

Primeras pruebas para que devuelva un String en vez de TRUE or FALSE con respecto a sintonización en radio AM o FM. Todavía había errores en el código para que corriera sin fallas.

```
1 package application;
2 import static org.junit.jupiter.api.Assertions.assertEquals;
3
4
5
6 public class MyTests {
7
8     @Test
9     public void radioTester() {
10         Radio tester = new Radioimp(); // MyClass is tested
11
12         // assert statements
13         //Aquí se ejecutara nuestro test.
14         tester.toggle();
15         assertEquals(true, tester.getState(), "Se acaba de prender el radio. Debe estar encendido");
16         tester.changeStation(true);
17         tester.saveStation(3);
18         tester.changeFrequency();
19         tester.changeStationButton(3);
20         tester.getStation();
21         assertEquals(true, tester.getFrequency(), "Debe estar en esta frecuencia FM.");
22         assertEquals(88.3, tester.getFrequency(), "Debe estar en esta estacion que se guarda en el 3.");
23     }
24 }
25 }
```

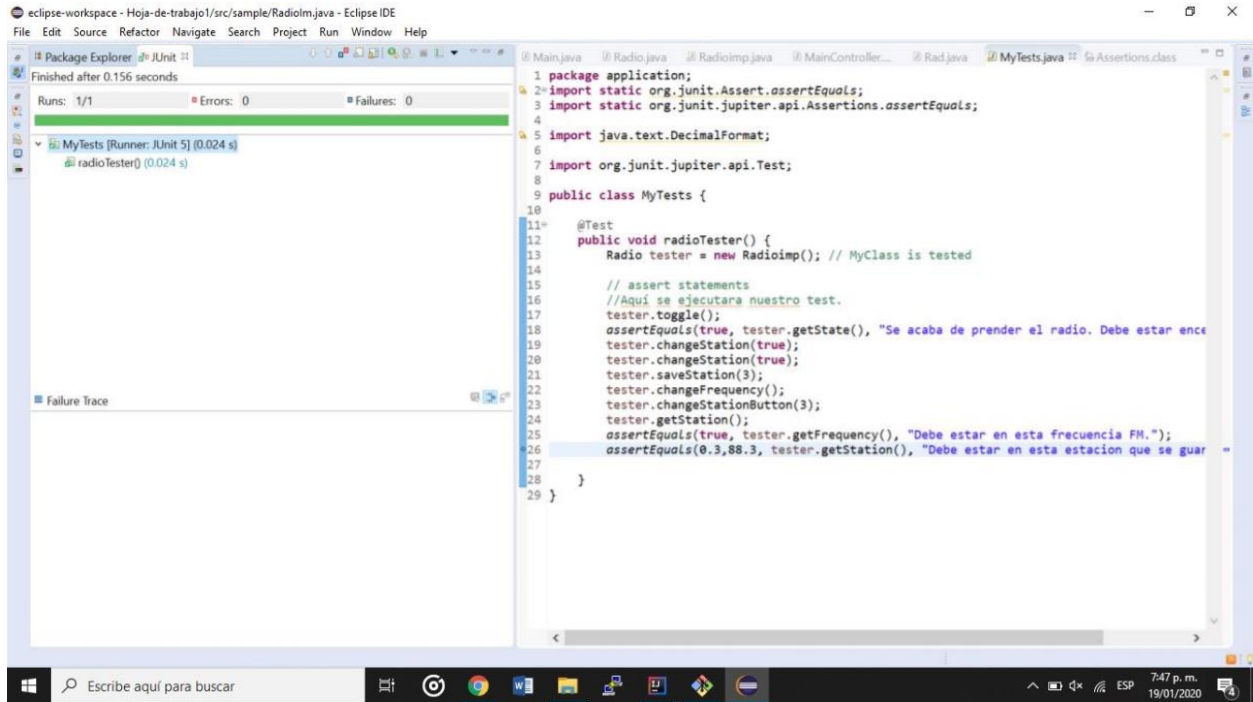
Failure Trace

```
AssertionError: Debe estar en esta estacion que se guarda en el 3. ==> expected: <FM>
Tests:radioTester(MyTests.java:22)
ForEachOps$ForEachOp$OpRef.accept(Unknown Source)
ReferencePipeline$2$1.accept(Unknown Source)
ForEachRemaining(Unknown Source)
Iterator$IteratorSplitter.forEachRemaining(Unknown Source)
AbstractPipeline.copyInto(Unknown Source)
AbstractPipeline.wrapAndCopyInto(Unknown Source)
ForEachOps$ForEachOp$OpRef.evaluateSequential(Unknown Source)
ForEachOps$ForEachOp$OpRef.evaluateSequential(Unknown Source)
AbstractPipeline.evaluate(Unknown Source)
```

```
1 package application;
2 import static org.junit.jupiter.api.Assertions.assertEquals;
3
4
5
6 public class MyTests {
7
8     @Test
9     public void radioTester() {
10         Radio tester = new Radioimp(); // MyClass is tested
11
12         // assert statements
13         //Aquí se ejecutara nuestro test.
14         tester.toggle();
15         assertEquals(true, tester.getState(), "Se acaba de prender el radio. Debe estar encendido");
16         tester.changeStation(true);
17         tester.saveStation(3);
18         tester.changeFrequency();
19         tester.changeStationButton(3);
20         tester.getStation();
21         assertEquals(true, tester.getFrequency(), "Debe estar en esta frecuencia FM.");
22         assertEquals(88.3, tester.getStation(), "Debe estar en esta estacion que se guarda en el 3.");
23     }
24 }
25 }
```

Failure Trace

```
AssertionError: Debe estar en esta estacion que se guarda en el 3. ==> expected: <88.3> but was: <88.30000000000001>
Tests:radioTester(MyTests.java:22)
ForEachOps$ForEachOp$OpRef.accept(Unknown Source)
ReferencePipeline$2$1.accept(Unknown Source)
ForEachRemaining(Unknown Source)
Iterator$IteratorSplitter.forEachRemaining(Unknown Source)
AbstractPipeline.copyInto(Unknown Source)
AbstractPipeline.wrapAndCopyInto(Unknown Source)
ForEachOps$ForEachOp$OpRef.evaluateSequential(Unknown Source)
ForEachOps$ForEachOp$OpRef.evaluateSequential(Unknown Source)
AbstractPipeline.evaluate(Unknown Source)
```



Ultima prueba donde se comprueba que todo funciona con normalidad y sin errores.