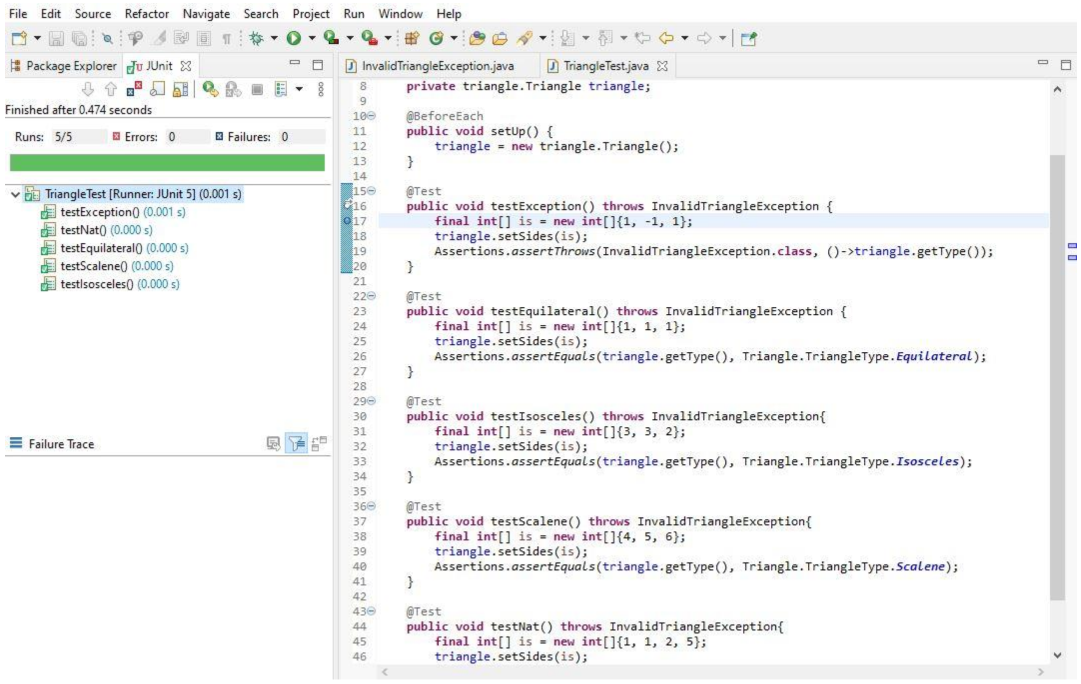
Unit testing TDDD04 LAB 1-report

Part 1

Test Cases:

Input1	Input2	Input3	Expected	Actual	Exception
1	2	3	invalid	invalid	no
3	2	1	invalid	invalid	no
2	2	2	Equilateral	Equilateral	no
A	B	C	invalid	Exception	Number format exception
1	2	a	invalid	Exception	Number format exception
abc	3	xyz	invalid	Exception	Number format exception
444444444444444	4	4	invalid	Exception	Number format exception
Empty	Empty	Empty	invalid	Exception	Number format exception

Test cases in code: pass



```
Finished after 0.474 seconds
Runs: 5/5    Errors: 0    Failures: 0

TriangleTest [Runner: JUnit 5] (0.001 s)
    testException() (0.001 s)
    testNat() (0.000 s)
    testEquilateral() (0.000 s)
    testScalene() (0.000 s)
    testIsosceles() (0.000 s)
```

```java
    private triangle.Triangle triangle;

    @BeforeEach
    public void setUp() {
        triangle = new triangle.Triangle();
    }

    @Test
    public void testException() throws InvalidTriangleException {
        final int[] is = new int[]{1, -1, 1};
        triangle.setSides(is);
        Assertions.assertThrows(InvalidTriangleException.class, ()->triangle.getType());
    }

    @Test
    public void testEquilateral() throws InvalidTriangleException {
        final int[] is = new int[]{1, 1, 1};
        triangle.setSides(is);
        Assertions.assertEquals(triangle.getType(), Triangle.TriangleType.Equilateral);
    }

    @Test
    public void testIsosceles() throws InvalidTriangleException{
        final int[] is = new int[]{3, 3, 2};
        triangle.setSides(is);
        Assertions.assertEquals(triangle.getType(), Triangle.TriangleType.Isosceles);
    }

    @Test
    public void testScalene() throws InvalidTriangleException{
        final int[] is = new int[]{4, 5, 6};
        triangle.setSides(is);
        Assertions.assertEquals(triangle.getType(), Triangle.TriangleType.Scalene);
    }

    @Test
    public void testNat() throws InvalidTriangleException{
        final int[] is = new int[]{1, 1, 2, 5};
        triangle.setSides(is);
```

Changes in triangle application:

1: I added try catch to handle wrong input error. For example if user enter char or string instead of int then this try catch will handle the error.

```java
try {
        intSides[i++] = Integer.parseInt(string);
}catch(NumberFormatException e) {
        System.err.println("invalid Input");
}
```

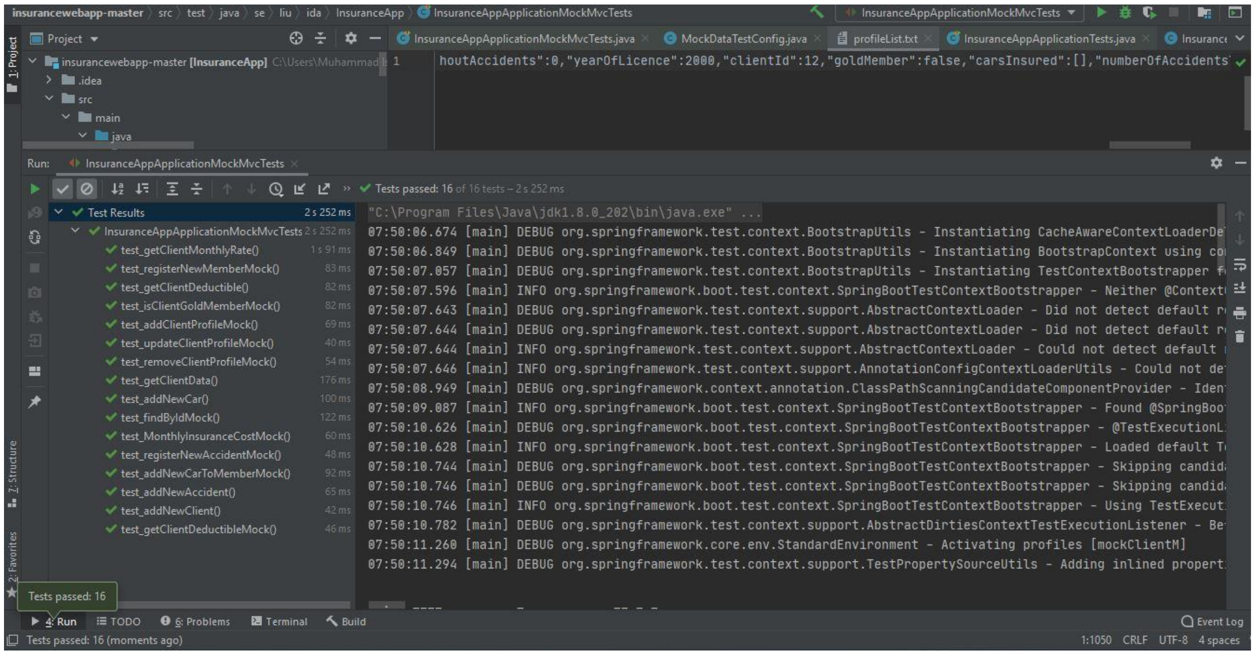
2: in below code it was throwing InvalidTriangleException but I change it with return because if it receive more than 3 parameters or less than 3 parameter for triangle it should return Nat a triangle.

```java
TriangleType result = TriangleType.NaT;

if (sides.length != 3) {
        //throw new InvalidTriangleException();
        return result;
}
```

Part 2

Mock tests:



```
houtAccidents":0,"yearOfLicence":2000,"clientId":12,"goldMember":false,"carsInsured":[],"numberOfAccidents'
```

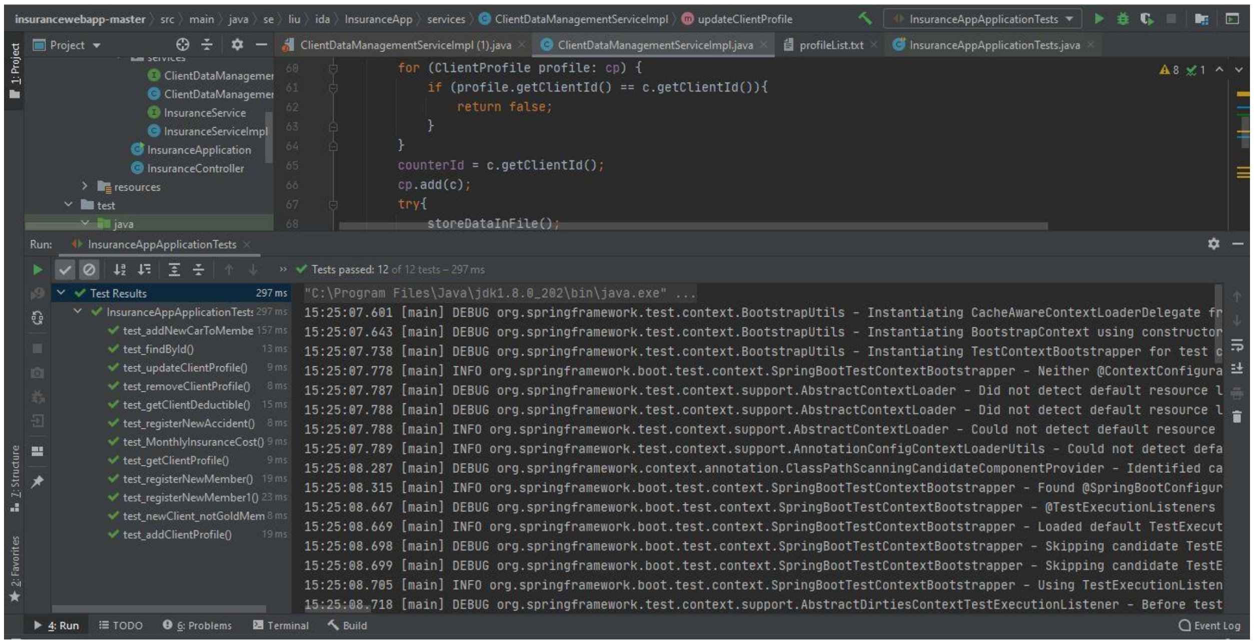
Tests passed: 16 of 16 tests – 2 s 252 ms

```
Test Results                                    2 s 252 ms
  InsuranceAppApplicationMockMvcTests           2 s 252 ms
    test_getClientMonthlyRate()                 1 s 91 ms
    test_registerNewMemberMock()                83 ms
    test_getClientDeductible()                  82 ms
    test_isClientGoldMemberMock()               82 ms
    test_addClientProfileMock()                 69 ms
    test_updateClientProfileMock()              40 ms
    test_removeClientProfileMock()              54 ms
    test_getClientData()                        176 ms
    test_addNewCar()                            100 ms
    test_findByIdMock()                         122 ms
    test_MonthlyInsuranceCostMock()             60 ms
    test_registerNewAccidentMock()              48 ms
    test_addNewCarToMemberMock()                92 ms
    test_addNewAccident()                       65 ms
    test_addNewClient()                         42 ms
    test_getClientDeductibleMock()              46 ms
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
07:50:06.674 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDe
07:50:06.849 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using co
07:50:07.057 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper f
07:50:07.596 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @Context
07:50:07.643 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default r
07:50:07.644 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default r
07:50:07.644 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default
07:50:07.646 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not de
07:50:08.949 [main] DEBUG org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider - Iden
07:50:09.087 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBoo
07:50:10.626 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionL
07:50:10.628 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default T
07:50:10.744 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Skipping candid
07:50:10.746 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Skipping candid
07:50:10.746 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecut
07:50:10.782 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Be
07:50:11.260 [main] DEBUG org.springframework.core.env.StandardEnvironment - Activating profiles [mockClientM]
07:50:11.294 [main] DEBUG org.springframework.test.context.support.TestPropertySourceUtils - Adding inlined propert
```

Tests passed: 16 (moments ago)

Integration Tests:



```java
        for (ClientProfile profile: cp) {
            if (profile.getClientId() == c.getClientId()){
                return false;
            }
        }
        counterId = c.getClientId();
        cp.add(c);
        try{
            storeDataInFile();
```

Tests passed: 12 of 12 tests – 297 ms

```
Test Results                                    297 ms
  InsuranceAppApplicationTests                  297 ms
    test_addNewCarToMembe                       157 ms
    test_findById()                             13 ms
    test_updateClientProfile()                  9 ms
    test_removeClientProfile()                  8 ms
    test_getClientDeductible()                  15 ms
    test_registerNewAccident()                  8 ms
    test_MonthlyInsuranceCost()                 9 ms
    test_getClientProfile()                     9 ms
    test_registerNewMember()                    19 ms
    test_registerNewMember1()                   23 ms
    test_newClient_notGoldMem                   8 ms
    test_addClientProfile()                     19 ms
```

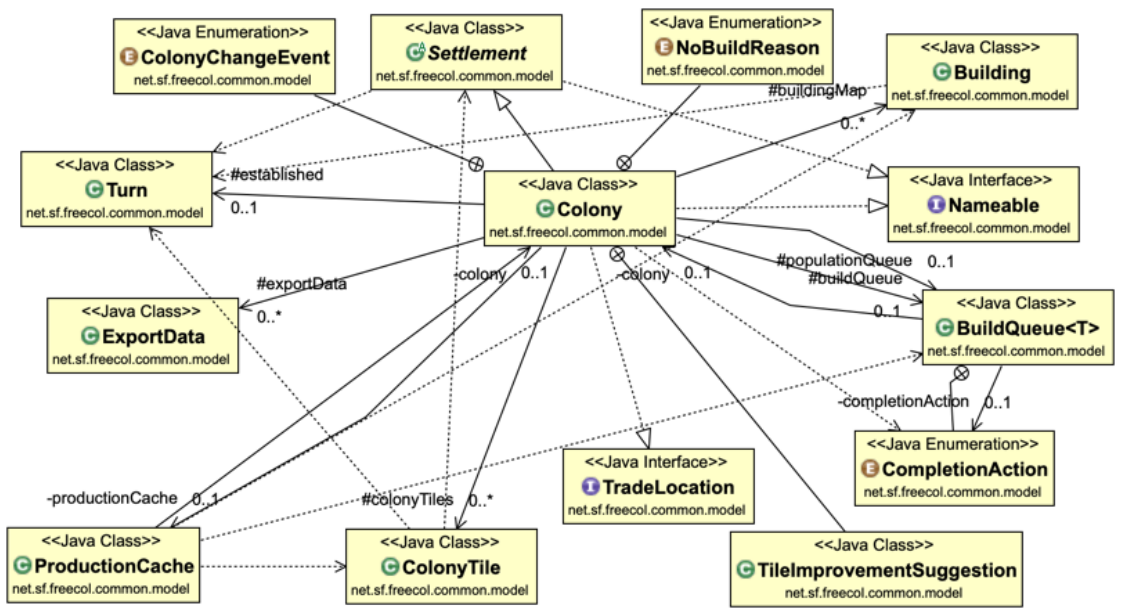
```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
15:25:07.601 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate fr
15:25:07.643 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor
15:25:07.738 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test c
15:25:07.778 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Neither @ContextConfigura
15:25:07.787 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource l
15:25:07.788 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource l
15:25:07.788 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource
15:25:07.789 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect defa
15:25:08.287 [main] DEBUG org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider - Identified ca
15:25:08.315 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootConfigur
15:25:08.667 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners
15:25:08.669 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecut
15:25:08.698 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Skipping candidate TestE
15:25:08.699 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Skipping candidate TestE
15:25:08.705 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Using TestExecutionListen
15:25:08.718 [main] DEBUG org.springframework.test.context.support.AbstractDirtiesContextTestExecutionListener - Before test
```

Motivation:

First I wrote Mock test and their stubs, by writing this I was able to understand the behaviors of functions/methods. Then I wrote some integration tests that all fails in start but after implementing code all tests are passed. As shown in above images. I try to write all cases test so I can achieve full coverage of testing.

Part 3-FreeCol

Dependency diagram:



<<Java Enumeration>> ColonyChangeEvent net.sf.freecol.common.model
<<Java Class>> Settlement net.sf.freecol.common.model
<<Java Enumeration>> NoBuildReason net.sf.freecol.common.model
<<Java Class>> Building net.sf.freecol.common.model
<<Java Class>> Turn net.sf.freecol.common.model
<<Java Class>> Colony net.sf.freecol.common.model
<<Java Interface>> Nameable net.sf.freecol.common.model
<<Java Class>> ExportData net.sf.freecol.common.model
<<Java Class>> BuildQueue<T> net.sf.freecol.common.model
<<Java Enumeration>> CompletionAction net.sf.freecol.common.model
<<Java Interface>> TradeLocation net.sf.freecol.common.model
<<Java Class>> ProductionCache net.sf.freecol.common.model
<<Java Class>> ColonyTile net.sf.freecol.common.model
<<Java Class>> TileImprovementSuggestion net.sf.freecol.common.model
#buildingMap
0..*
#established
0..1
#exportData
0..*
-colony
0..1
#populationQueue
#buildQueue
0..1
-completionAction
-productionCache
#colonyTiles

Questions:

1: Have all statements in the test methods been executed? If not, what does that tell us?

This shows, our tests are not running through all the software. It means skipping some parts of the code.

2: Suggest a design change that would simplify testing Colony methods?

In above design the thing that creating problem in testing is high dependency of classes on other classes. We can change that to simplify the testing. We can use TDD approach to minimize it. By trying to create individuals objects of each class.

3: Describe why it is a problem, in concrete terms and in this situation, to have high coupling?

In this case when we wrote test cases for colony class, we can't access the dependent classes freely. Because when we create an abject of a dependent class, it can depending on other class. That make it harder to reach the depth of code for testing. Reason is high coupling between classes.

4: What are the effects on your test code? On your code coverage? Could there be issues in understanding the reason behind failing tests?

Test code will be difficult to test, like we can test main class but for other classes it could be difficult to test all methods and difficult to write test for deeper methods. This reduce the code coverage and might be chances to skip part of code in testing. Yes, it will be difficult because code complexity is high due to high coupling and find a reason behind failing of test is also difficult.