065CJ", "BMW", "sedan", "ADAYUB1997");
27          car.setVehicleOverview(true);
28          vehicleList.add(car);
29
30          car.setCar("W0L0AHL4878062512", "TI069AJ", "OPEL", "kombi", "AGASAB5423");
31          car.setVehicleOverview(true);
32          vehicleList.add(car1);
33
34          car.setCar("WAUZZZ4L47D017597", "WRAWNSRA", "AUDI", "sedan", "TGGSTF3123");
35          car.setVehicleOverview(false);
36          vehicleList.add(car2);
```

```
67
68      public List<Vehicle> getVehicleList() { return vehicleList; }
71
72      public Vehicle getCar(String identificationNumber){
73          for (int i = 0; i < vehicleList.size(); i++){
74              if(vehicleList.get(i).getIdentificationNumber().equals(identificationNumber)){
75                  return vehicleList.get(i);
76              }
77          }
78          return null;
79      }
80
81      public List<Vehicle> getListPositiveOverview(){
82          List<Vehicle> overViewPositive = new ArrayList<>();
83          for (int i = 0; i < vehicleList.size(); i++){
84              if(vehicleList.get(i).getVehicleOverView()==true){
85                  overViewPositive.add(vehicleList.get(i));
86              }
87          }
88          return overViewPositive;
89      }
```

Przykładowe testy VehicleList w Javie:

AssertJ EasyMock:

```
13  public class VehicleListAssertJEasyMockTest {
14
15      @Test
16      public void getVehicleList(){
17
18          //given
19          VehicleList vehicleList = createNiceMock(VehicleList.class);
20
21          //when
22          expect(vehicleList.getVehicleList()).andStubReturn(setVehicle());
23          replay(vehicleList);
24
25          //then
26          assertThat(vehicleList.getVehicleList().size()).isEqualTo(10);
27
28      }
29
30      private List<Vehicle> setVehicle(){
31
32          VehicleList vehicleList = new VehicleList();
33          vehicleList.initList();
34
35          return vehicleList.getVehicleList();
36
37      }
```

AssertJ Mockito:

```java
public class VehicleListAssertJMockitoTest {

    @Test
    public void getVehicleList(){

        //given
        VehicleList vehicleList = mock(VehicleList.class);

        //when
        when(vehicleList.getVehicleList()).thenReturn(setVehicle());

        //then
        assertThat(vehicleList.getVehicleList().size()).isEqualTo(10);

    }

    private List<Vehicle> setVehicle(){

        VehicleList vehicleList = new VehicleList();
        vehicleList.initList();

        return vehicleList.getVehicleList();
```

EasyMock:

```java
public class VehicleListEasyMockTest {

    @Test
    public void getVehicleList(){

        //given
        VehicleList vehicleList = createNiceMock(VehicleList.class);

        //when
        expect(vehicleList.getVehicleList()).andStubReturn(setVehicle());
        replay(vehicleList);

        //then
        Assertions.assertEquals(vehicleList.getVehicleList().size(), 10);

    }
```

EasyMock Mockito:

```
13  public class VehicleListTest {
14
15      @Test
16      public void getVehicleList(){
17
18          //given
19          VehicleList vehicleList = mock(VehicleList.class);
20
21          //when
22          when(vehicleList.getVehicleList()).thenReturn(setVehicle());
23
24          //then
25          Assertions.assertEquals(vehicleList.getVehicleList().size(), 10);
26
27      }
28
29      private List<Vehicle> setVehicle(){
30
31          VehicleList vehicleList = new VehicleList();
32          vehicleList.initList();
33
34          return vehicleList.getVehicleList();
35
36      }
```

4. Wydajność testów

a. Testy w c#

Lp.	Test name	Test 1	Test 2	Test 3	Test 4	Test 5	Min	Max	**Avg**
1	Mock_compare	470	268	268	256	254	254	470	303.2
2	mock_compare.Tests.NUnit.Builder	435	230	241	228	227	228	435	272.2
3	CarDealerListNUnitJustMockTest	235	135	149	128	138	128	235	157
4	CarDealerListNUnitMoqTest	175	72	71	76	69	69	175	92.6
5	CarListNUnitJustMockTest	4	3	3	3	3	3	4	3.2
6	CarListNUnitMoqTest	2	2	2	2	2	2	2	2
7	ReportListNUnitJustMockTest	3	3	3	4	3	3	4	3.2
8	ReportListNUnitTypeMockTest	2	2	1	2	1	1	2	1.6
9	SalesmanListNUnitJustMockTest	3	3	3	3	3	3	3	3
10	SalesmanListNUnitTypeMockTest	2	2	1	2	1	1	2	1.6
11	VehicleListNUnitJustMockTest	7	6	6	6	5	5	7	6
12	VehicleListNUnitMoqTest	2	2	2	2	2	2	2	2
13	CsvImportNUnitTest	9	2	2	2	2	2	9	3.4
14	mock_compare.Tests.XUnit.Builder	22	32	21	22	21	21	32	23.6
15	CarDealerListXUnitJustMockTest	4	8	4	4	4	4	8	4.8
16	CarDealerListXUnitMoqTest	1	1	1	1	1	1	1	1
17	CarListXUnitJustMockTest	1	4	2	2	2	1	4	2.2
18	CarListXUnitMoqTest	1	1	1	1	1	1	1	1
19	ReportListXUnitJustMockTest	2	3	2	1	2	1	3	2
20	ReportListXUnitMoqTest	1	1	1	3	1	1	3	1.4
21	SalesmanListXUnitJustMockTest	1	1	1	4	4	1	4	2.2
22	SalesmanListXUnitMoqTest	4	4	4	4	4	4	4	4
23	VehicleListXUnitJustMock	6	8	4	4	4	4	8	5.2

24	VehicleListXUnitMoqTest	4	4	4	4	4	4	4	4
25	mock_compare.Tests.XUnit.Files	4	4	4	4	4	4	4	4
26	CsvImportXUnitTest	4	4	4	4	4	4	4	4
27	getVehiceList	3	3	3	3	2	2	3	2.8
28	quantityOfNegativeOverviews	3	2	2	2	1	1	3	2
29	quantityOfPositiveOverviews	1	1	1	1	1	1	1	1
30	convertCsvToArraySuccess	9	2	2	2	2	2	9	3.4

Czas jest podawany w ms.

NUnitJustMock – 0 vs NUnitMoq – 5

JustMock w każdym z testów był wolniejszy od Moq.

XUnitJustMock – 1 vs XUnitMoq – 4

JustMock tylko w jednym przypadku był szybszy od Moq.

b. Testy w Java

Lp.	Test name	Test 1	Test 2	Test 3	Test 4	Test 5	Min	Max	**Avg**
1	builder	884	1292	853	823	993	823	1292	969
2	CarDelaerListAssertJEasyMockTest	243	269	264	212	235	212	269	244.6
3	CarDelaerListAssertJMockitoTest	495	866	458	470	620	470	866	581.8
4	CarDelaerListEasyMockTest	2	1	2	3	2	1	3	2
5	CarDealerListTest	1	1	1	1	1	1	1	1
6	CarListAssertJEasyMockTest	6	6	7	6	5	5	7	6
7	CarListAssertJMockitoTest	38	36	36	35	44	35	44	37.8
8	CarListListEasyMockTest	1	1	1	1	1	1	1	1
9	CarListTest	1	1	1	2	1	1	2	1.2
10	ReportListAssertJEasyMockTest	4	3	4	4	3	3	4	3.6
11	ReportListAssertJMockitoTest	20	19	14	12	15	12	20	16
12	ReportListEasyMockTest	1	1	1	1	1	1	1	1
13	ReportListTest	1	2	1	1	1	1	2	1.2

14	SalesmanListAssertJEasyMockTest	3	5	2	3	3	2	5	3.2
15	SalesmanListAssertJMockitoTest	16	21	11	12	11	11	21	14.2
16	SalesmanListEasyMockTest	1	1	1	1	1	1	1	1
17	SalesmanListTest	1	1	1	1	2	1	2	1.2
18	VehicleListAssertJEasyMockTest	6	7	6	9	5	6	9	6.6
19	VehicleListAssertJMockitoTest	39	48	43	45	39	39	48	42.8
20	VehicleListEasyMockTest	2	1	1	2	2	1	2	1.6
21	VehicleListTest	3	3	2	3	2	2	3	2.6

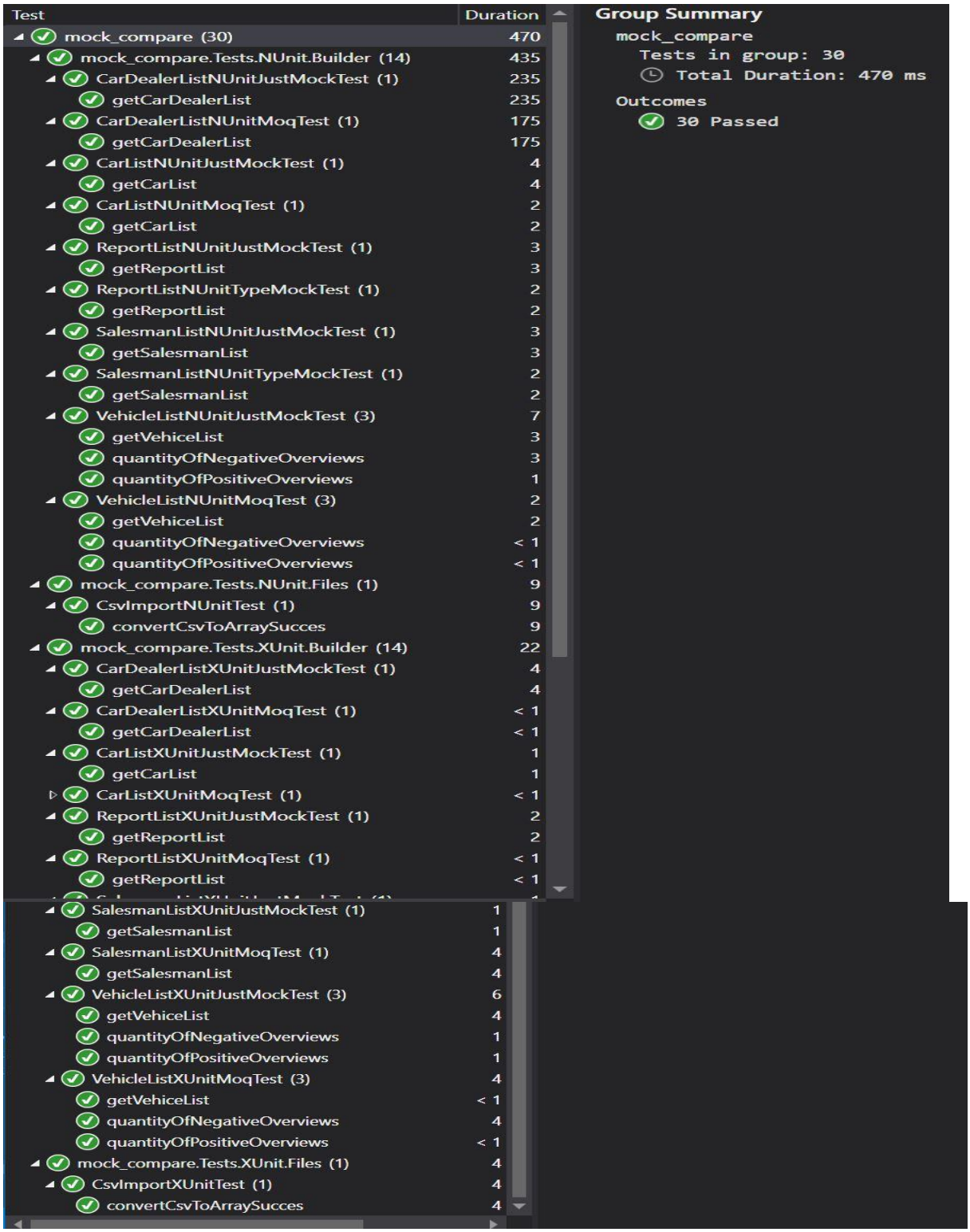
AssertJEasyMock – 0 vs AssertJMockito – 0 vs EasyMock – 4 vs Mockito - 1

Według tabeli testów najszybszym sposobem na testy jest EasyMock, a zaraz po nim Mockito. Używanie AssertJ znacznie przedłuża wykonywanie testów, lecz AssertJEasyMock w większości przypadków nie ustępuje znacznie zwycięzcom EasyMock i Mockito.

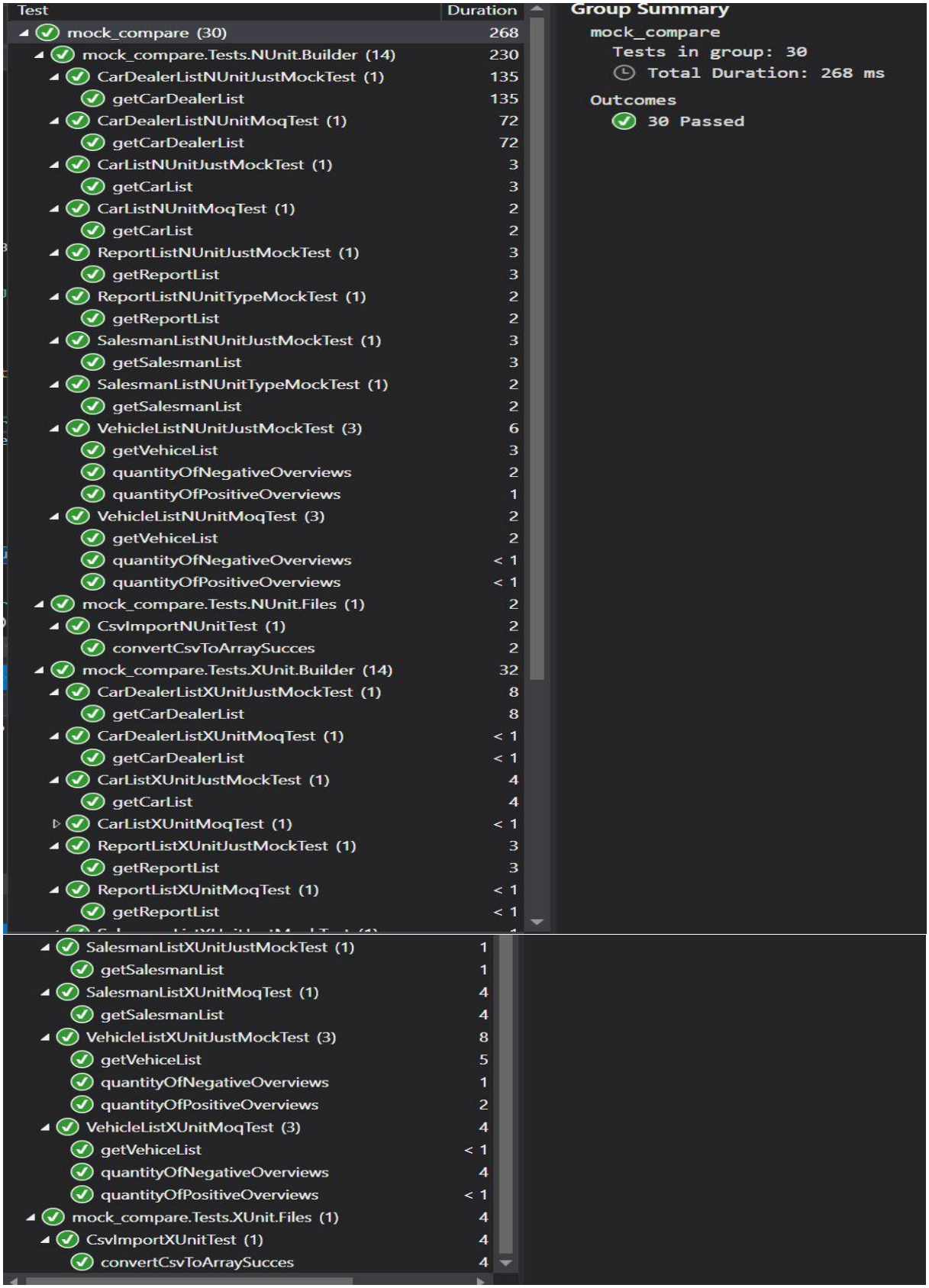
5. Screeny testów

C#

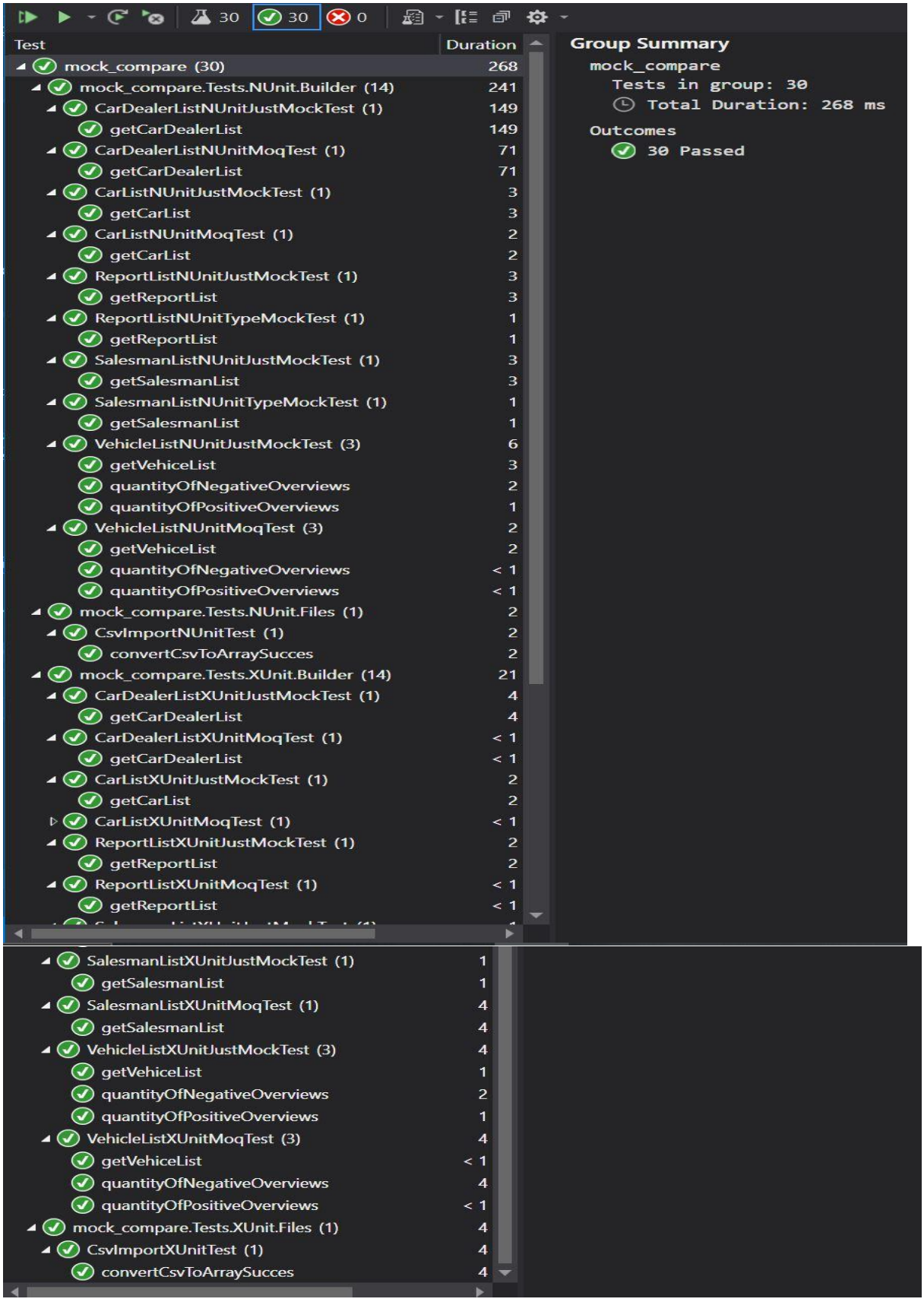
Test1:



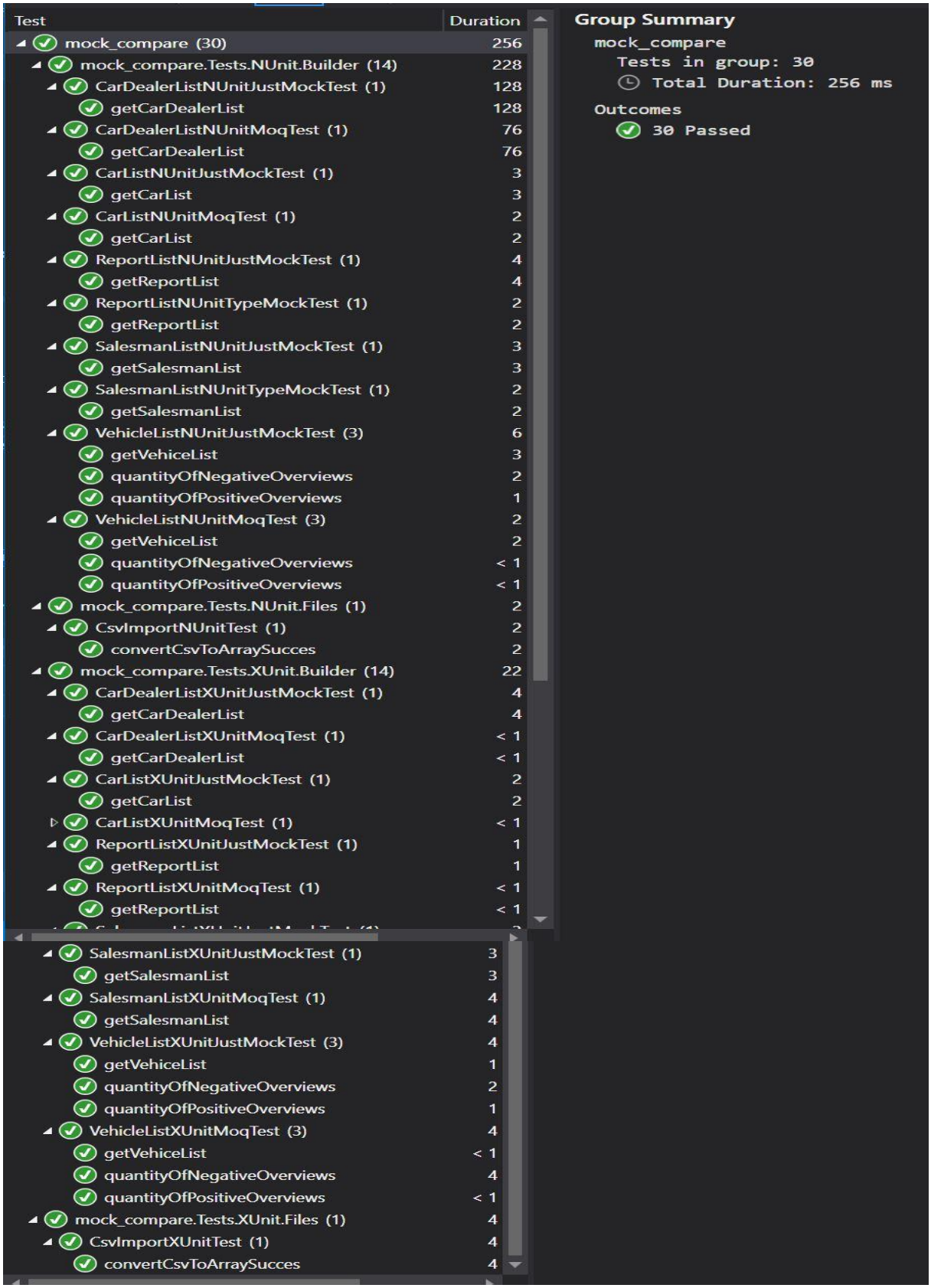
Test2:


Test
Duration
mock_compare (30) 268
mock_compare.Tests.NUnit.Builder (14) 230
CarDealerListNUnitJustMockTest (1) 135
getCarDealerList 135
CarDealerListNUnitMoqTest (1) 72
getCarDealerList 72
CarListNUnitJustMockTest (1) 3
getCarList 3
CarListNUnitMoqTest (1) 2
getCarList 2
ReportListNUnitJustMockTest (1) 3
getReportList 3
ReportListNUnitTypeMockTest (1) 2
getReportList 2
SalesmanListNUnitJustMockTest (1) 3
getSalesmanList 3
SalesmanListNUnitTypeMockTest (1) 2
getSalesmanList 2
VehicleListNUnitJustMockTest (3) 6
getVehiceList 3
quantityOfNegativeOverviews 2
quantityOfPositiveOverviews 1
VehicleListNUnitMoqTest (3) 2
getVehiceList 2
quantityOfNegativeOverviews < 1
quantityOfPositiveOverviews < 1
mock_compare.Tests.NUnit.Files (1) 2
CsvImportNUnitTest (1) 2
convertCsvToArraySucces 2
mock_compare.Tests.XUnit.Builder (14) 32
CarDealerListXUnitJustMockTest (1) 8
getCarDealerList 8
CarDealerListXUnitMoqTest (1) < 1
getCarDealerList < 1
CarListXUnitJustMockTest (1) 4
getCarList 4
CarListXUnitMoqTest (1) < 1
ReportListXUnitJustMockTest (1) 3
getReportList 3
ReportListXUnitMoqTest (1) < 1
getReportList < 1
Group Summary
mock_compare
Tests in group: 30
Total Duration: 268 ms
Outcomes
30 Passed
SalesmanListXUnitJustMockTest (1) 1
getSalesmanList 1
SalesmanListXUnitMoqTest (1) 4
getSalesmanList 4
VehicleListXUnitJustMockTest (3) 8
getVehiceList 5
quantityOfNegativeOverviews 1
quantityOfPositiveOverviews 2
VehicleListXUnitMoqTest (3) 4
getVehiceList < 1
quantityOfNegativeOverviews 4
quantityOfPositiveOverviews < 1
mock_compare.Tests.XUnit.Files (1) 4
CsvImportXUnitTest (1) 4
convertCsvToArraySucces 4

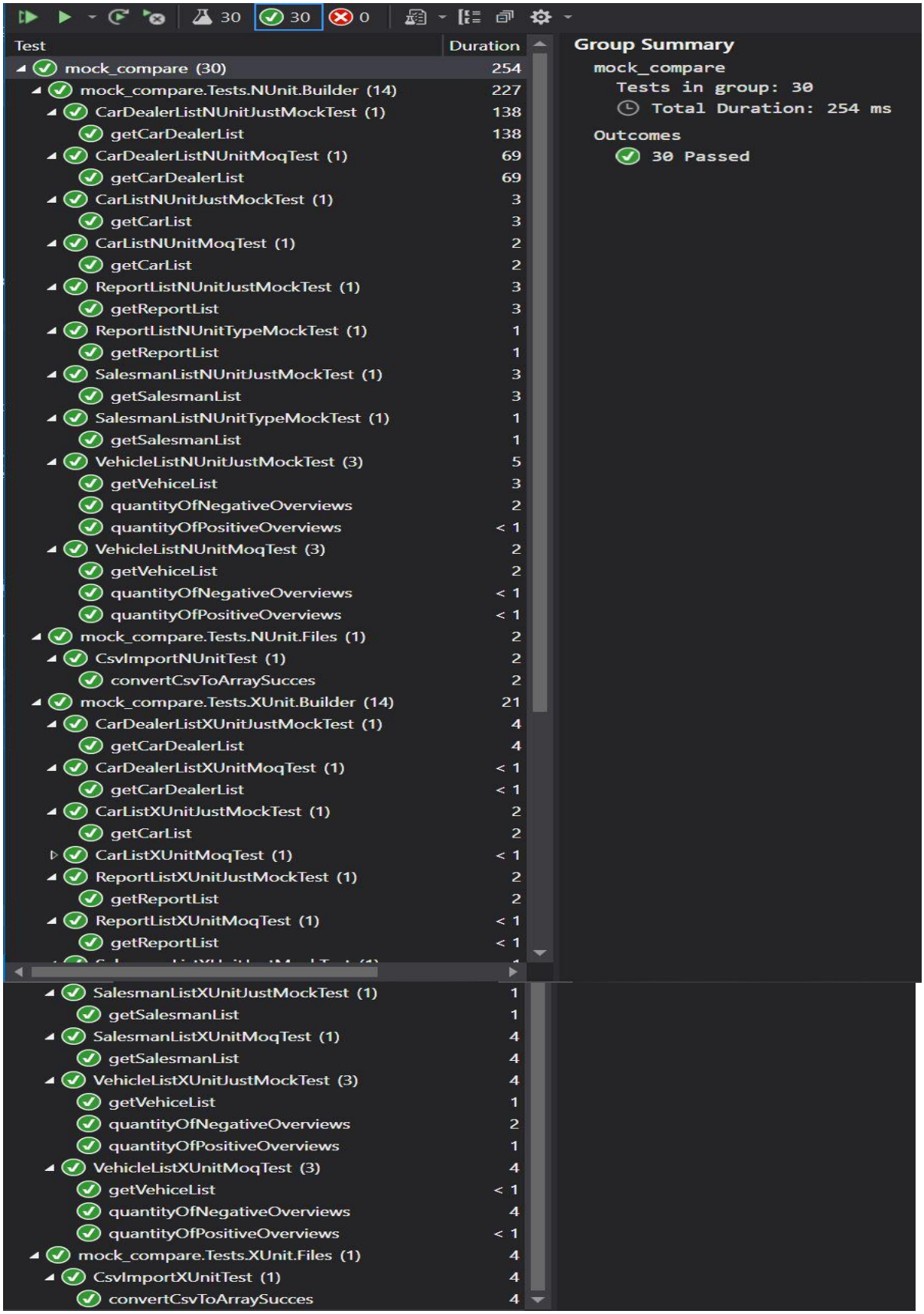
Test3:


30
30
0
Test
Duration
mock_compare (30) 268
mock_compare.Tests.NUnit.Builder (14) 241
CarDealerListNUnitJustMockTest (1) 149
getCarDealerList 149
CarDealerListNUnitMoqTest (1) 71
getCarDealerList 71
CarListNUnitJustMockTest (1) 3
getCarList 3
CarListNUnitMoqTest (1) 2
getCarList 2
ReportListNUnitJustMockTest (1) 3
getReportList 3
ReportListNUnitTypeMockTest (1) 1
getReportList 1
SalesmanListNUnitJustMockTest (1) 3
getSalesmanList 3
SalesmanListNUnitTypeMockTest (1) 1
getSalesmanList 1
VehicleListNUnitJustMockTest (3) 6
getVehiceList 3
quantityOfNegativeOverviews 2
quantityOfPositiveOverviews 1
VehicleListNUnitMoqTest (3) 2
getVehiceList 2
quantityOfNegativeOverviews < 1
quantityOfPositiveOverviews < 1
mock_compare.Tests.NUnit.Files (1) 2
CsvImportNUnitTest (1) 2
convertCsvToArraySucces 2
mock_compare.Tests.XUnit.Builder (14) 21
CarDealerListXUnitJustMockTest (1) 4
getCarDealerList 4
CarDealerListXUnitMoqTest (1) < 1
getCarDealerList < 1
CarListXUnitJustMockTest (1) 2
getCarList 2
CarListXUnitMoqTest (1) < 1
ReportListXUnitJustMockTest (1) 2
getReportList 2
ReportListXUnitMoqTest (1) < 1
getReportList < 1
Group Summary
mock_compare
Tests in group: 30
Total Duration: 268 ms
Outcomes
30 Passed
SalesmanListXUnitJustMockTest (1) 1
getSalesmanList 1
SalesmanListXUnitMoqTest (1) 4
getSalesmanList 4
VehicleListXUnitJustMockTest (3) 4
getVehiceList 1
quantityOfNegativeOverviews 2
quantityOfPositiveOverviews 1
VehicleListXUnitMoqTest (3) 4
getVehiceList < 1
quantityOfNegativeOverviews 4
quantityOfPositiveOverviews < 1
mock_compare.Tests.XUnit.Files (1) 4
CsvImportXUnitTest (1) 4
convertCsvToArraySucces 4

Test4:

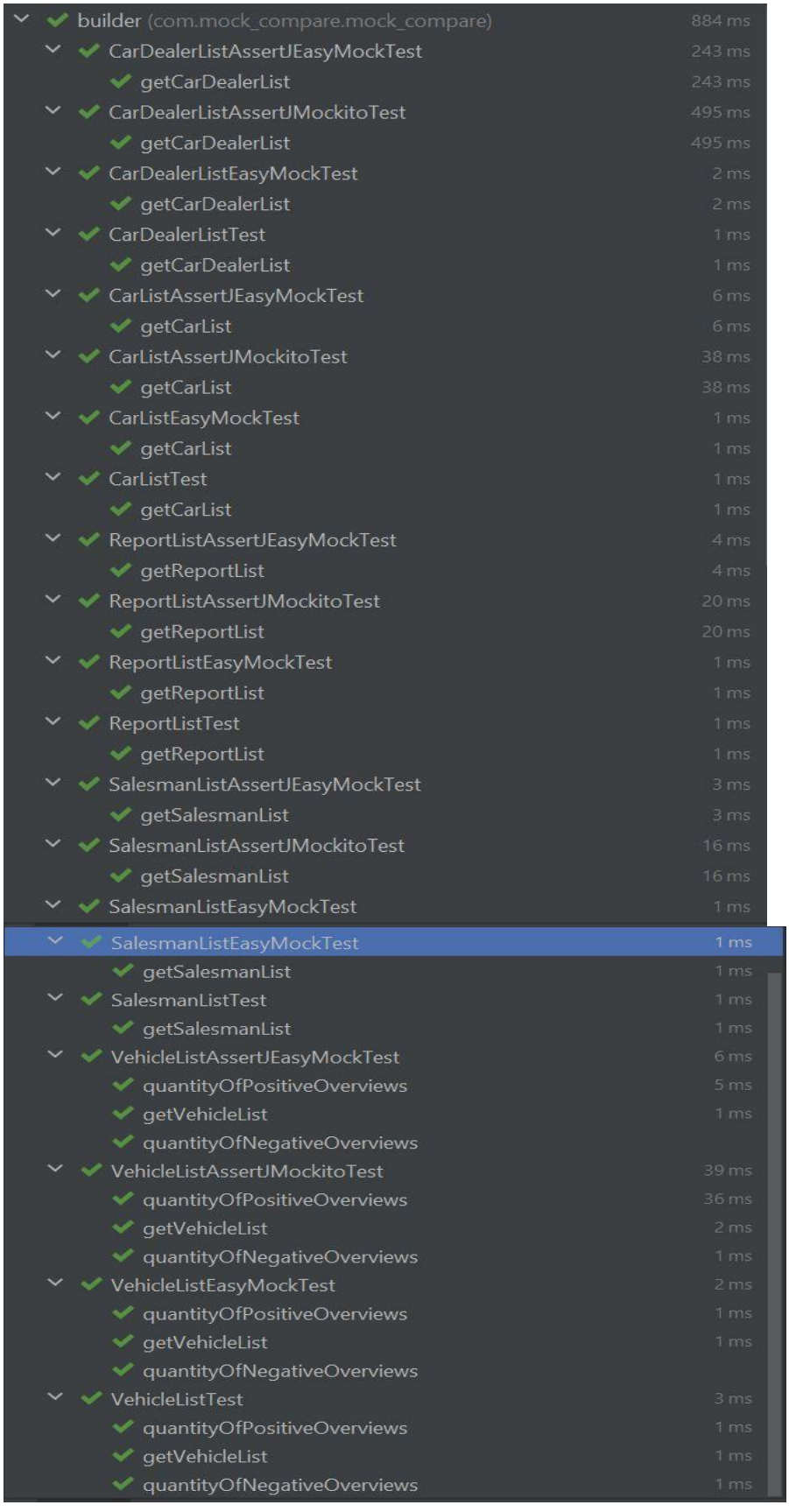

Test
Duration
mock_compare (30) 256
mock_compare.Tests.NUnit.Builder (14) 228
CarDealerListNUnitJustMockTest (1) 128
getCarDealerList 128
CarDealerListNUnitMoqTest (1) 76
getCarDealerList 76
CarListNUnitJustMockTest (1) 3
getCarList 3
CarListNUnitMoqTest (1) 2
getCarList 2
ReportListNUnitJustMockTest (1) 4
getReportList 4
ReportListNUnitTypeMockTest (1) 2
getReportList 2
SalesmanListNUnitJustMockTest (1) 3
getSalesmanList 3
SalesmanListNUnitTypeMockTest (1) 2
getSalesmanList 2
VehicleListNUnitJustMockTest (3) 6
getVehiceList 3
quantityOfNegativeOverviews 2
quantityOfPositiveOverviews 1
VehicleListNUnitMoqTest (3) 2
getVehiceList 2
quantityOfNegativeOverviews < 1
quantityOfPositiveOverviews < 1
mock_compare.Tests.NUnit.Files (1) 2
CsvImportNUnitTest (1) 2
convertCsvToArraySucces 2
mock_compare.Tests.XUnit.Builder (14) 22
CarDealerListXUnitJustMockTest (1) 4
getCarDealerList 4
CarDealerListXUnitMoqTest (1) < 1
getCarDealerList < 1
CarListXUnitJustMockTest (1) 2
getCarList 2
CarListXUnitMoqTest (1) < 1
ReportListXUnitJustMockTest (1) 1
getReportList 1
ReportListXUnitMoqTest (1) < 1
getReportList < 1
SalesmanListXUnitJustMockTest (1) 3
getSalesmanList 3
SalesmanListXUnitMoqTest (1) 4
getSalesmanList 4
VehicleListXUnitJustMockTest (3) 4
getVehiceList 1
quantityOfNegativeOverviews 2
quantityOfPositiveOverviews 1
VehicleListXUnitMoqTest (3) 4
getVehiceList < 1
quantityOfNegativeOverviews 4
quantityOfPositiveOverviews < 1
mock_compare.Tests.XUnit.Files (1) 4
CsvImportXUnitTest (1) 4
convertCsvToArraySucces 4
Group Summary
mock_compare
Tests in group: 30
Total Duration: 256 ms
Outcomes
30 Passed

Test5:

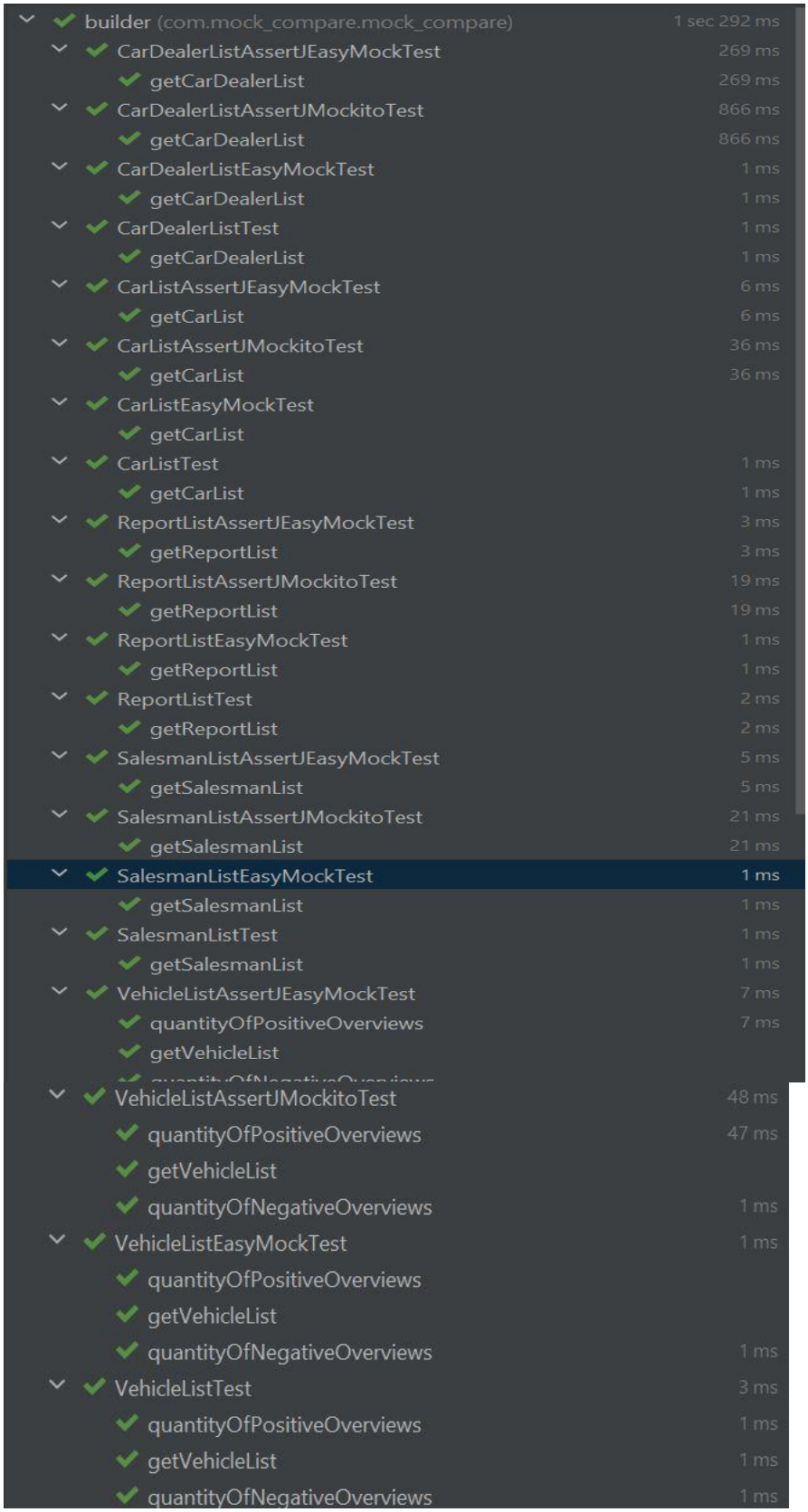

30
30
0
Test
Duration
mock_compare (30) 254
mock_compare.Tests.NUnit.Builder (14) 227
CarDealerListNUnitJustMockTest (1) 138
getCarDealerList 138
CarDealerListNUnitMoqTest (1) 69
getCarDealerList 69
CarListNUnitJustMockTest (1) 3
getCarList 3
CarListNUnitMoqTest (1) 2
getCarList 2
ReportListNUnitJustMockTest (1) 3
getReportList 3
ReportListNUnitTypeMockTest (1) 1
getReportList 1
SalesmanListNUnitJustMockTest (1) 3
getSalesmanList 3
SalesmanListNUnitTypeMockTest (1) 1
getSalesmanList 1
VehicleListNUnitJustMockTest (3) 5
getVehiceList 3
quantityOfNegativeOverviews 2
quantityOfPositiveOverviews < 1
VehicleListNUnitMoqTest (3) 2
getVehiceList 2
quantityOfNegativeOverviews < 1
quantityOfPositiveOverviews < 1
mock_compare.Tests.NUnit.Files (1) 2
CsvImportNUnitTest (1) 2
convertCsvToArraySucces 2
mock_compare.Tests.XUnit.Builder (14) 21
CarDealerListXUnitJustMockTest (1) 4
getCarDealerList 4
CarDealerListXUnitMoqTest (1) < 1
getCarDealerList < 1
CarListXUnitJustMockTest (1) 2
getCarList 2
CarListXUnitMoqTest (1) < 1
ReportListXUnitJustMockTest (1) 2
getReportList 2
ReportListXUnitMoqTest (1) < 1
getReportList < 1
SalesmanListXUnitJustMockTest (1) 1
getSalesmanList 1
SalesmanListXUnitMoqTest (1) 4
getSalesmanList 4
VehicleListXUnitJustMockTest (3) 4
getVehiceList 1
quantityOfNegativeOverviews 2
quantityOfPositiveOverviews 1
VehicleListXUnitMoqTest (3) 4
getVehiceList < 1
quantityOfNegativeOverviews 4
quantityOfPositiveOverviews < 1
mock_compare.Tests.XUnit.Files (1) 4
CsvImportXUnitTest (1) 4
convertCsvToArraySucces 4
Group Summary
mock_compare
Tests in group: 30
Total Duration: 254 ms
Outcomes
30 Passed

Java

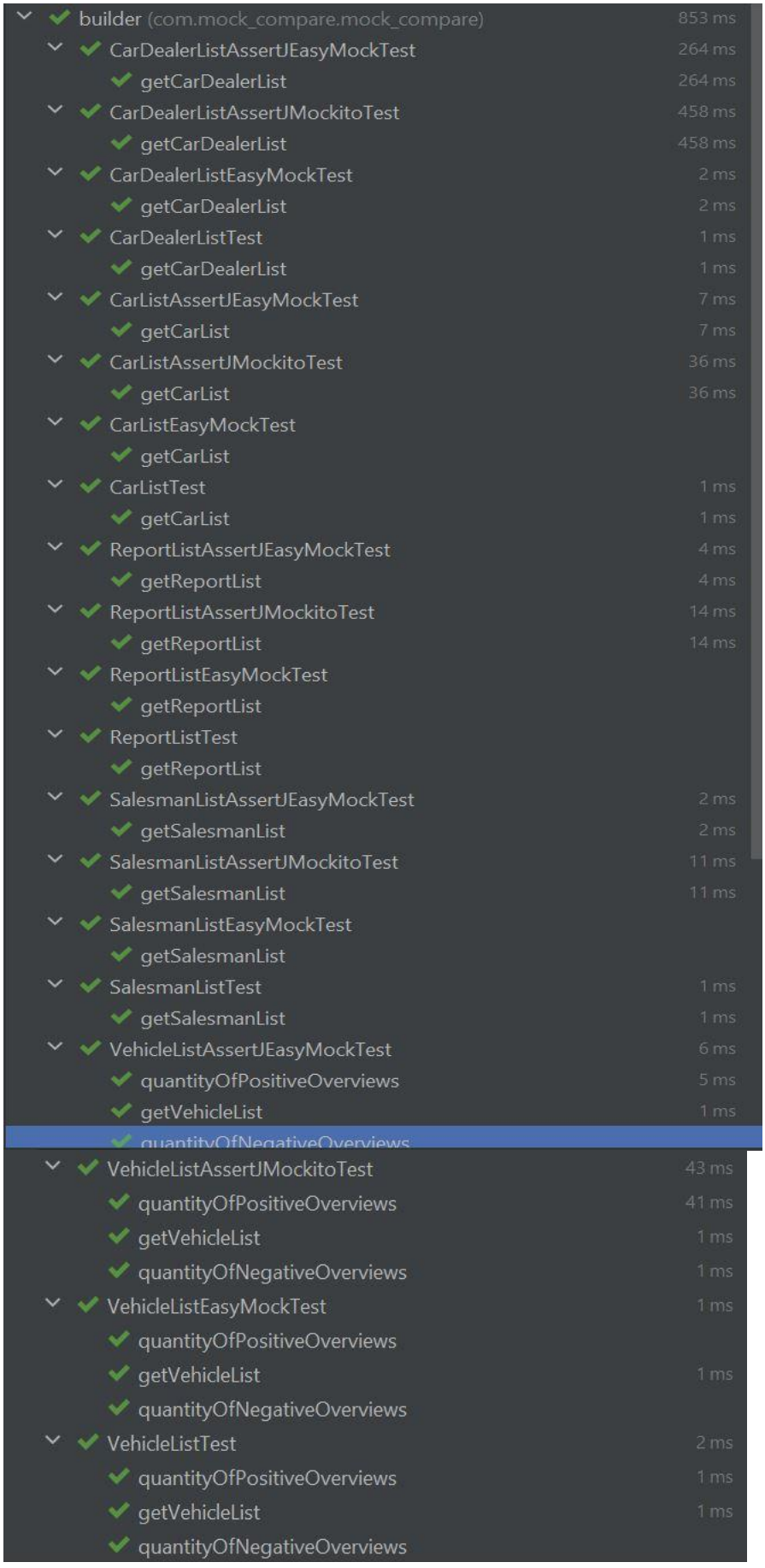
Test1:


builder (com.mock_compare.mock_compare) 884 ms
CarDealerListAssertJEasyMockTest 243 ms
getCarDealerList 243 ms
CarDealerListAssertJMockitoTest 495 ms
getCarDealerList 495 ms
CarDealerListEasyMockTest 2 ms
getCarDealerList 2 ms
CarDealerListTest 1 ms
getCarDealerList 1 ms
CarListAssertJEasyMockTest 6 ms
getCarList 6 ms
CarListAssertJMockitoTest 38 ms
getCarList 38 ms
CarListEasyMockTest 1 ms
getCarList 1 ms
CarListTest 1 ms
getCarList 1 ms
ReportListAssertJEasyMockTest 4 ms
getReportList 4 ms
ReportListAssertJMockitoTest 20 ms
getReportList 20 ms
ReportListEasyMockTest 1 ms
getReportList 1 ms
ReportListTest 1 ms
getReportList 1 ms
SalesmanListAssertJEasyMockTest 3 ms
getSalesmanList 3 ms
SalesmanListAssertJMockitoTest 16 ms
getSalesmanList 16 ms
SalesmanListEasyMockTest 1 ms
SalesmanListEasyMockTest 1 ms
getSalesmanList 1 ms
SalesmanListTest 1 ms
getSalesmanList 1 ms
VehicleListAssertJEasyMockTest 6 ms
quantityOfPositiveOverviews 5 ms
getVehicleList 1 ms
quantityOfNegativeOverviews
VehicleListAssertJMockitoTest 39 ms
quantityOfPositiveOverviews 36 ms
getVehicleList 2 ms
quantityOfNegativeOverviews 1 ms
VehicleListEasyMockTest 2 ms
quantityOfPositiveOverviews 1 ms
getVehicleList 1 ms
quantityOfNegativeOverviews
VehicleListTest 3 ms
quantityOfPositiveOverviews 1 ms
getVehicleList 1 ms
quantityOfNegativeOverviews 1 ms

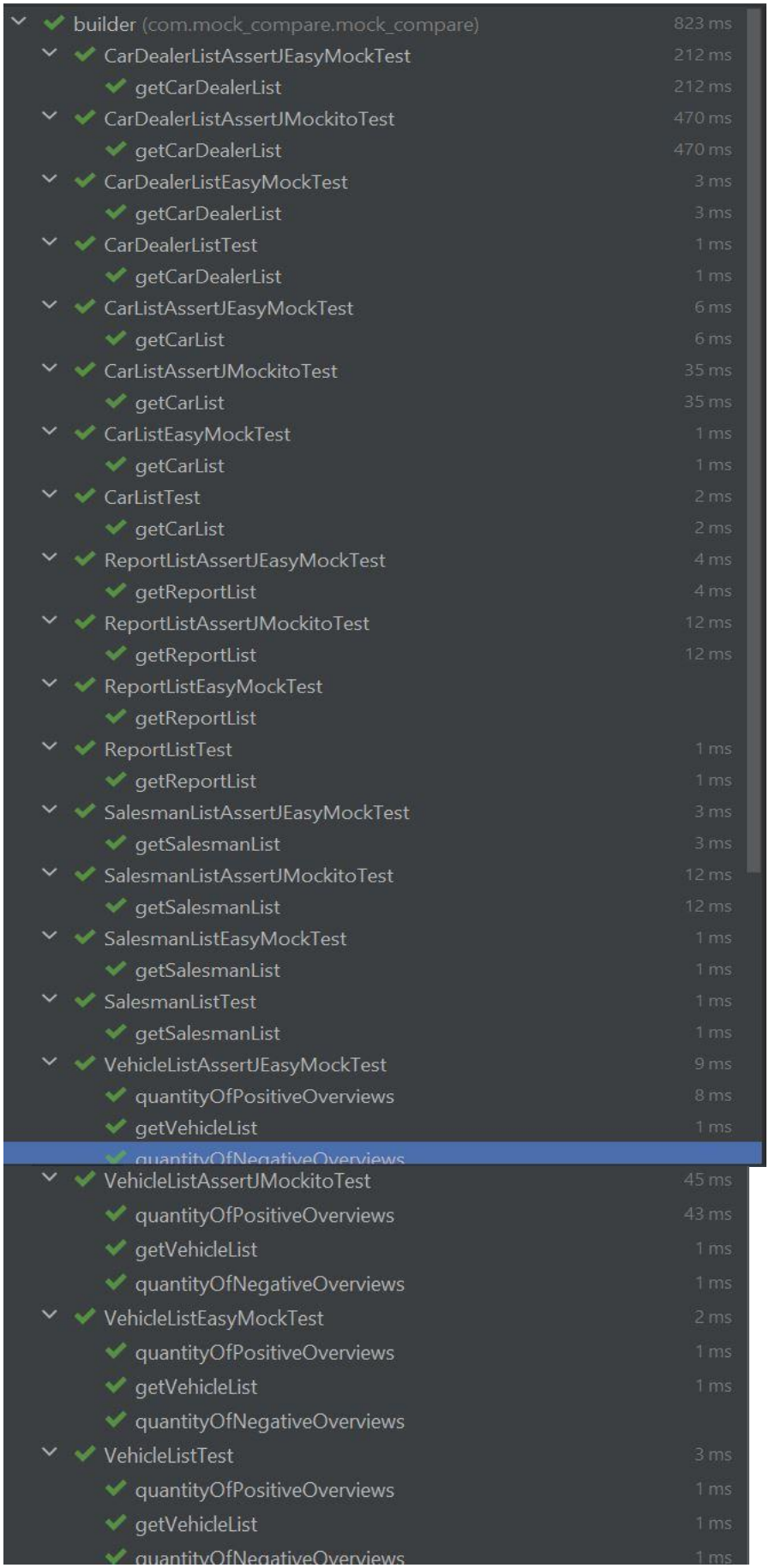
Test2:


builder (com.mock_compare.mock_compare) 1 sec 292 ms
CarDealerListAssertJEasyMockTest 269 ms
getCarDealerList 269 ms
CarDealerListAssertJMockitoTest 866 ms
getCarDealerList 866 ms
CarDealerListEasyMockTest 1 ms
getCarDealerList 1 ms
CarDealerListTest 1 ms
getCarDealerList 1 ms
CarListAssertJEasyMockTest 6 ms
getCarList 6 ms
CarListAssertJMockitoTest 36 ms
getCarList 36 ms
CarListEasyMockTest
getCarList
CarListTest 1 ms
getCarList 1 ms
ReportListAssertJEasyMockTest 3 ms
getReportList 3 ms
ReportListAssertJMockitoTest 19 ms
getReportList 19 ms
ReportListEasyMockTest 1 ms
getReportList 1 ms
ReportListTest 2 ms
getReportList 2 ms
SalesmanListAssertJEasyMockTest 5 ms
getSalesmanList 5 ms
SalesmanListAssertJMockitoTest 21 ms
getSalesmanList 21 ms
SalesmanListEasyMockTest 1 ms
getSalesmanList 1 ms
SalesmanListTest 1 ms
getSalesmanList 1 ms
VehicleListAssertJEasyMockTest 7 ms
quantityOfPositiveOverviews 7 ms
getVehicleList
VehicleListAssertJMockitoTest 48 ms
quantityOfPositiveOverviews 47 ms
getVehicleList
quantityOfNegativeOverviews 1 ms
VehicleListEasyMockTest 1 ms
quantityOfPositiveOverviews
getVehicleList
quantityOfNegativeOverviews 1 ms
VehicleListTest 3 ms
quantityOfPositiveOverviews 1 ms
getVehicleList 1 ms
quantityOfNegativeOverviews 1 ms

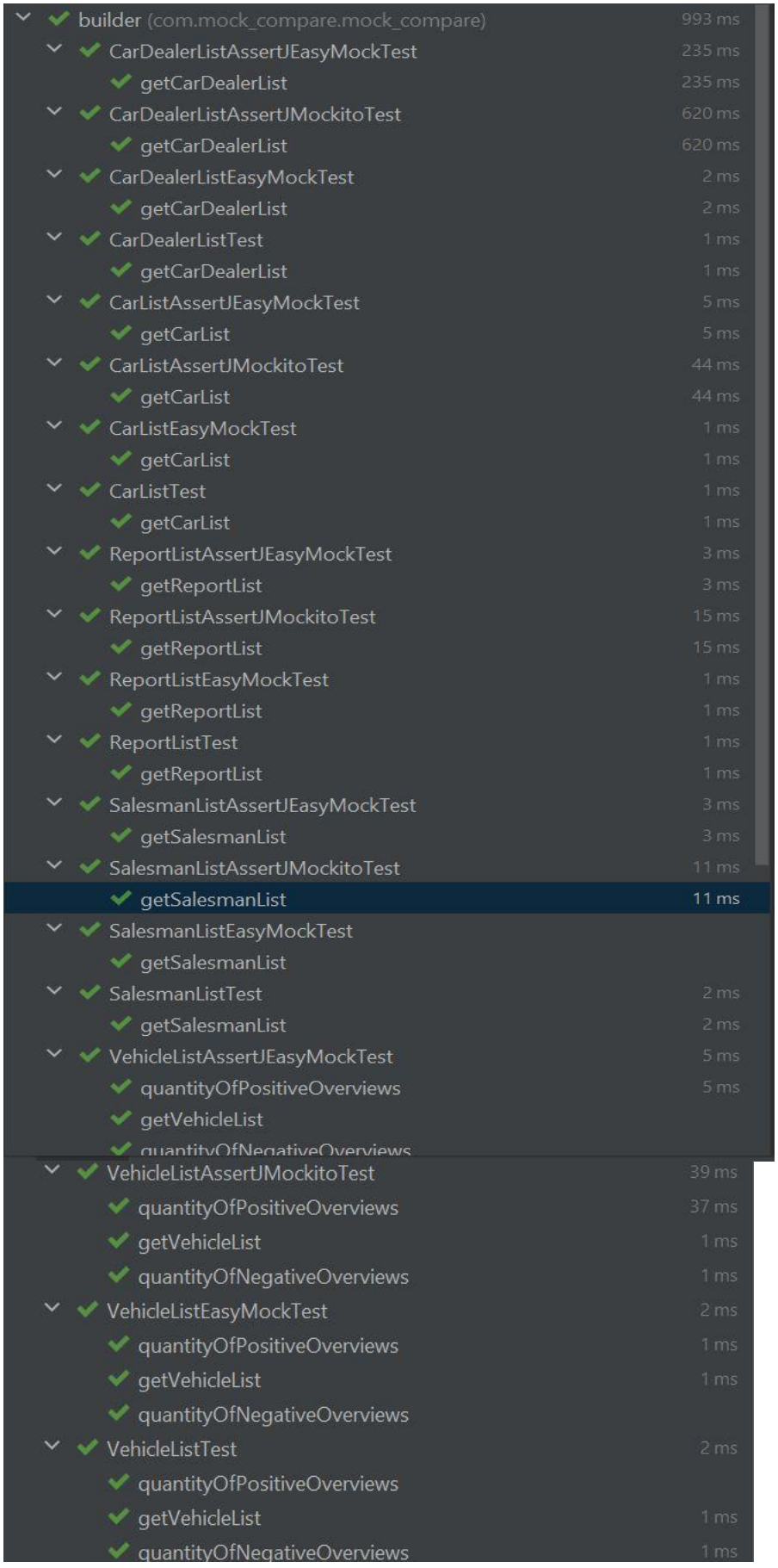
Test3


builder (com.mock_compare.mock_compare) 853 ms
CarDealerListAssertJEasyMockTest 264 ms
getCarDealerList 264 ms
CarDealerListAssertJMockitoTest 458 ms
getCarDealerList 458 ms
CarDealerListEasyMockTest 2 ms
getCarDealerList 2 ms
CarDealerListTest 1 ms
getCarDealerList 1 ms
CarListAssertJEasyMockTest 7 ms
getCarList 7 ms
CarListAssertJMockitoTest 36 ms
getCarList 36 ms
CarListEasyMockTest
getCarList
CarListTest 1 ms
getCarList 1 ms
ReportListAssertJEasyMockTest 4 ms
getReportList 4 ms
ReportListAssertJMockitoTest 14 ms
getReportList 14 ms
ReportListEasyMockTest
getReportList
ReportListTest
getReportList
SalesmanListAssertJEasyMockTest 2 ms
getSalesmanList 2 ms
SalesmanListAssertJMockitoTest 11 ms
getSalesmanList 11 ms
SalesmanListEasyMockTest
getSalesmanList
SalesmanListTest 1 ms
getSalesmanList 1 ms
VehicleListAssertJEasyMockTest 6 ms
quantityOfPositiveOverviews 5 ms
getVehicleList 1 ms
VehicleListAssertJMockitoTest 43 ms
quantityOfPositiveOverviews 41 ms
getVehicleList 1 ms
quantityOfNegativeOverviews 1 ms
VehicleListEasyMockTest 1 ms
quantityOfPositiveOverviews
getVehicleList 1 ms
quantityOfNegativeOverviews
VehicleListTest 2 ms
quantityOfPositiveOverviews 1 ms
getVehicleList 1 ms
quantityOfNegativeOverviews

Test4:


builder (com.mock_compare.mock_compare) 823 ms
CarDealerListAssertJEasyMockTest 212 ms
getCarDealerList 212 ms
CarDealerListAssertJMockitoTest 470 ms
getCarDealerList 470 ms
CarDealerListEasyMockTest 3 ms
getCarDealerList 3 ms
CarDealerListTest 1 ms
getCarDealerList 1 ms
CarListAssertJEasyMockTest 6 ms
getCarList 6 ms
CarListAssertJMockitoTest 35 ms
getCarList 35 ms
CarListEasyMockTest 1 ms
getCarList 1 ms
CarListTest 2 ms
getCarList 2 ms
ReportListAssertJEasyMockTest 4 ms
getReportList 4 ms
ReportListAssertJMockitoTest 12 ms
getReportList 12 ms
ReportListEasyMockTest
getReportList
ReportListTest 1 ms
getReportList 1 ms
SalesmanListAssertJEasyMockTest 3 ms
getSalesmanList 3 ms
SalesmanListAssertJMockitoTest 12 ms
getSalesmanList 12 ms
SalesmanListEasyMockTest 1 ms
getSalesmanList 1 ms
SalesmanListTest 1 ms
getSalesmanList 1 ms
VehicleListAssertJEasyMockTest 9 ms
quantityOfPositiveOverviews 8 ms
getVehicleList 1 ms
quantityOfNegativeOverviews
VehicleListAssertJMockitoTest 45 ms
quantityOfPositiveOverviews 43 ms
getVehicleList 1 ms
quantityOfNegativeOverviews 1 ms
VehicleListEasyMockTest 2 ms
quantityOfPositiveOverviews 1 ms
getVehicleList 1 ms
quantityOfNegativeOverviews
VehicleListTest 3 ms
quantityOfPositiveOverviews 1 ms
getVehicleList 1 ms
quantityOfNegativeOverviews 1 ms

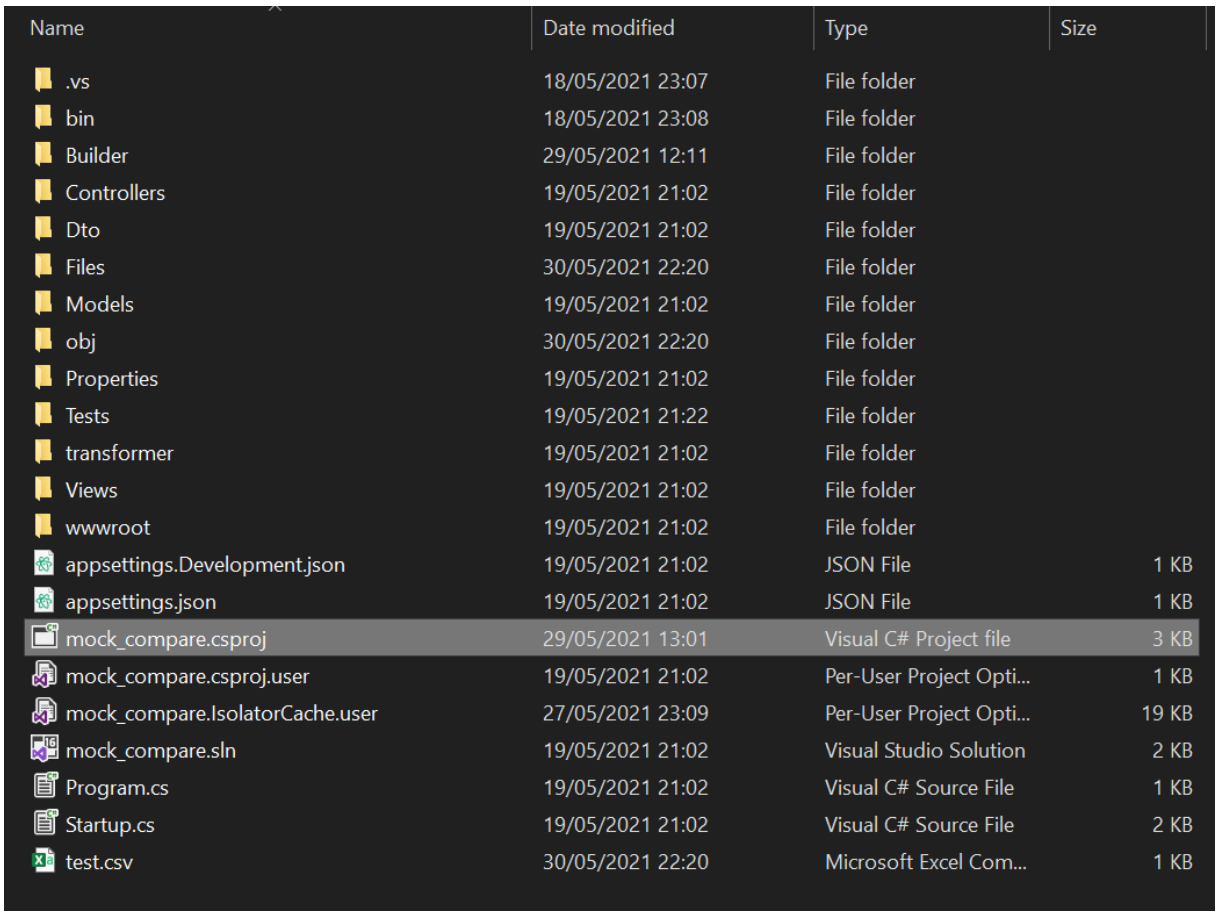
Test5:


builder (com.mock_compare.mock_compare) 993 ms
CarDealerListAssertJEasyMockTest 235 ms
getCarDealerList 235 ms
CarDealerListAssertJMockitoTest 620 ms
getCarDealerList 620 ms
CarDealerListEasyMockTest 2 ms
getCarDealerList 2 ms
CarDealerListTest 1 ms
getCarDealerList 1 ms
CarListAssertJEasyMockTest 5 ms
getCarList 5 ms
CarListAssertJMockitoTest 44 ms
getCarList 44 ms
CarListEasyMockTest 1 ms
getCarList 1 ms
CarListTest 1 ms
getCarList 1 ms
ReportListAssertJEasyMockTest 3 ms
getReportList 3 ms
ReportListAssertJMockitoTest 15 ms
getReportList 15 ms
ReportListEasyMockTest 1 ms
getReportList 1 ms
ReportListTest 1 ms
getReportList 1 ms
SalesmanListAssertJEasyMockTest 3 ms
getSalesmanList 3 ms
SalesmanListAssertJMockitoTest 11 ms
getSalesmanList 11 ms
SalesmanListEasyMockTest
getSalesmanList
SalesmanListTest 2 ms
getSalesmanList 2 ms
VehicleListAssertJEasyMockTest 5 ms
quantityOfPositiveOverviews 5 ms
getVehicleList
VehicleListAssertJMockitoTest 39 ms
quantityOfPositiveOverviews 37 ms
getVehicleList 1 ms
quantityOfNegativeOverviews 1 ms
VehicleListEasyMockTest 2 ms
quantityOfPositiveOverviews 1 ms
getVehicleList 1 ms
quantityOfNegativeOverviews
VehicleListTest 2 ms
quantityOfPositiveOverviews
getVehicleList 1 ms
quantityOfNegativeOverviews 1 ms

7. Problemy z wywołaniem testu w C#

Na początku próby testowania, testy nie startowały. W celu pozbycia się tego błędu należy dodać odpowiednią linijkę kodu do pliku o rozszerzeniu .csproj.

```
<PropertyGroup>
  <TargetFramework>netcoreapp3.1</TargetFramework>
  <IsPackable>false</IsPackable>
  <GenerateProgramFile>false</GenerateProgramFile>
</PropertyGroup>
```



Name	Date modified	Type	Size
.vs	18/05/2021 23:07	File folder	
bin	18/05/2021 23:08	File folder	
Builder	29/05/2021 12:11	File folder	
Controllers	19/05/2021 21:02	File folder	
Dto	19/05/2021 21:02	File folder	
Files	30/05/2021 22:20	File folder	
Models	19/05/2021 21:02	File folder	
obj	30/05/2021 22:20	File folder	
Properties	19/05/2021 21:02	File folder	
Tests	19/05/2021 21:22	File folder	
transformer	19/05/2021 21:02	File folder	
Views	19/05/2021 21:02	File folder	
wwwroot	19/05/2021 21:02	File folder	
appsettings.Development.json	19/05/2021 21:02	JSON File	1 KB
appsettings.json	19/05/2021 21:02	JSON File	1 KB
mock_compare.csproj	29/05/2021 13:01	Visual C# Project file	3 KB
mock_compare.csproj.user	19/05/2021 21:02	Per-User Project Opti...	1 KB
mock_compare.IsolatorCache.user	27/05/2021 23:09	Per-User Project Opti...	19 KB
mock_compare.sln	19/05/2021 21:02	Visual Studio Solution	2 KB
Program.cs	19/05/2021 21:02	Visual C# Source File	1 KB
Startup.cs	19/05/2021 21:02	Visual C# Source File	2 KB
test.csv	30/05/2021 22:20	Microsoft Excel Com...	1 KB

mock_compare.csproj - Notepad

File Edit Format View Help

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
    <IsPackable>false</IsPackable>
    <GenerateProgramFile>false</GenerateProgramFile>
  </PropertyGroup>

  <ItemGroup>
    <Compile Remove="Tests\NUnit\Models\**" />
    <Compile Remove="Tests\NUnit\Repository\**" />
    <Compile Remove="Tests\XUnit\Dto\**" />
    <Compile Remove="Tests\XUnit\Models\**" />
    <Content Remove="Tests\NUnit\Models\**" />
    <Content Remove="Tests\NUnit\Repository\**" />
    <Content Remove="Tests\XUnit\Dto\**" />
    <Content Remove="Tests\XUnit\Models\**" />
    <EmbeddedResource Remove="Tests\NUnit\Models\**" />
    <EmbeddedResource Remove="Tests\NUnit\Repository\**" />
    <EmbeddedResource Remove="Tests\XUnit\Dto\**" />
    <EmbeddedResource Remove="Tests\XUnit\Models\**" />
    <None Remove="Tests\NUnit\Models\**" />
    <None Remove="Tests\NUnit\Repository\**" />
    <None Remove="Tests\XUnit\Dto\**" />
    <None Remove="Tests\XUnit\Models\**" />
  </ItemGroup>
```

8. Podsumowanie

C#:

Według przeprowadzonych 5 testów i wyciągniętych średnich w języku c# Moq był szybszą metodą przeprowadzania testów oraz XUnit uzyskał lepsze czasy od NUnita.

Przy przeprowadzonych testach Moq był w każdym wypadku szybszy od JustMocka.

NUnitJustMock – 0 vs NUnitMoq – 5

Przy przeprowadzonych testach JustMock tylko jeden raz był szybszy od Moq.

XUnitJustMock – 1 vs XUnitMoq – 4

Java:

Według tabeli testów najszybszym sposobem na testy jest EasyMock, a zaraz po nim Mockito. Używanie AssertJ znacznie przedłuża wykonywanie testów, lecz AssertJEasyMock w większości przypadków nie ustępuje znacznie zwycięzcom EasyMock i Mockito.

AssertJEasyMock – 0 vs AssertJMockito – 0 vs EasyMock – 4 vs Mockito - 1

Java vs c#:

Według wyliczonej średniej czasu z testów określonych we wcześniejszych tabelach C# był trzykrotnie szybszy od Javy.

Średnia C#: 30.65 ms

Średnia Javy: 92.35 ms

Z przeprowadzonych testów wynika, że jeżeli zależy nam na czasie wykonywania testów to powinniśmy wybrać C#, lecz naszym zdaniem czas wykonywania testów to nie wszystko. Trzeba też wziąć pod uwagę czas jaki jest wymagany, aby dany test napisać i tutaj zdecydowanym zwycięzcą jest Java. Przysporzyła ona mniej problemów podczas pisania, a praca testera była wydajniejsza i w tym samym przedziale czasu mógł on napisać więcej testów.