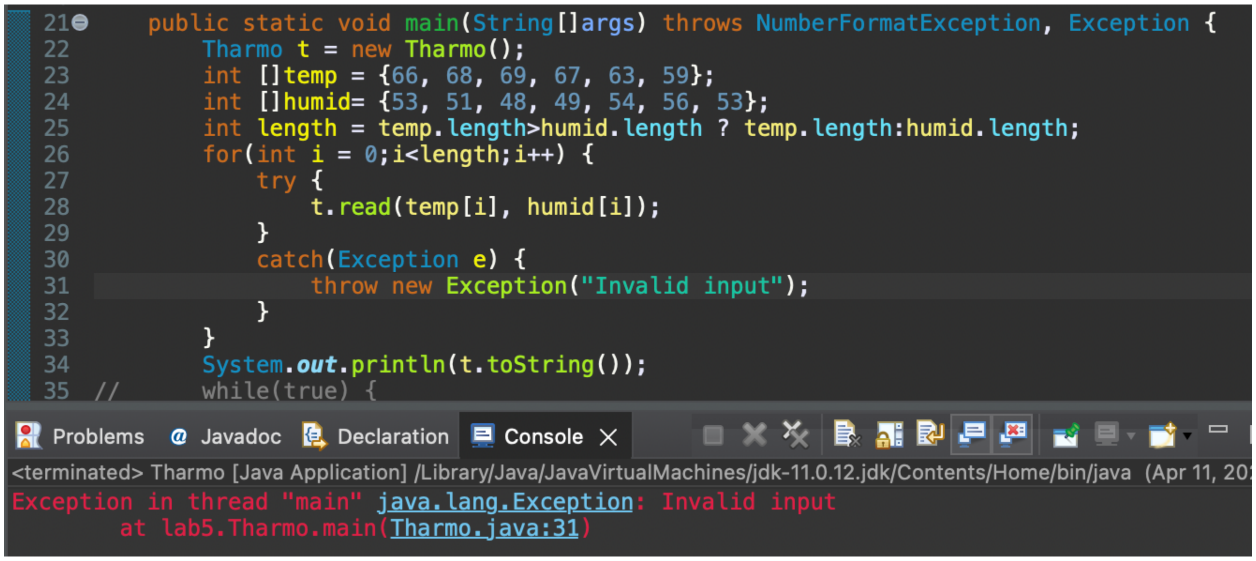
Lab5 part 2
Yuichi Hamamoto

a) It throws an exception because at the end of sequence it misses temperature input which indicates one of sensor is broken

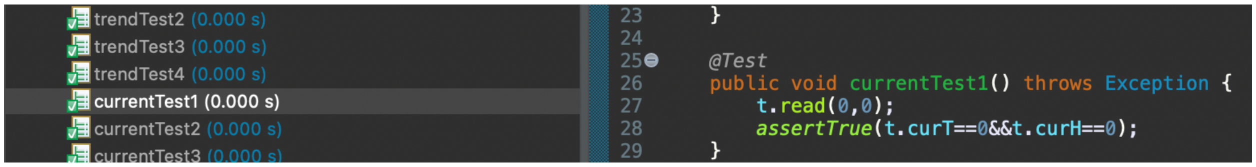


```
    public static void main(String[]args) throws NumberFormatException, Exception {
        Tharmo t = new Tharmo();
        int []temp = {66, 68, 69, 67, 63, 59};
        int []humid= {53, 51, 48, 49, 54, 56, 53};
        int length = temp.length>humid.length ? temp.length:humid.length;
        for(int i = 0;i<length;i++) {
            try {
                t.read(temp[i], humid[i]);
            }
            catch(Exception e) {
                throw new Exception("Invalid input");
            }
        }
        System.out.println(t.toString());
//      while(true) {
```

```
<terminated> Tharmo [Java Application] /Library/Java/JavaVirtualMachines/jdk-11.0.12.jdk/Contents/Home/bin/java  (Apr 11, 20
Exception in thread "main" java.lang.Exception: Invalid input
	at lab5.Tharmo.main(Tharmo.java:31)
```

b) Current humidity and temperature

Test checks if the current temp and humid set to be 0 after reading (0, 0)

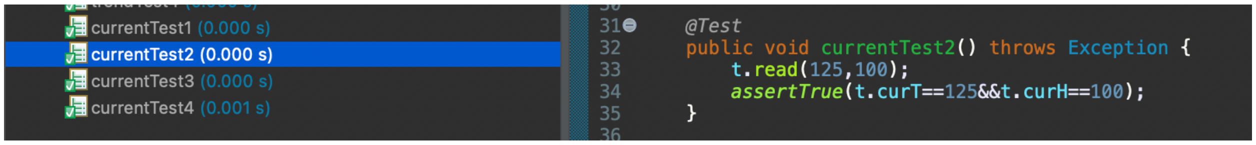


```
trendTest2 (0.000 s)
trendTest3 (0.000 s)
trendTest4 (0.000 s)
currentTest1 (0.000 s)
currentTest2 (0.000 s)
currentTest3 (0.000 s)
```

```
    }

    @Test
    public void currentTest1() throws Exception {
        t.read(0,0);
        assertTrue(t.curT==0&&t.curH==0);
    }
```

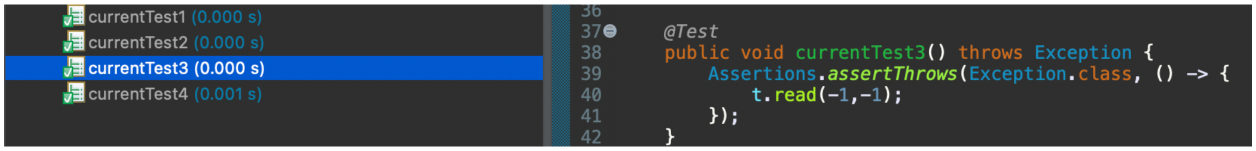
Test checks if the current temp and humid set to be 125 and 100 respectively after reading (125, 100)



```
currentTest1 (0.000 s)
currentTest2 (0.000 s)
currentTest3 (0.000 s)
currentTest4 (0.001 s)
```

```
    @Test
    public void currentTest2() throws Exception {
        t.read(125,100);
        assertTrue(t.curT==125&&t.curH==100);
    }
```

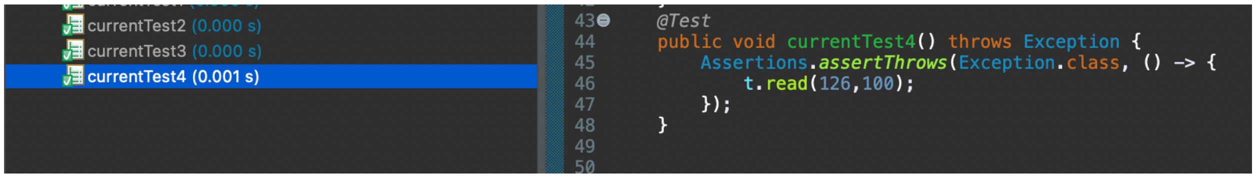
Test checks if it throws an exception when it receives invalid input (-1, -1)



```
currentTest1 (0.000 s)
currentTest2 (0.000 s)
currentTest3 (0.000 s)
currentTest4 (0.001 s)
```

```
    @Test
    public void currentTest3() throws Exception {
        Assertions.assertThrows(Exception.class, () -> {
            t.read(-1,-1);
        });
    }
```

Test checks if it throws an exception when it receives invalid input (126, 100)

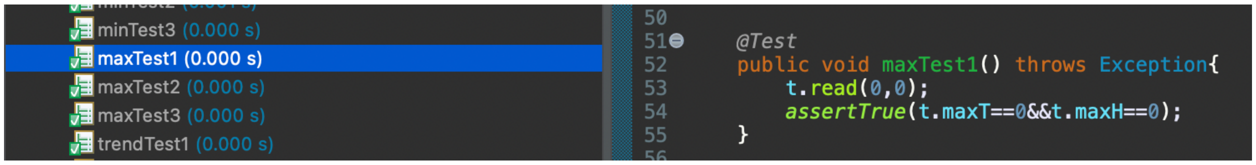


```
currentTest2 (0.000 s)
currentTest3 (0.000 s)
currentTest4 (0.001 s)
```

```java
@Test
public void currentTest4() throws Exception {
    Assertions.assertThrows(Exception.class, () -> {
        t.read(126,100);
    });
}
```

c) Max humidity and temperature

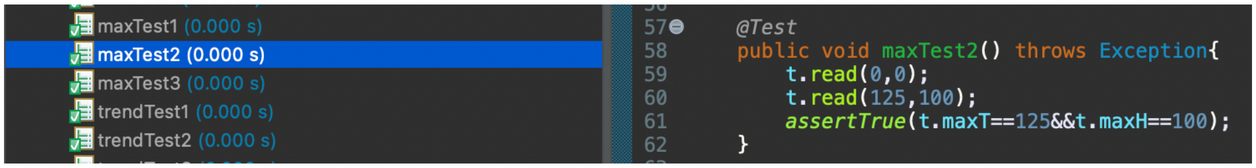
Test checks if the max temp and humid set to be 0 after reading (0, 0)



```
minTest3 (0.000 s)
maxTest1 (0.000 s)
maxTest2 (0.000 s)
maxTest3 (0.000 s)
trendTest1 (0.000 s)
```

```java
@Test
public void maxTest1() throws Exception{
    t.read(0,0);
    assertTrue(t.maxT==0&&t.maxH==0);
}
```

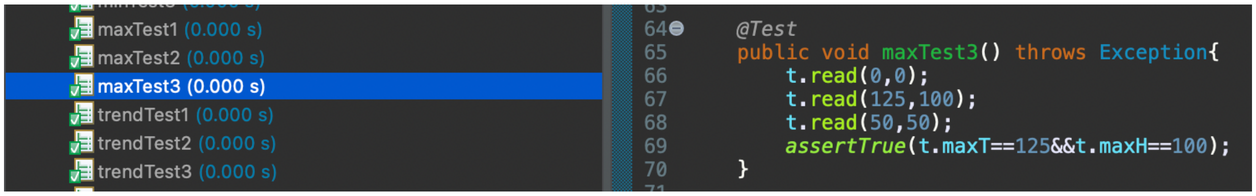
Test checks if the max temp and humid set to be 125 and 100 respectively after reading (0, 0), (125, 100)



```
maxTest1 (0.000 s)
maxTest2 (0.000 s)
maxTest3 (0.000 s)
trendTest1 (0.000 s)
trendTest2 (0.000 s)
```

```java
@Test
public void maxTest2() throws Exception{
    t.read(0,0);
    t.read(125,100);
    assertTrue(t.maxT==125&&t.maxH==100);
}
```

Test checks if the max temp and humid set to be 125 and 100 respectively after reading (0, 0), (125, 100), (50, 50)

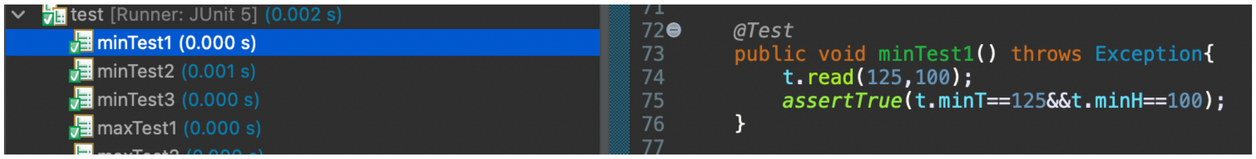


```
maxTest1 (0.000 s)
maxTest2 (0.000 s)
maxTest3 (0.000 s)
trendTest1 (0.000 s)
trendTest2 (0.000 s)
trendTest3 (0.000 s)
```

```java
@Test
public void maxTest3() throws Exception{
    t.read(0,0);
    t.read(125,100);
    t.read(50,50);
    assertTrue(t.maxT==125&&t.maxH==100);
}
```

d) Min humidity and temperature

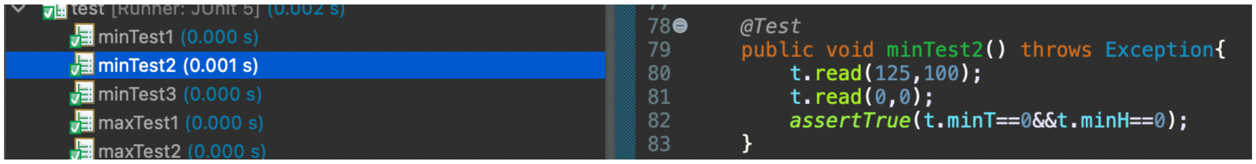
Test checks if the min temp and humid set to be 125 and 100 respectively after reading (125, 100)



```
test [Runner: JUnit 5] (0.002 s)
    minTest1 (0.000 s)
    minTest2 (0.001 s)
    minTest3 (0.000 s)
    maxTest1 (0.000 s)
```

```java
@Test
public void minTest1() throws Exception{
    t.read(125,100);
    assertTrue(t.minT==125&&t.minH==100);
}
```

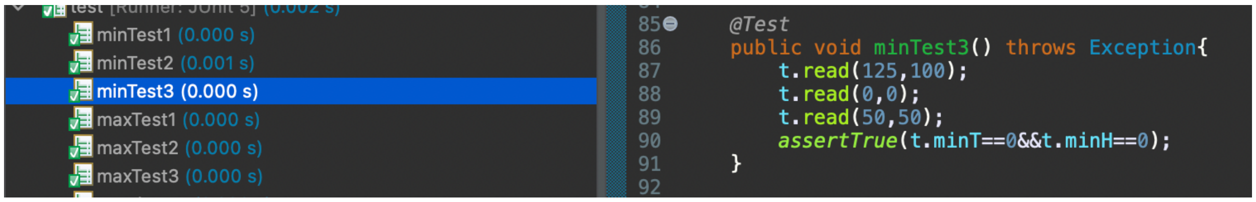
Test checks if the min temp and humid set to be 0 and 0 respectively after reading (125, 100), (0, 0)



```
test [Runner: JUnit 5] (0.002 s)
  minTest1 (0.000 s)
  minTest2 (0.001 s)
  minTest3 (0.000 s)
  maxTest1 (0.000 s)
  maxTest2 (0.000 s)
```

```java
	@Test
	public void minTest2() throws Exception{
		t.read(125,100);
		t.read(0,0);
		assertTrue(t.minT==0&&t.minH==0);
	}
```

Test checks if the min temp and humid set to be 0 and 0 respectively after reading (125, 100), (0, 0), (50, 50)

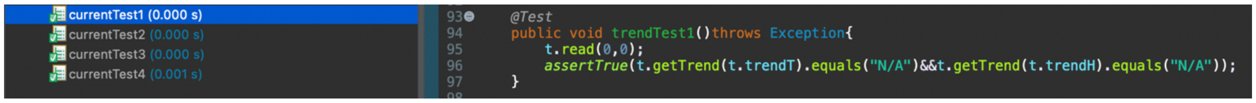


```
test [Runner: JUnit 5] (0.002 s)
  minTest1 (0.000 s)
  minTest2 (0.001 s)
  minTest3 (0.000 s)
  maxTest1 (0.000 s)
  maxTest2 (0.000 s)
  maxTest3 (0.000 s)
```

```java
	@Test
	public void minTest3() throws Exception{
		t.read(125,100);
		t.read(0,0);
		t.read(50,50);
		assertTrue(t.minT==0&&t.minH==0);
	}
```

e) Trends: Write separate test cases for each trend (up, stable, and down)

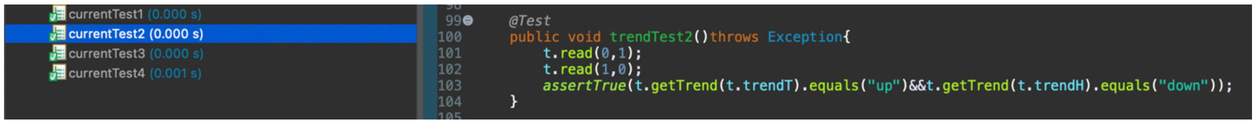
Test checks if trend of temp and humid is set to be N/A after reading (0, 0) as it does not have any previous data to compare and determine the trend.



```
  currentTest1 (0.000 s)
  currentTest2 (0.000 s)
  currentTest3 (0.000 s)
  currentTest4 (0.001 s)
```

```java
	@Test
	public void trendTest1()throws Exception{
		t.read(0,0);
		assertTrue(t.getTrend(t.trendT).equals("N/A")&&t.getTrend(t.trendH).equals("N/A"));
	}
```

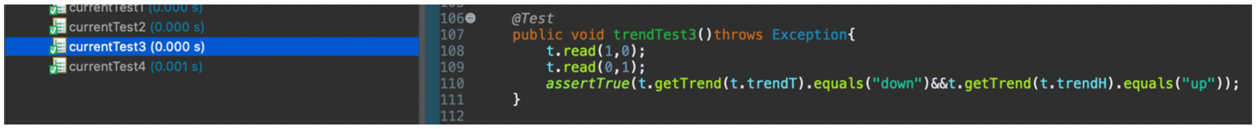
Test checks if trend of temp and humid is set to be up and down respectively after reading (0, 1) and (1, 0).



```
  currentTest1 (0.000 s)
  currentTest2 (0.000 s)
  currentTest3 (0.000 s)
  currentTest4 (0.001 s)
```

```java
	@Test
	public void trendTest2()throws Exception{
		t.read(0,1);
		t.read(1,0);
		assertTrue(t.getTrend(t.trendT).equals("up")&&t.getTrend(t.trendH).equals("down"));
	}
```

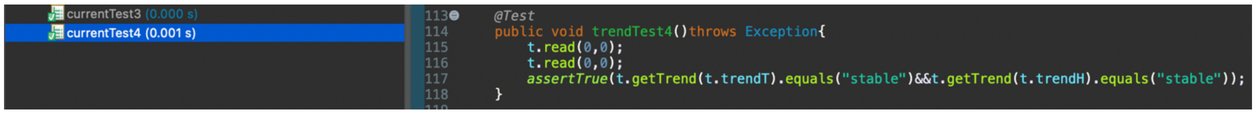
Test checks if trend of temp and humid is set to be down and up respectively after reading (1, 0) and (0, 1).



```
  currentTest1 (0.000 s)
  currentTest2 (0.000 s)
  currentTest3 (0.000 s)
  currentTest4 (0.001 s)
```

```java
	@Test
	public void trendTest3()throws Exception{
		t.read(1,0);
		t.read(0,1);
		assertTrue(t.getTrend(t.trendT).equals("down")&&t.getTrend(t.trendH).equals("up"));
	}
```

Test checks if trend of temp and humid is set to be stable after reading (0, 1) and (1, 0).



```
    @Test
    public void trendTest4()throws Exception{
        t.read(0,0);
        t.read(0,0);
        assertTrue(t.getTrend(t.trendT).equals("stable")&&t.getTrend(t.trendH).equals("stable"));
    }
```

iv) I did not have to refactor my code because my code was already refactored as I coded but I can imagine that it would be harder or more pain to test if it is not refactored as I would need to test more functions.

v) It might be harder to read the code, but I think it would be about the same.

vi) I personally prefer refactored one because I need to read and test fewer functions.co