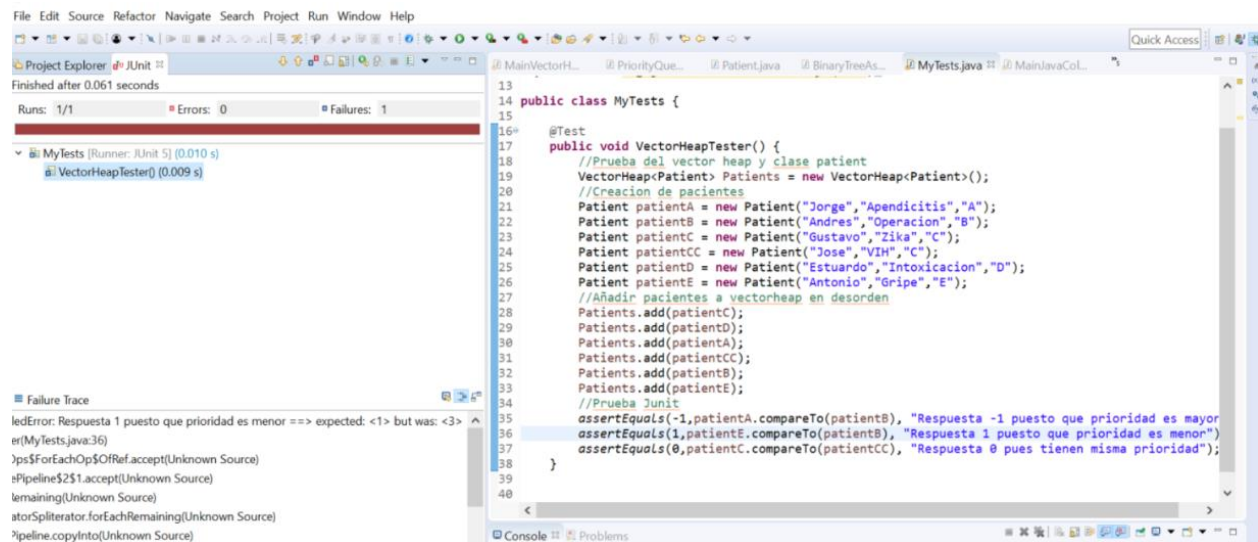
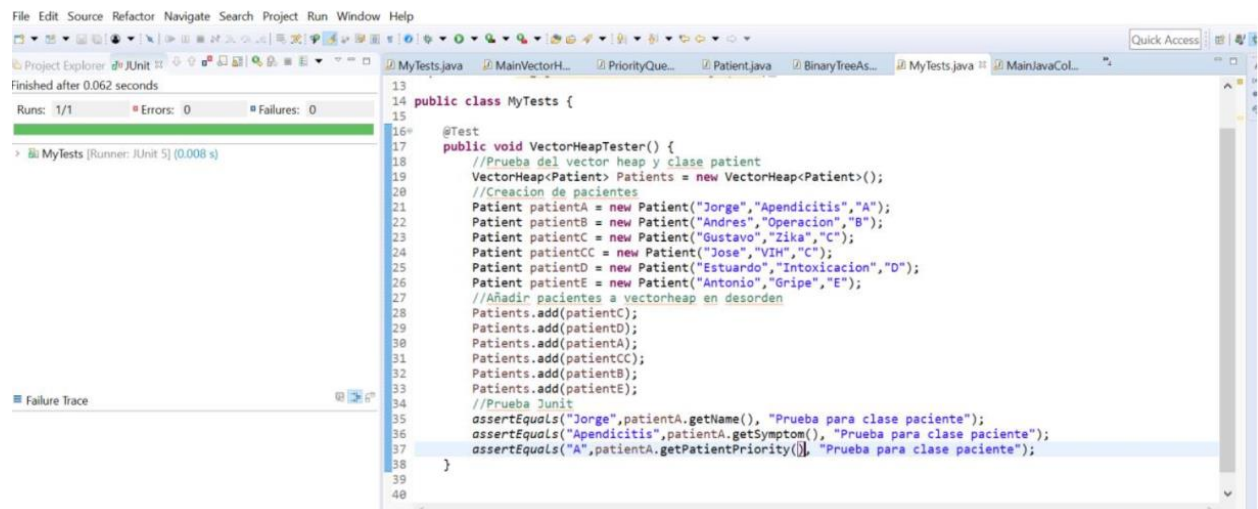


DANIELA VILLAMAR
#19086

PRUEBAS JUNIT

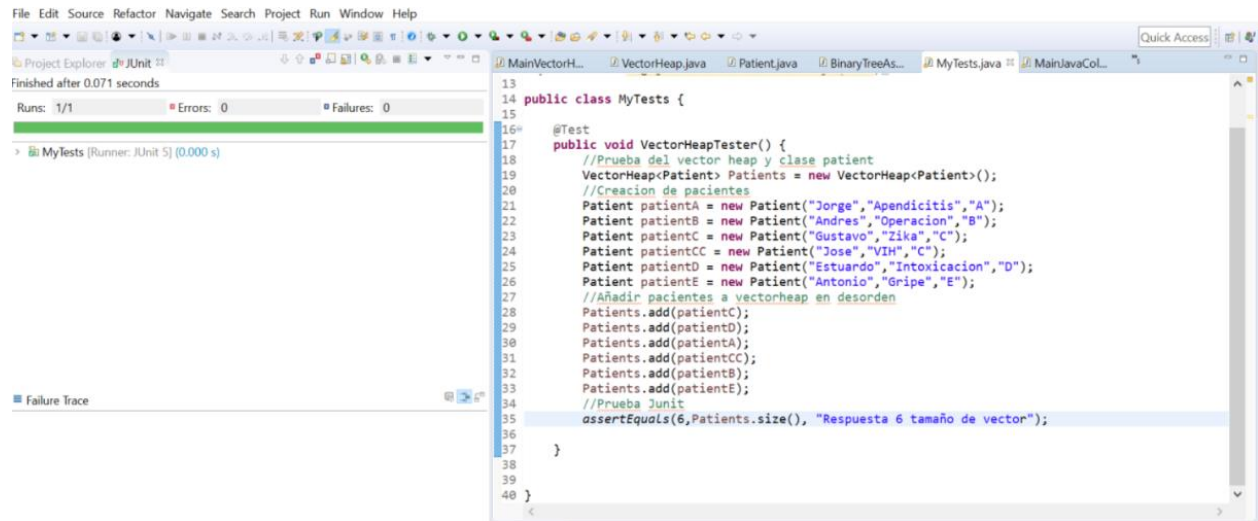


Error de comparable porque solo debía de devolver 1 o -1. Después no fue necesario porque compara si es menor a 0.



Prueba clase paciente

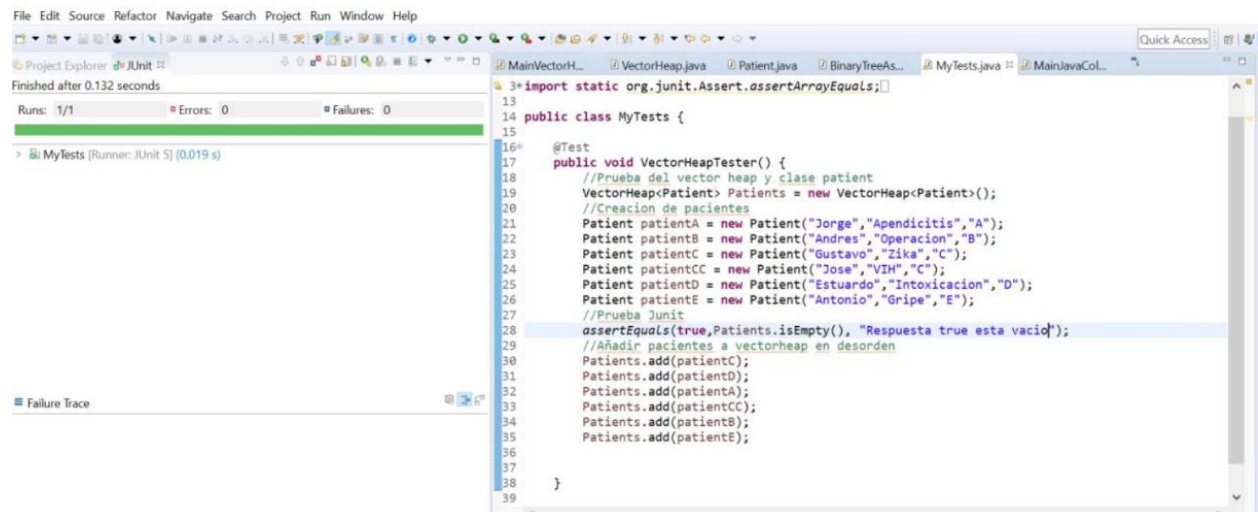
DANIELA VILLAMAR
#19086



```
File Edit Source Refactor Navigate Search Project Run Window Help
MainVectorH... VectorHeap.java Patient.java BinaryTreeAs... MyTests.java MainJavaCol...
Project Explorer JUnit
Finished after 0.071 seconds
Runs: 1/1 Errors: 0 Failures: 0
MyTests [Runner: JUnit 5] (0.000 s)
Failure Trace

13 public class MyTests {
14     @Test
15     public void VectorHeapTester() {
16         //Prueba del vector heap y clase patient
17         VectorHeap<Patient> Patients = new VectorHeap<Patient>();
18         //Creacion de pacientes
19         Patient patientA = new Patient("Jorge", "Apendicitis", "A");
20         Patient patientB = new Patient("Andres", "Operacion", "B");
21         Patient patientC = new Patient("Gustavo", "Zika", "C");
22         Patient patientCC = new Patient("Jose", "VIH", "C");
23         Patient patientD = new Patient("Estuardo", "Intoxicacion", "D");
24         Patient patientE = new Patient("Antonio", "Gripe", "E");
25         //Añadir pacientes a vectorheap en desorden
26         Patients.add(patientC);
27         Patients.add(patientD);
28         Patients.add(patientA);
29         Patients.add(patientCC);
30         Patients.add(patientB);
31         Patients.add(patientE);
32         //Prueba JUnit
33         assertEquals(6, Patients.size(), "Respuesta 6 tamaño de vector");
34     }
35 }
36
37
38
39
40 }
```

Prueba tamaño vector

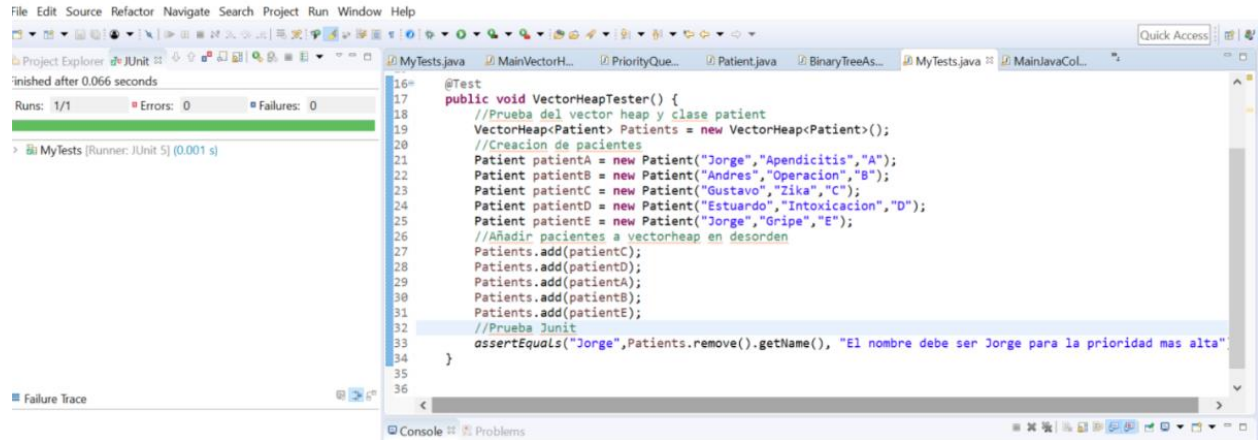


```
File Edit Source Refactor Navigate Search Project Run Window Help
MainVectorH... VectorHeap.java Patient.java BinaryTreeAs... MyTests.java MainJavaCol...
Project Explorer JUnit
Finished after 0.132 seconds
Runs: 1/1 Errors: 0 Failures: 0
MyTests [Runner: JUnit 5] (0.019 s)
Failure Trace

3 import static org.junit.Assert.assertEquals;
13 public class MyTests {
14     @Test
15     public void VectorHeapTester() {
16         //Prueba del vector heap y clase patient
17         VectorHeap<Patient> Patients = new VectorHeap<Patient>();
18         //Creacion de pacientes
19         Patient patientA = new Patient("Jorge", "Apendicitis", "A");
20         Patient patientB = new Patient("Andres", "Operacion", "B");
21         Patient patientC = new Patient("Gustavo", "Zika", "C");
22         Patient patientCC = new Patient("Jose", "VIH", "C");
23         Patient patientD = new Patient("Estuardo", "Intoxicacion", "D");
24         Patient patientE = new Patient("Antonio", "Gripe", "E");
25         //Prueba JUnit
26         assertEquals(true, Patients.isEmpty(), "Respuesta true esta vacio");
27         //Añadir pacientes a vectorheap en desorden
28         Patients.add(patientC);
29         Patients.add(patientD);
30         Patients.add(patientA);
31         Patients.add(patientCC);
32         Patients.add(patientB);
33         Patients.add(patientE);
34     }
35 }
36
37
38
39
40 }
```

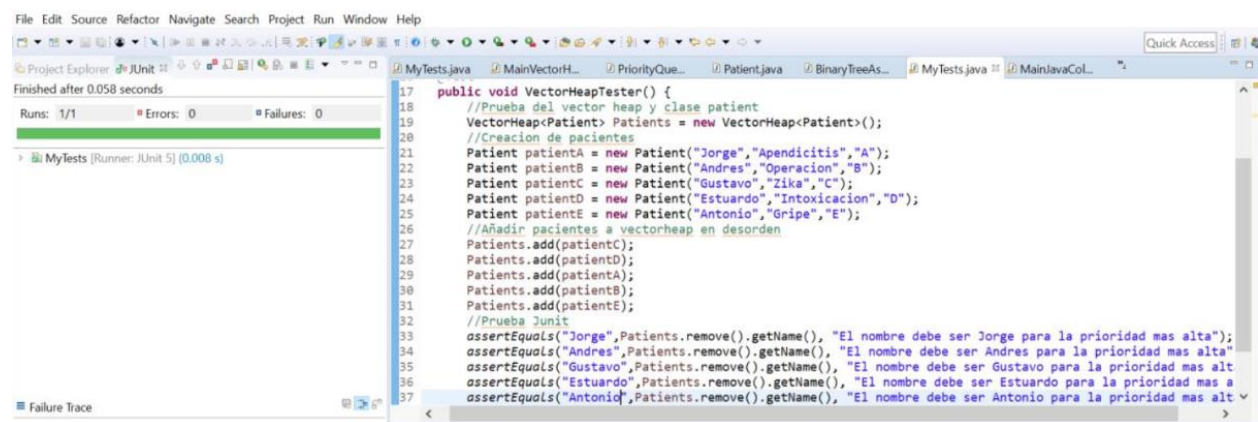
Prueba is empty vectorheap

DANIELA VILLAMAR
#19086



```
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit
Finished after 0.066 seconds
Runs: 1/1 Errors: 0 Failures: 0
> MyTests [Runner: JUnit 5] (0.001 s)
16 @Test
17 public void VectorHeapTester() {
18     //Prueba del vector heap y clase patient
19     VectorHeap<Patient> Patients = new VectorHeap<Patient>();
20     //Creacion de pacientes
21     Patient patientA = new Patient("Jorge", "Apendicitis", "A");
22     Patient patientB = new Patient("Andres", "Operacion", "B");
23     Patient patientC = new Patient("Gustavo", "Zika", "C");
24     Patient patientD = new Patient("Estuardo", "Intoxicacion", "D");
25     Patient patientE = new Patient("Jorge", "Gripe", "E");
26     //Aadir pacientes a vectorheap en desorden
27     Patients.add(patientC);
28     Patients.add(patientD);
29     Patients.add(patientA);
30     Patients.add(patientB);
31     Patients.add(patientE);
32     //Prueba Junit
33     assertEquals("Jorge", Patients.remove().getName(), "El nombre debe ser Jorge para la prioridad mas alta");
34 }
35
36
Console Problems
```

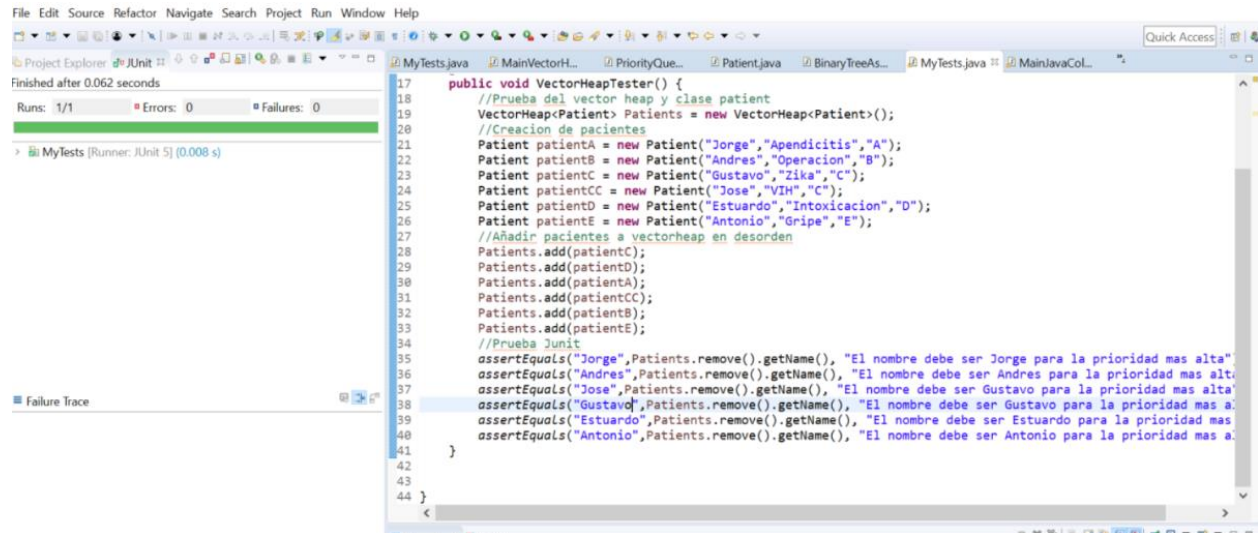
Verificación prioridad máxima



```
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit
Finished after 0.058 seconds
Runs: 1/1 Errors: 0 Failures: 0
> MyTests [Runner: JUnit 5] (0.008 s)
17 public void VectorHeapTester() {
18     //Prueba del vector heap y clase patient
19     VectorHeap<Patient> Patients = new VectorHeap<Patient>();
20     //Creacion de pacientes
21     Patient patientA = new Patient("Jorge", "Apendicitis", "A");
22     Patient patientB = new Patient("Andres", "Operacion", "B");
23     Patient patientC = new Patient("Gustavo", "Zika", "C");
24     Patient patientD = new Patient("Estuardo", "Intoxicacion", "D");
25     Patient patientE = new Patient("Antonio", "Gripe", "E");
26     //Aadir pacientes a vectorheap en desorden
27     Patients.add(patientC);
28     Patients.add(patientD);
29     Patients.add(patientA);
30     Patients.add(patientB);
31     Patients.add(patientE);
32     //Prueba Junit
33     assertEquals("Jorge", Patients.remove().getName(), "El nombre debe ser Jorge para la prioridad mas alta");
34     assertEquals("Andres", Patients.remove().getName(), "El nombre debe ser Andres para la prioridad mas alta");
35     assertEquals("Gustavo", Patients.remove().getName(), "El nombre debe ser Gustavo para la prioridad mas alta");
36     assertEquals("Estuardo", Patients.remove().getName(), "El nombre debe ser Estuardo para la prioridad mas alta");
37     assertEquals("Antonio", Patients.remove().getName(), "El nombre debe ser Antonio para la prioridad mas alta");
38 }
```

Prueba de prioridad para todos los valores de prioridad

DANIELA VILLAMAR
#19086



```
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit
Finished after 0.062 seconds
Runs: 1/1 Errors: 0 Failures: 0
> MyTests [Runner: JUnit 5] (0.008 s)
17 public void VectorHeapTester() {
18     //Prueba del vector heap y clase patient
19     VectorHeap<Patient> Patients = new VectorHeap<Patient>();
20     //Creacion de pacientes
21     Patient patientA = new Patient("Jorge", "Apendicitis", "A");
22     Patient patientB = new Patient("Andres", "Operacion", "B");
23     Patient patientC = new Patient("Gustavo", "Zika", "C");
24     Patient patientCC = new Patient("Jose", "VIH", "C");
25     Patient patientD = new Patient("Estuardo", "Intoxicacion", "D");
26     Patient patientE = new Patient("Antonio", "Gripe", "E");
27     //AÑadir pacientes a vectorheap en desorden
28     Patients.add(patientC);
29     Patients.add(patientD);
30     Patients.add(patientA);
31     Patients.add(patientCC);
32     Patients.add(patientB);
33     Patients.add(patientE);
34     //Prueba Junit
35     assertEquals("Jorge", Patients.remove().getName(), "El nombre debe ser Jorge para la prioridad mas alta");
36     assertEquals("Andres", Patients.remove().getName(), "El nombre debe ser Andres para la prioridad mas alta");
37     assertEquals("Jose", Patients.remove().getName(), "El nombre debe ser Gustavo para la prioridad mas alta");
38     assertEquals("Gustavo", Patients.remove().getName(), "El nombre debe ser Gustavo para la prioridad mas alta");
39     assertEquals("Estuardo", Patients.remove().getName(), "El nombre debe ser Estuardo para la prioridad mas alta");
40     assertEquals("Antonio", Patients.remove().getName(), "El nombre debe ser Antonio para la prioridad mas alta");
41 }
42
43
44 }
```

Prueba cuando existen dos con la misma prioridad