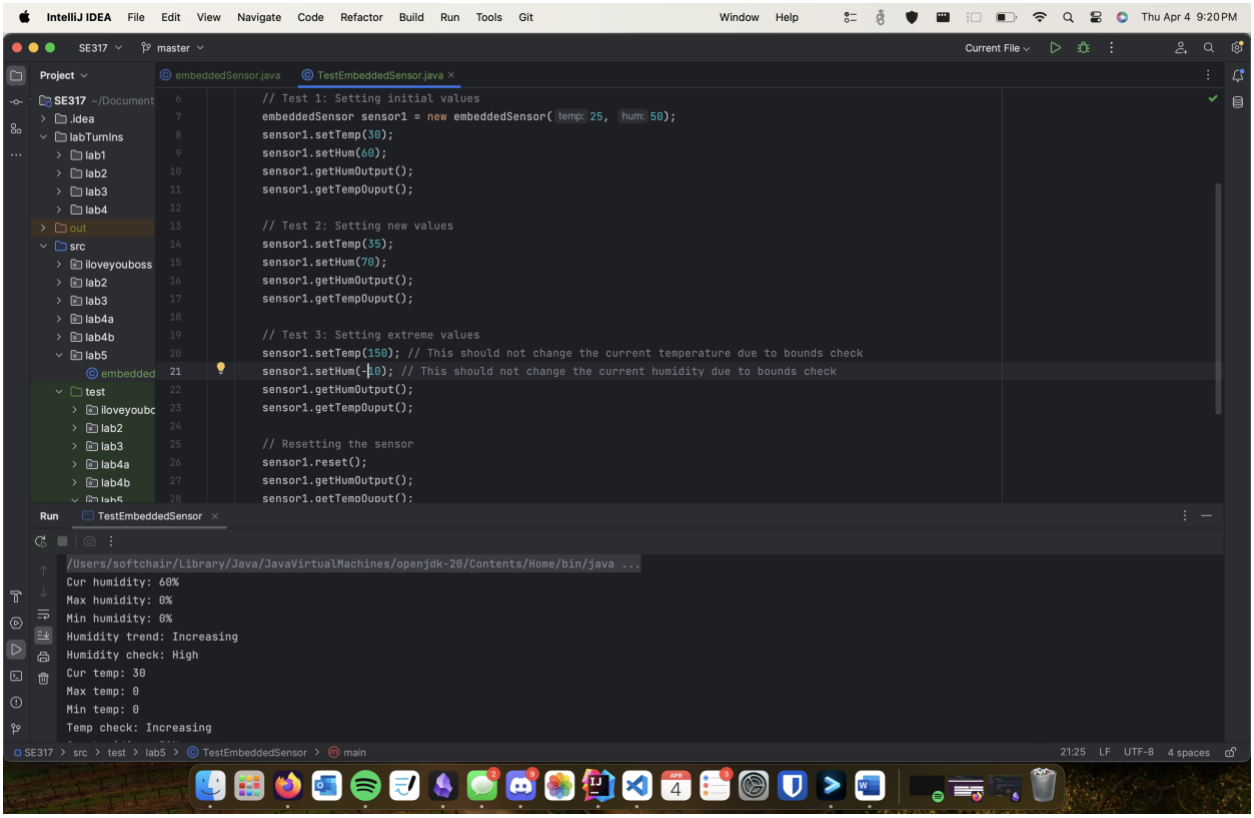
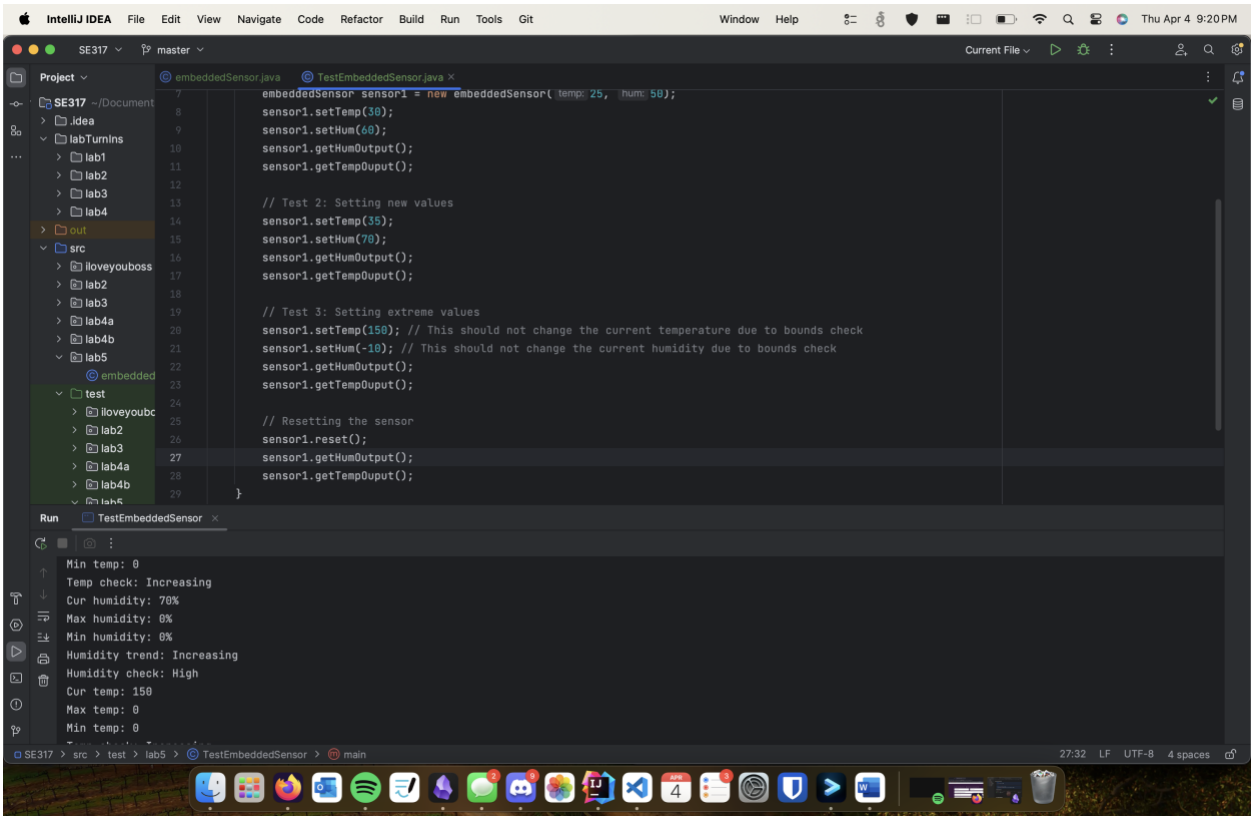
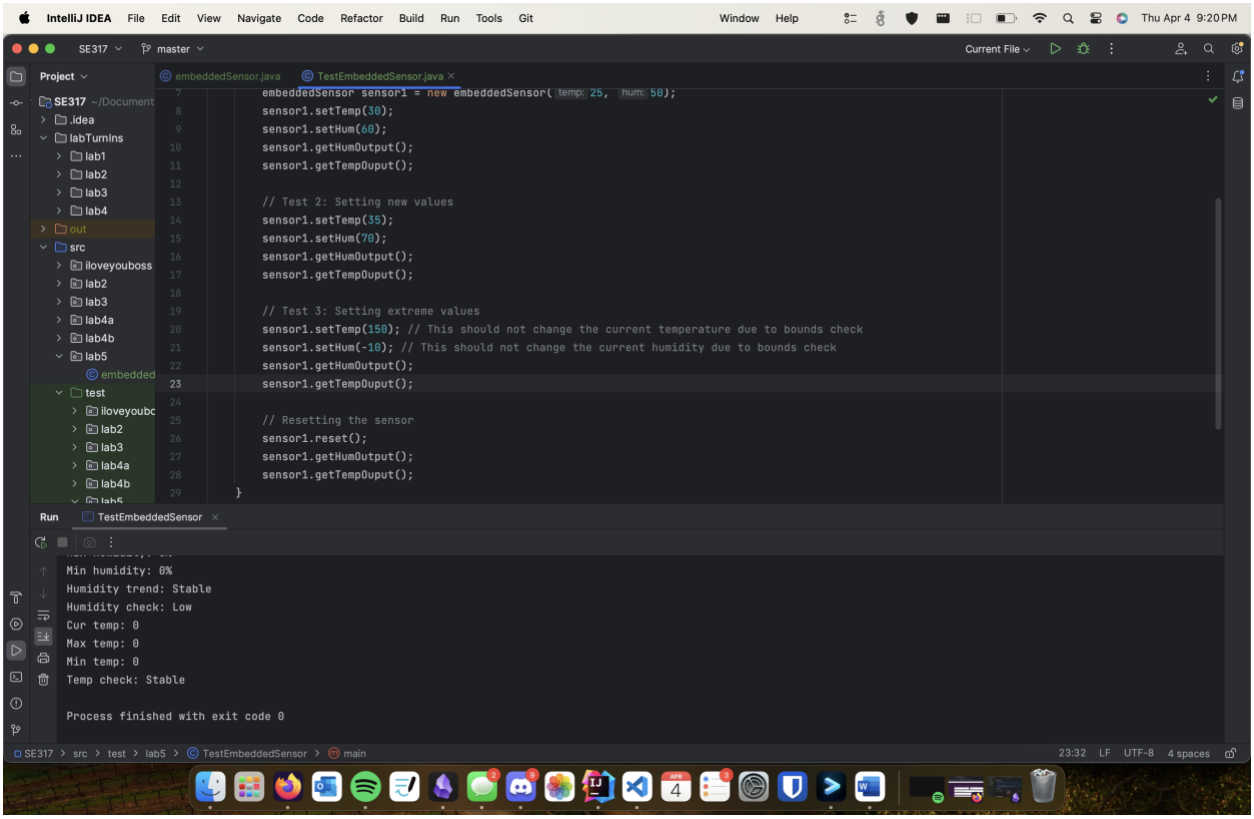
Lab 5

Camden Fergen

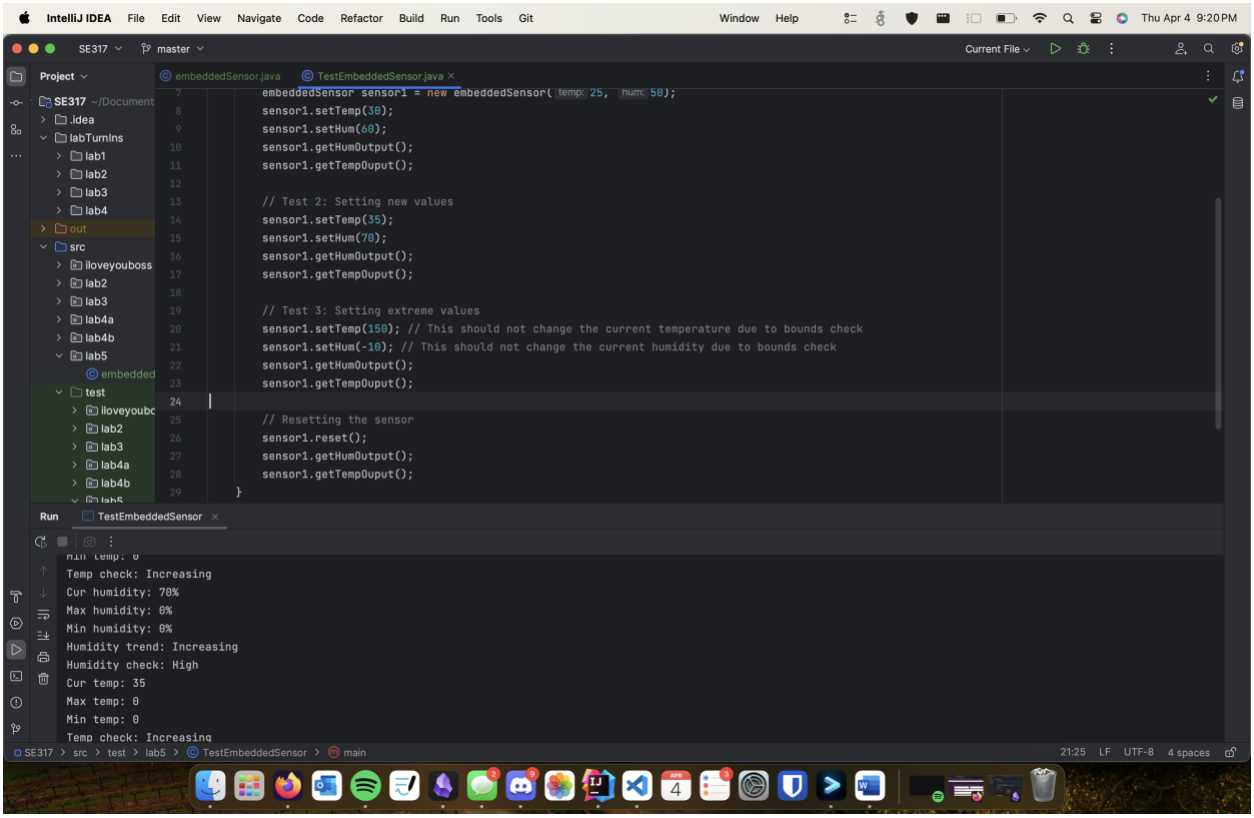
Part 1:



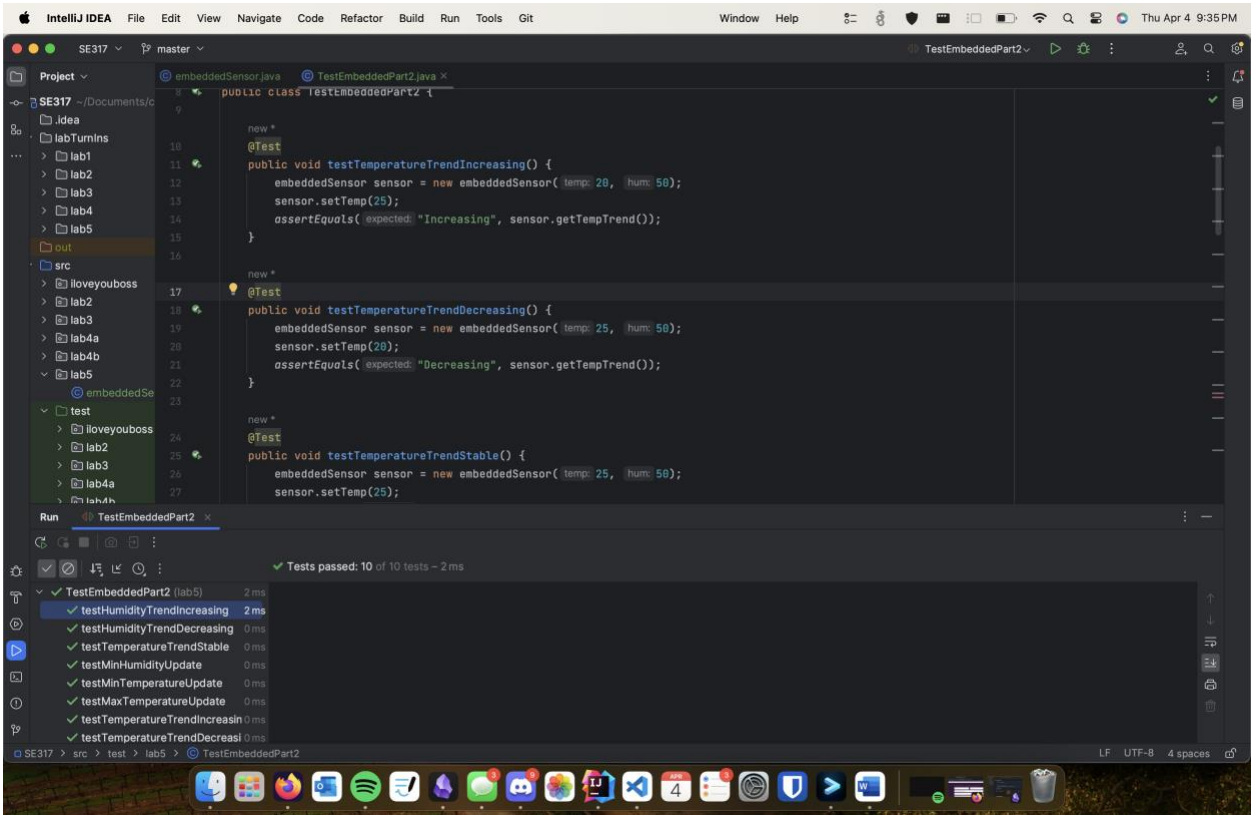

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help Thu Apr 4 9:20 PM
SE317 master Current File
Project
embeddedSensor.java TestEmbeddedSensor.java
SE317 ~/Document
.idea
labTurnIns
lab1
lab2
lab3
lab4
out
src
iloveyouboss
lab2
lab3
lab4a
lab4b
lab5
embedded
test
iloveyoubc
lab2
lab3
lab4a
lab4b
embeddedSensor sensor1 = new embeddedSensor(temp: 25, hum: 50);
sensor1.setTemp(30);
sensor1.setHum(60);
sensor1.getHumOutput();
sensor1.getTempOuput();
// Test 2: Setting new values
sensor1.setTemp(35);
sensor1.setHum(70);
sensor1.getHumOutput();
sensor1.getTempOuput();
// Test 3: Setting extreme values
sensor1.setTemp(150); // This should not change the current temperature due to bounds check
sensor1.setHum(-10); // This should not change the current humidity due to bounds check
sensor1.getHumOutput();
sensor1.getTempOuput();
// Resetting the sensor
sensor1.reset();
sensor1.getHumOutput();
sensor1.getTempOuput();
}
Run TestEmbeddedSensor
Min temp: 0
Temp check: Increasing
Cur humidity: 70%
Max humidity: 0%
Min humidity: 0%
Humidity trend: Increasing
Humidity check: High
Cur temp: 150
Max temp: 0
Min temp: 0
SE317 > src > test > lab5 > TestEmbeddedSensor > main
27:32 LF UTF-8 4 spaces

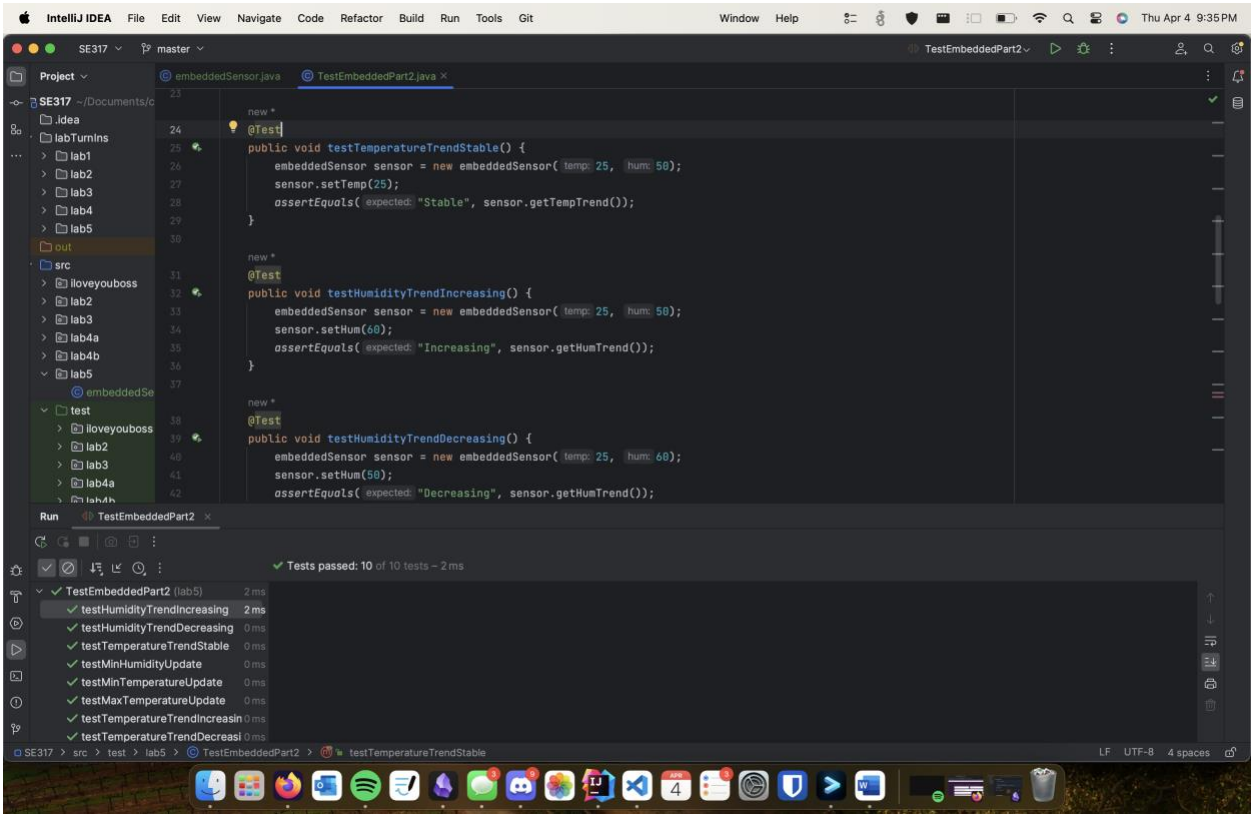

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help Thu Apr 4 9:20 PM
SE317 master Current File
Project
embeddedSensor.java TestEmbeddedSensor.java
SE317 ~/Document
.idea
labTurnIns
lab1
lab2
lab3
lab4
out
src
iloveyouboss
lab2
lab3
lab4a
lab4b
lab5
embedded
test
iloveyoubc
lab2
lab3
lab4a
lab4b
embeddedSensor sensor1 = new embeddedSensor(temp: 25, hum: 50);
sensor1.setTemp(30);
sensor1.setHum(60);
sensor1.getHumOutput();
sensor1.getTempOuput();
// Test 2: Setting new values
sensor1.setTemp(35);
sensor1.setHum(70);
sensor1.getHumOutput();
sensor1.getTempOuput();
// Test 3: Setting extreme values
sensor1.setTemp(150); // This should not change the current temperature due to bounds check
sensor1.setHum(-10); // This should not change the current humidity due to bounds check
sensor1.getHumOutput();
sensor1.getTempOuput();
// Resetting the sensor
sensor1.reset();
sensor1.getHumOutput();
sensor1.getTempOuput();
}
Run TestEmbeddedSensor
Min humidity: 0%
Humidity trend: Stable
Humidity check: Low
Cur temp: 0
Max temp: 0
Min temp: 0
Temp check: Stable
Process finished with exit code 0
SE317 > src > test > lab5 > TestEmbeddedSensor > main
23:32 LF UTF-8 4 spaces

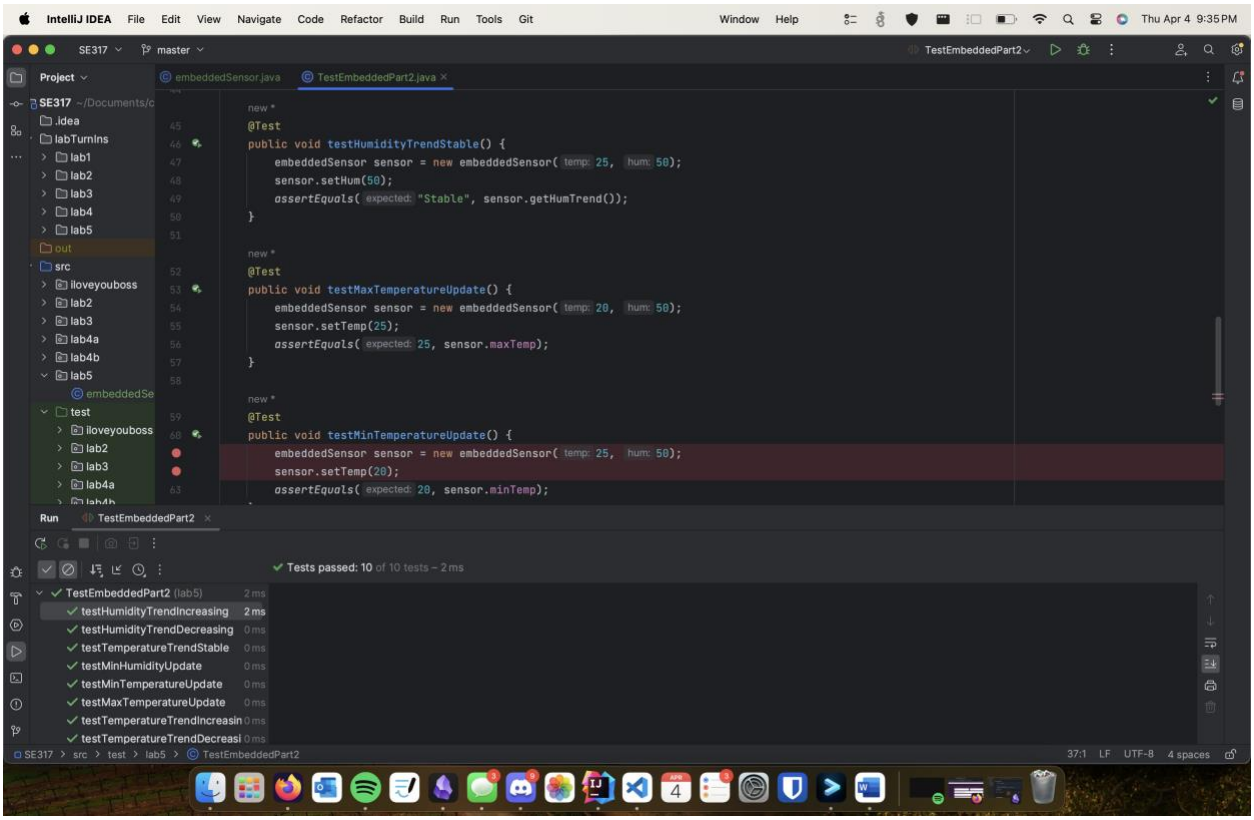
C

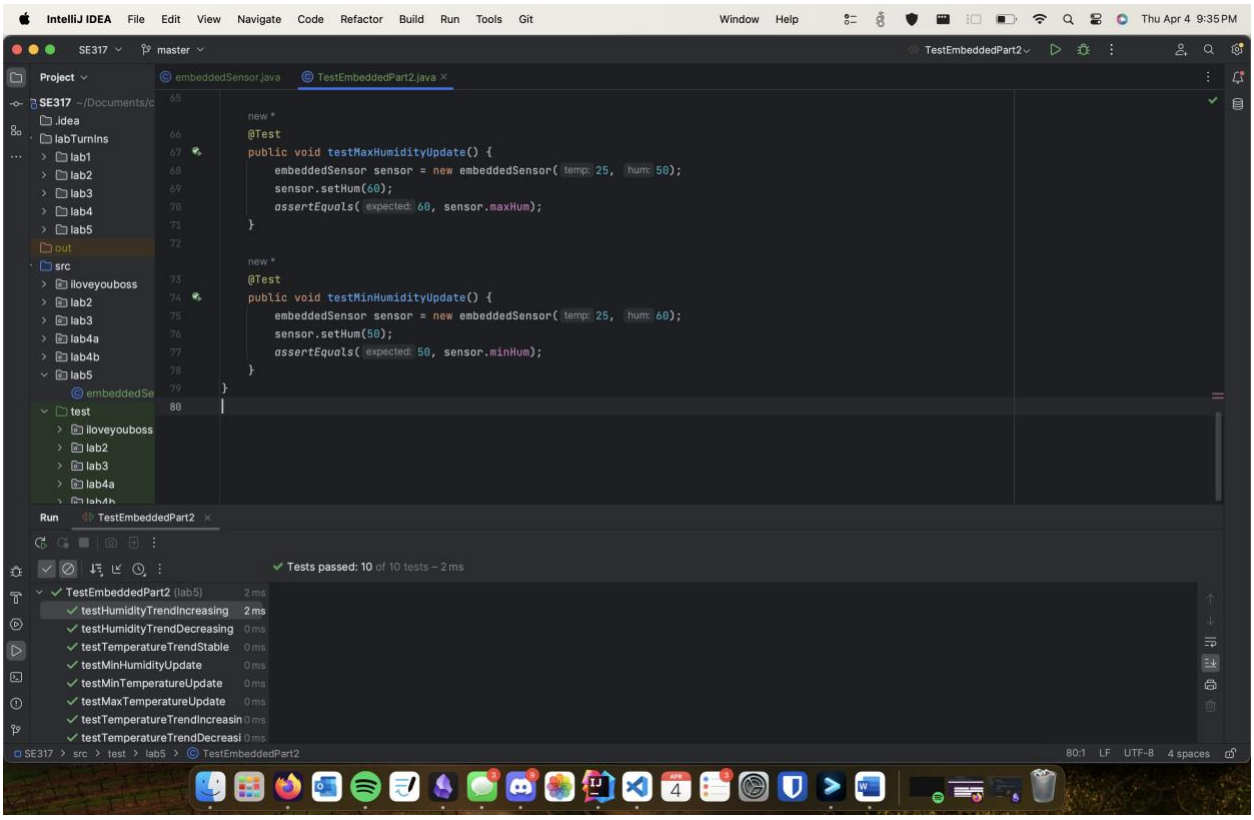

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help Thu Apr 4 9:20 PM
SE317 master Current File
Project
embeddedSensor.java TestEmbeddedSensor.java
SE317 ~/Document
.idea
labTurnIns
lab1
lab2
lab3
lab4
out
src
iloveyouboss
lab2
lab3
lab4a
lab4b
lab5
embedded
test
iloveyoubc
lab2
lab3
lab4a
lab4b
embeddedSensor sensor1 = new embeddedSensor(temp: 25, hum: 50);
sensor1.setTemp(30);
sensor1.setHum(60);
sensor1.getHumOutput();
sensor1.getTempOuput();
// Test 2: Setting new values
sensor1.setTemp(35);
sensor1.setHum(70);
sensor1.getHumOutput();
sensor1.getTempOuput();
// Test 3: Setting extreme values
sensor1.setTemp(150); // This should not change the current temperature due to bounds check
sensor1.setHum(-10); // This should not change the current humidity due to bounds check
sensor1.getHumOutput();
sensor1.getTempOuput();
// Resetting the sensor
sensor1.reset();
sensor1.getHumOutput();
sensor1.getTempOuput();
}
Run TestEmbeddedSensor
Temp check: Increasing
Cur humidity: 70%
Max humidity: 0%
Min humidity: 0%
Humidity trend: Increasing
Humidity check: High
Cur temp: 35
Max temp: 0
Min temp: 0
Temp check: Increasing
SE317 > src > test > lab5 > TestEmbeddedSensor > main
21:25 LF UTF-8 4 spaces

Part 2A:


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help Thu Apr 4 9:35 PM
SE317 master TestEmbeddedPart2
Project
embeddedSensor.java TestEmbeddedPart2.java
public class TestEmbeddedPart2 {
new *
@Test
public void testTemperatureTrendIncreasing() {
embeddedSensor sensor = new embeddedSensor(temp: 20, hum: 50);
sensor.setTemp(25);
assertEquals(expected: "Increasing", sensor.getTempTrend());
}
new *
@Test
public void testTemperatureTrendDecreasing() {
embeddedSensor sensor = new embeddedSensor(temp: 25, hum: 50);
sensor.setTemp(20);
assertEquals(expected: "Decreasing", sensor.getTempTrend());
}
new *
@Test
public void testTemperatureTrendStable() {
embeddedSensor sensor = new embeddedSensor(temp: 25, hum: 50);
sensor.setTemp(25);
Run TestEmbeddedPart2
Tests passed: 10 of 10 tests – 2 ms
TestEmbeddedPart2 (lab5) 2 ms
testHumidityTrendIncreasing 2 ms
testHumidityTrendDecreasing 0 ms
testTemperatureTrendStable 0 ms
testMinHumidityUpdate 0 ms
testMinTemperatureUpdate 0 ms
testMaxTemperatureUpdate 0 ms
testTemperatureTrendIncreasin 0 ms
testTemperatureTrendDecreasi 0 ms
SE317 src test lab5 TestEmbeddedPart2
LF UTF-8 4 spaces


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help Thu Apr 4 9:35 PM
SE317 master TestEmbeddedPart2
Project
embeddedSensor.java TestEmbeddedPart2.java
new *
@Test
public void testTemperatureTrendStable() {
embeddedSensor sensor = new embeddedSensor(temp: 25, hum: 50);
sensor.setTemp(25);
assertEquals(expected: "Stable", sensor.getTempTrend());
}
new *
@Test
public void testHumidityTrendIncreasing() {
embeddedSensor sensor = new embeddedSensor(temp: 25, hum: 50);
sensor.setHum(60);
assertEquals(expected: "Increasing", sensor.getHumTrend());
}
new *
@Test
public void testHumidityTrendDecreasing() {
embeddedSensor sensor = new embeddedSensor(temp: 25, hum: 60);
sensor.setHum(50);
assertEquals(expected: "Decreasing", sensor.getHumTrend());
Run TestEmbeddedPart2
Tests passed: 10 of 10 tests – 2 ms
TestEmbeddedPart2 (lab5) 2 ms
testHumidityTrendIncreasing 2 ms
testHumidityTrendDecreasing 0 ms
testTemperatureTrendStable 0 ms
testMinHumidityUpdate 0 ms
testMinTemperatureUpdate 0 ms
testMaxTemperatureUpdate 0 ms
testTemperatureTrendIncreasin 0 ms
testTemperatureTrendDecreasi 0 ms
SE317 src test lab5 TestEmbeddedPart2 testTemperatureTrendStable
LF UTF-8 4 spaces





Test Criterion

Temperature Trend: The system should correctly identify the trend of temperature changes (increasing, decreasing, or stable).

Humidity Trend: The system should correctly identify the trend of humidity changes (increasing, decreasing, or stable).

Maximum Temperature: The system should correctly update and display the maximum temperature value.

Minimum Temperature: The system should correctly update and display the minimum temperature value.

Maximum Humidity: The system should correctly update and display the maximum humidity value.

Minimum Humidity: The system should correctly update and display the minimum humidity value.

Test Requirement (TR) Set

TR1: Verify that the temperature trend is correctly identified as increasing when the current temperature is higher than the last temperature.

TR2: Verify that the temperature trend is correctly identified as decreasing when the current temperature is lower than the last temperature.

TR3: Verify that the temperature trend is correctly identified as stable when the current temperature is equal to the last temperature.

TR4: Verify that the humidity trend is correctly identified as increasing when the current humidity is higher than the last humidity.

TR5: Verify that the humidity trend is correctly identified as decreasing when the current humidity is lower than the last humidity.

TR6: Verify that the humidity trend is correctly identified as stable when the current humidity is equal to the last humidity.

TR7: Verify that the maximum temperature is correctly updated and displayed when a new higher temperature is set.

TR8: Verify that the minimum temperature is correctly updated and displayed when a new lower temperature is set.

TR9: Verify that the maximum humidity is correctly updated and displayed when a new higher humidity is set.

TR10: Verify that the minimum humidity is correctly updated and displayed when a new lower humidity is set.

Test Set

TS1: Test temperature trend with increasing values.

TS2: Test temperature trend with decreasing values.

TS3: Test temperature trend with stable values.

TS4: Test humidity trend with increasing values.

TS5: Test humidity trend with decreasing values.

TS6: Test humidity trend with stable values.

TS7: Test maximum temperature update.

TS8: Test minimum temperature update.

TS9: Test maximum humidity update.

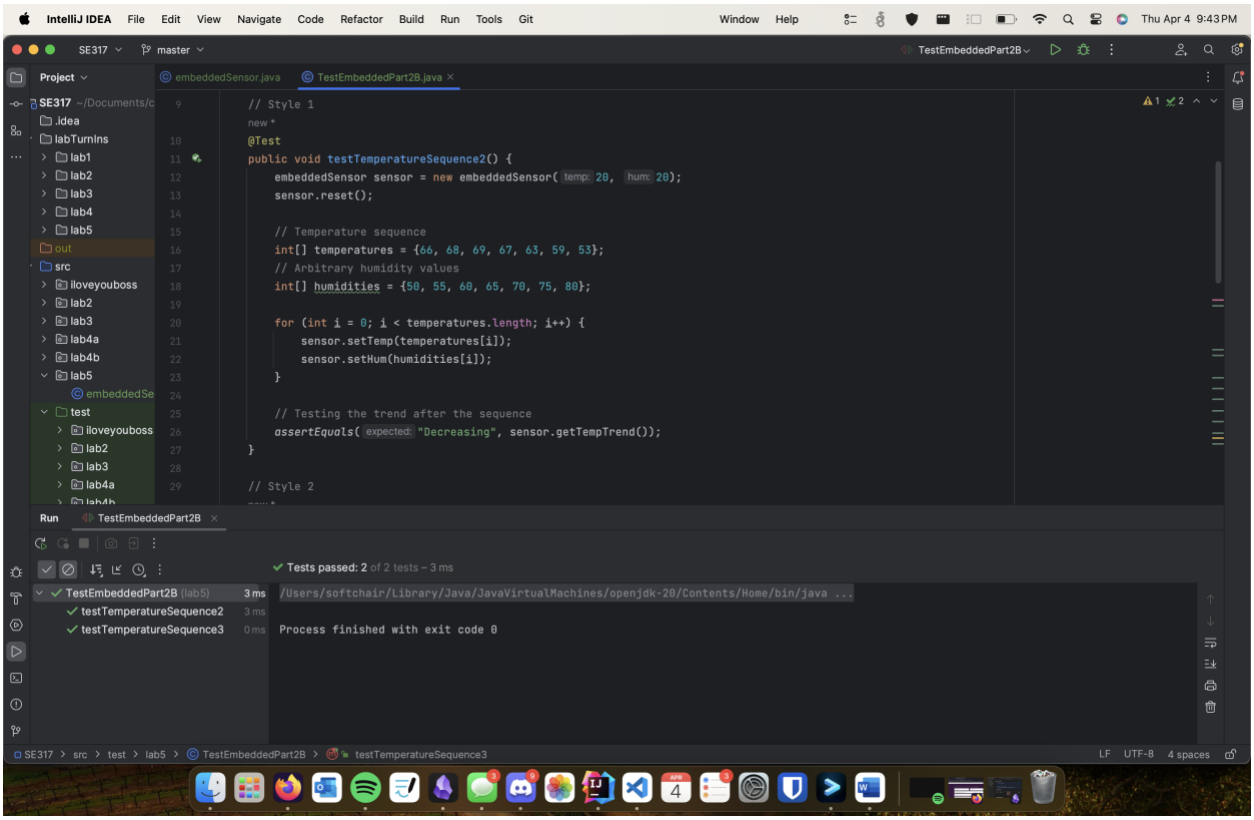
TS10: Test minimum humidity update.

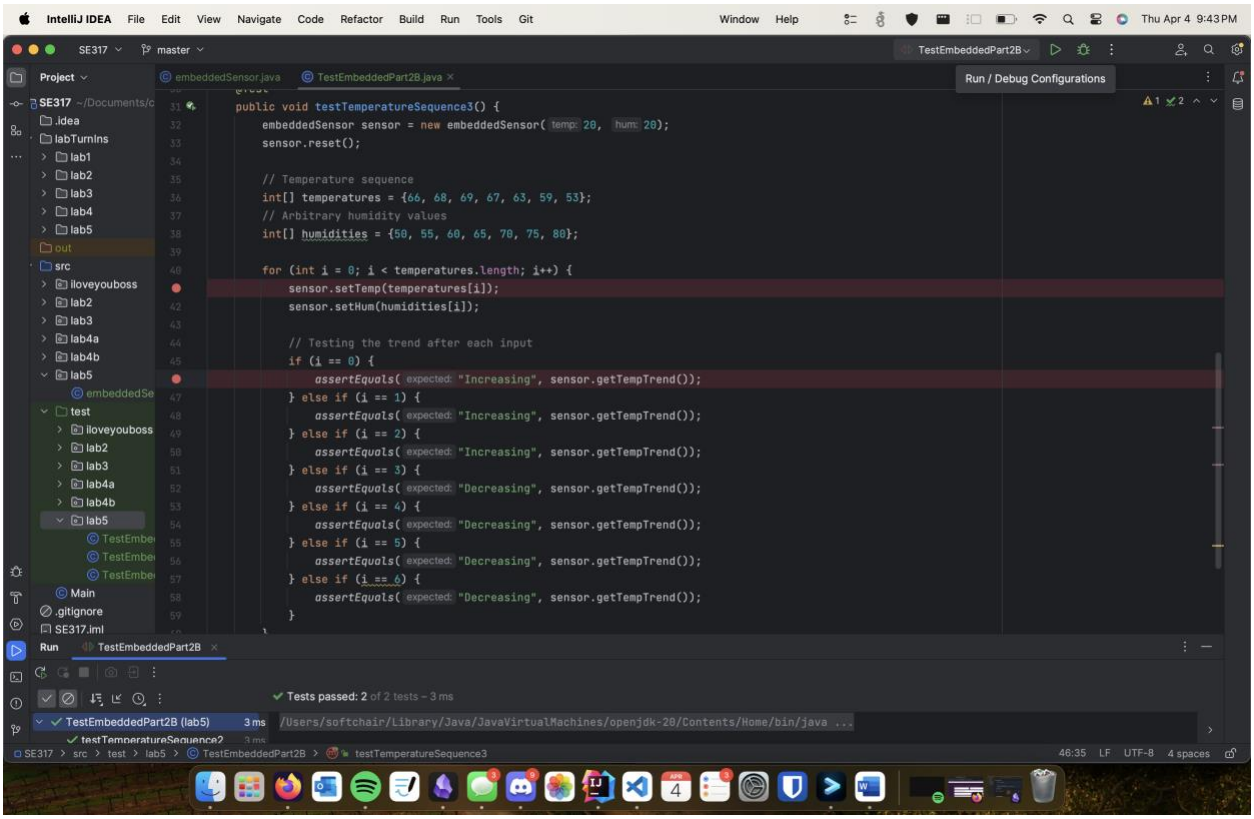
Total Number of Test Criteria: 10

Total Number of Test Cases: 10

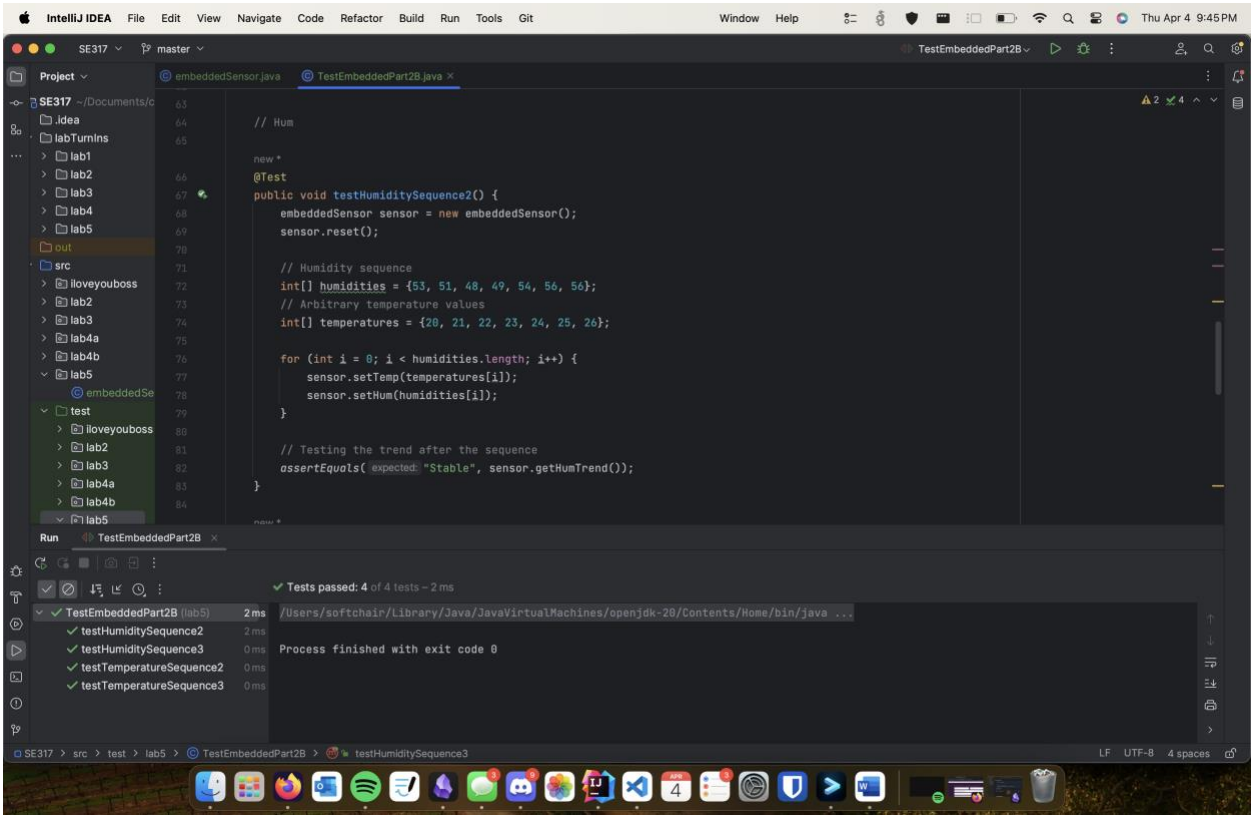
Part2b:

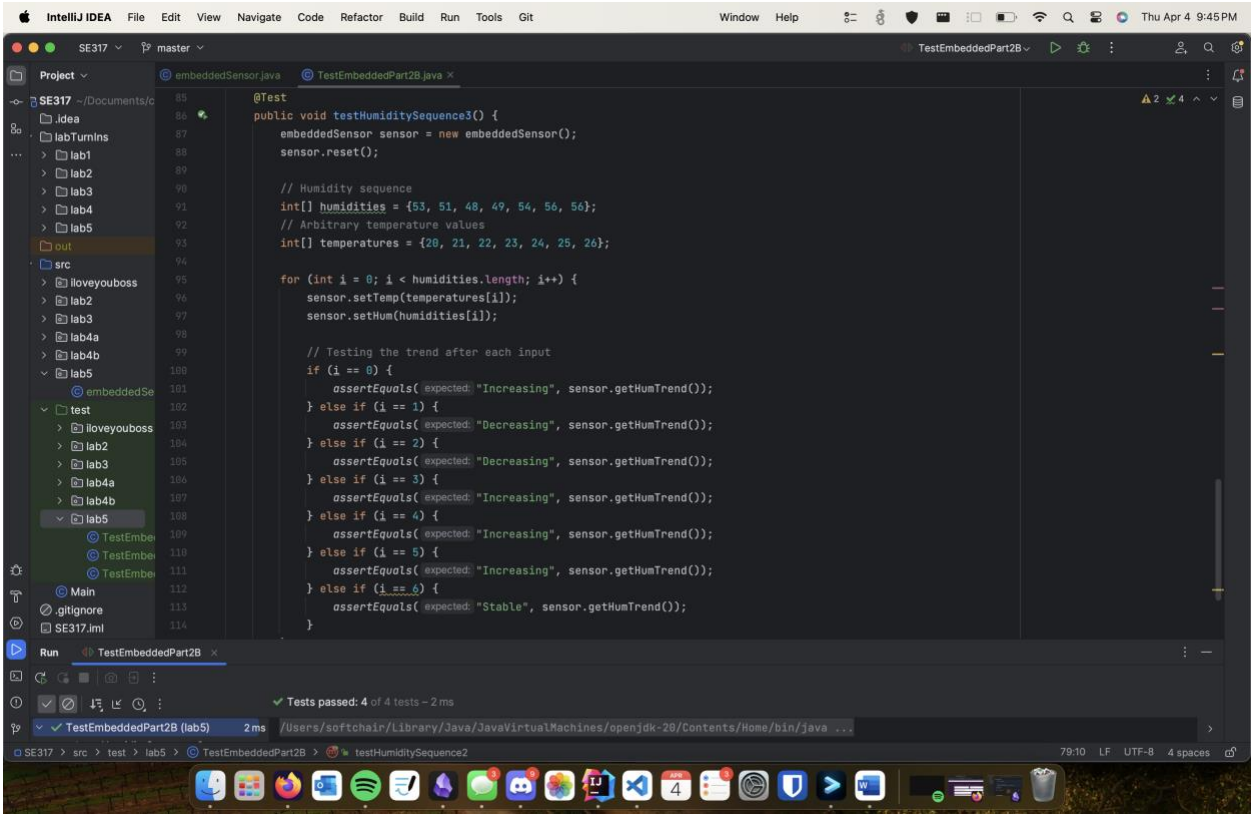
Temp:


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help Thu Apr 4 9:43 PM
SE317 master TestEmbeddedPart2B
Project
embeddedSensor.java TestEmbeddedPart2B.java
// Style 1
new *
@Test
public void testTemperatureSequence2() {
embeddedSensor sensor = new embeddedSensor(temp: 20, hum: 20);
sensor.reset();
// Temperature sequence
int[] temperatures = {66, 68, 69, 67, 63, 59, 53};
// Arbitrary humidity values
int[] humidities = {50, 55, 60, 65, 70, 75, 80};
for (int i = 0; i < temperatures.length; i++) {
sensor.setTemp(temperatures[i]);
sensor.setHum(humidities[i]);
}
// Testing the trend after the sequence
assertEquals(expected: "Decreasing", sensor.getTempTrend());
}
// Style 2
Run TestEmbeddedPart2B
Tests passed: 2 of 2 tests – 3 ms
TestEmbeddedPart2B (lab5) 3 ms
testTemperatureSequence2 3 ms
testTemperatureSequence3 0 ms
/Users/softchair/Library/Java/JavaVirtualMachines/openjdk-20/Contents/Home/bin/java ...
Process finished with exit code 0
SE317 > src > test > lab5 > TestEmbeddedPart2B > testTemperatureSequence3
LF UTF-8 4 spaces


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help Thu Apr 4 9:43 PM
SE317 master TestEmbeddedPart2B
Run / Debug Configurations
public void testTemperatureSequence3() {
embeddedSensor sensor = new embeddedSensor(temp: 20, hum: 20);
sensor.reset();
// Temperature sequence
int[] temperatures = {66, 68, 69, 67, 63, 59, 53};
// Arbitrary humidity values
int[] humidities = {50, 55, 60, 65, 70, 75, 80};
for (int i = 0; i < temperatures.length; i++) {
sensor.setTemp(temperatures[i]);
sensor.setHum(humidities[i]);
// Testing the trend after each input
if (i == 0) {
assertEquals(expected: "Increasing", sensor.getTempTrend());
} else if (i == 1) {
assertEquals(expected: "Increasing", sensor.getTempTrend());
} else if (i == 2) {
assertEquals(expected: "Increasing", sensor.getTempTrend());
} else if (i == 3) {
assertEquals(expected: "Decreasing", sensor.getTempTrend());
} else if (i == 4) {
assertEquals(expected: "Decreasing", sensor.getTempTrend());
} else if (i == 5) {
assertEquals(expected: "Decreasing", sensor.getTempTrend());
} else if (i == 6) {
assertEquals(expected: "Decreasing", sensor.getTempTrend());
}
Tests passed: 2 of 2 tests – 3 ms
TestEmbeddedPart2B (lab5) 3 ms
/Users/softchair/Library/Java/JavaVirtualMachines/openjdk-20/Contents/Home/bin/java ...
SE317 > src > test > lab5 > TestEmbeddedPart2B > testTemperatureSequence3
46:35 LF UTF-8 4 spaces

Hum:


// Hum
@Test
public void testHumiditySequence2() {
embeddedSensor sensor = new embeddedSensor();
sensor.reset();
// Humidity sequence
int[] humidities = {53, 51, 48, 49, 54, 56, 56};
// Arbitrary temperature values
int[] temperatures = {20, 21, 22, 23, 24, 25, 26};
for (int i = 0; i < humidities.length; i++) {
sensor.setTemp(temperatures[i]);
sensor.setHum(humidities[i]);
}
// Testing the trend after the sequence
assertEquals(expected: "Stable", sensor.getHumTrend());
}
Tests passed: 4 of 4 tests – 2 ms
TestEmbeddedPart2B (lab5) 2 ms
testHumiditySequence2 2 ms
testHumiditySequence3 0 ms
testTemperatureSequence2 0 ms
testTemperatureSequence3 0 ms
/Users/softchair/Library/Java/JavaVirtualMachines/openjdk-20/Contents/Home/bin/java ...
Process finished with exit code 0


@Test
public void testHumiditySequence3() {
embeddedSensor sensor = new embeddedSensor();
sensor.reset();
// Humidity sequence
int[] humidities = {53, 51, 48, 49, 54, 56, 56};
// Arbitrary temperature values
int[] temperatures = {20, 21, 22, 23, 24, 25, 26};
for (int i = 0; i < humidities.length; i++) {
sensor.setTemp(temperatures[i]);
sensor.setHum(humidities[i]);
// Testing the trend after each input
if (i == 0) {
assertEquals(expected: "Increasing", sensor.getHumTrend());
} else if (i == 1) {
assertEquals(expected: "Decreasing", sensor.getHumTrend());
} else if (i == 2) {
assertEquals(expected: "Decreasing", sensor.getHumTrend());
} else if (i == 3) {
assertEquals(expected: "Increasing", sensor.getHumTrend());
} else if (i == 4) {
assertEquals(expected: "Increasing", sensor.getHumTrend());
} else if (i == 5) {
assertEquals(expected: "Increasing", sensor.getHumTrend());
} else if (i == 6) {
assertEquals(expected: "Stable", sensor.getHumTrend());
}
Tests passed: 4 of 4 tests – 2 ms
TestEmbeddedPart2B (lab5) 2 ms
/Users/softchair/Library/Java/JavaVirtualMachines/openjdk-20/Contents/Home/bin/java ...

a. What is the difference between testing the 7 inputs in a sequence and testing them individually. How are the two tests cases designed? (Use narrative description, no test code needed.)
    a. Testing them individually makes sure that each method that's used to set them is working as expected, but testing them in a sequence ensures that when setting a temp, it wont mess with other data points its not supposed to. The key differences between individually and sequentially is how they are isolated vs integrated, as well as dependency on the pervious test case.
b. As the same person who developed, refactored, and tested the code, does your refactored code make it easier or harder to test the system, explain with examples.
    a. It makes it easier to test the functions and ensure the algorithms for getting increasing/decreasing and also min max are working since its only in one function now. This allows you to simply test those two instead of having to test a lot of different functions for both temp and hum. It does make it harder to test these functions though if for example the other functions calls the algorithms incorrectly, then you will need to ensure this which could be overlooked.
c. Does your refactored code parametrize your tests ? Explain.
    a. The refactored code does not parametrize the tests.
d. If you received the refactored code (written by another developer) to just test it, would it be easier or harder than case above?
    a. It could be more difficult if the code was not documented well since you wouldn't know the expected out puts, but it can be easier once you know the basic functions, it can be easy to make sure to test all the functions and ensure they are working well together and then also by themselves
e. Would you prefer to test the original code or the refactored code, if both were written by another developer?
    a. I would prefer the original code as it makes more sense in terms of getting trend and also setting min max, but I can see why it would be easier to test the refactored code as there are less things to go wrong. If I had a strong sense and know what the code was supposed to be doing, the more generic code would be easier to test