

Lab: More Mocking with Mockito

Spies, Consecutive Calls, and Practice

Setup

1. Create a Maven or Gradle project and give it a name of your choice.
2. Download the [files for this lab](#) and copy them into your project's 'src/main/java' folder, making sure to keep the structure of the project's packages intact.
3. Import the following dependencies
 - a. [JUnit](#) (junit-jupiter-api)
 - b. [Mockito](#)
 - c. [JUnit-params](#) (Optional: only if you want to use annotations)
4. Note: There are known issues with both the Maven and Gradle integrations in IntelliJ that may cause problems with running tests or compiling your code. Refer to the [Setup and Test Running Issues](#) document if you encounter any of these issues.

Lab

In this lab, you will be testing the provided WeatherStation class. This is a simple program that takes measurement data from instruments in the local area and uses that data to determine if there is a possibility of a catastrophic weather event. None of the instruments inside the 'instruments' package are currently implemented so you will be using mocks to provide data to the station to make sure the warning systems are working as intended.

One tool that the Weather Station uses that has already been written is the SatelliteUplink class. This is a tool used to provide secondary data and info on nearby stations in order to make suggestions on whether the station should or should not in fact issue a warning. Since the SatelliteUplink class is already written, you will be using the Mockito Spy feature in order to verify that the Uplink is in fact working as intended. The Uplink has already been provided with dummy data so you do not need to mock it to provide data for the SatelliteUplink class.

You will be responsible for writing tests for each of the public methods in the WeatherStation class (not including setters) according to the following requirements:

- **Your tests should achieve 100% line coverage for WeatherStation and SatelliteUplink.**
 - **Note:** You should **not** create a SatelliteUplinkTest.java file. Line coverage for this class will be achieved by the fact that it is called by the WeatherStation class,

so its methods will be called when your tests call methods on the WeatherStation class.

- You must use 'verify' to confirm that each of the following is called in at least one test:
 - checkNearbyAreaStorms is called in the runStormCheckForArea method.
 - checkNearbyAreaTornados is called in the runTornadoCheckForArea method.
 - For each mock you create for an instrument, verify that the methods on that instrument are called.
 - **Note:** For all verifies on the SatelliteUplink class, you will need to use spies as discussed in the reading.
- For the calibration test methods (anemometerCalibrationCheck() and temperatureCalibrationCheck()) you will need two tests.
 - one positive
 - one negative.
 - **Hint:** This can only be done with consecutive calls as described in the reading.

To help you with your tests, we are providing the following information on the SatelliteUplink class and how each method works.

DummyData

The dummy data provided is as follows:

- Air Pressure: 800
- Humidity : 30
- Wind Speed: 15
- Temperature: 75

This data will always suggest sending a warning if other stations are checked for weather warnings.

runStormCheckForArea(...) Method

Averages all parameters passed with the corresponding satellite datum value. A storm warning will be suggested based on the following criterion:

Humidity average exceeds 30 and temperature average exceeds 70

OR

Humidity average exceeds 30 and air pressure is less than 900

If none of these conditions are met, we check nearby areas (via Satellite) to see if they have issued a storm warning and follow their lead.

runTornadoCheckForArea(...) Method

Averages all parameters passed with the corresponding satellite datum value. Also checks the difference in air pressure against earlier readings (earlier reading will be 800) and do the same

for humidity (earlier reading will be 30, same as current). A tornado warning will be suggested based on the following criterion:

Difference in air pressure is above 150

OR

Humidity difference exceeds 5 AND wind speed average is greater than 15

If none of these conditions are met then we check nearby areas (via Satellite) to see if they have issued a tornado warning and follow their lead.

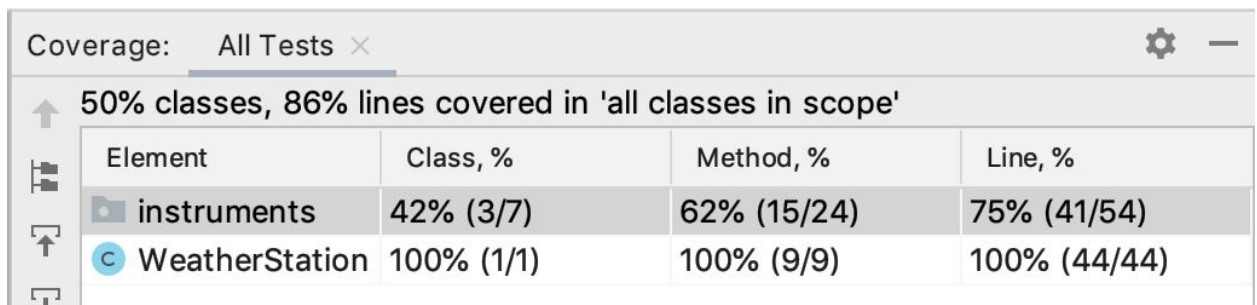
WARNING

There are 3 bugs in the project. You can look for them if you like, but they are specifically designed so that if you are using verify and proper mocking techniques you will be able to easily find and fix them. Part of the assignment is to fix the bugs.

Submission

To receive credit for the lab submit the following items to Canvas:

1. Two screenshots showing all of your tests passing for WeatherStation and SatelliteUplink.



Coverage: All Tests ×				
50% classes, 86% lines covered in 'all classes in scope'				
Element	Class, %	Method, %	Line, %	
instruments	42% (3/7)	62% (15/24)	75% (41/54)	
WeatherStation	100% (1/1)	100% (9/9)	100% (44/44)	

2. WeatherStationTest.java
3. Your modified WeatherStation and SatelliteUplink classes.