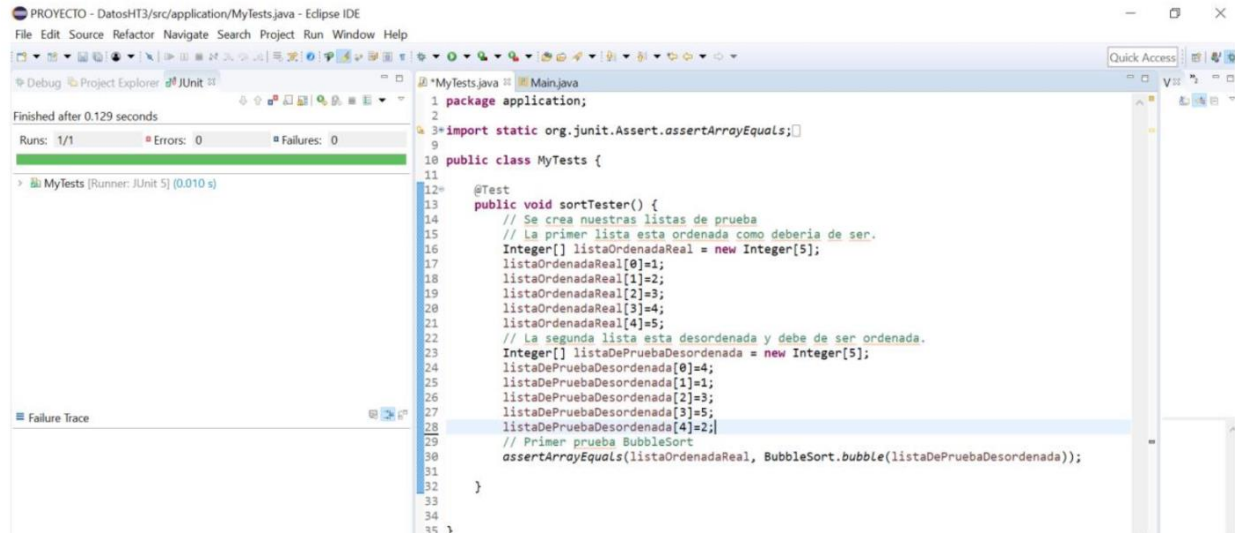


DANIELA VILLAMAR 19086

JUNIT TEST DE SORTS

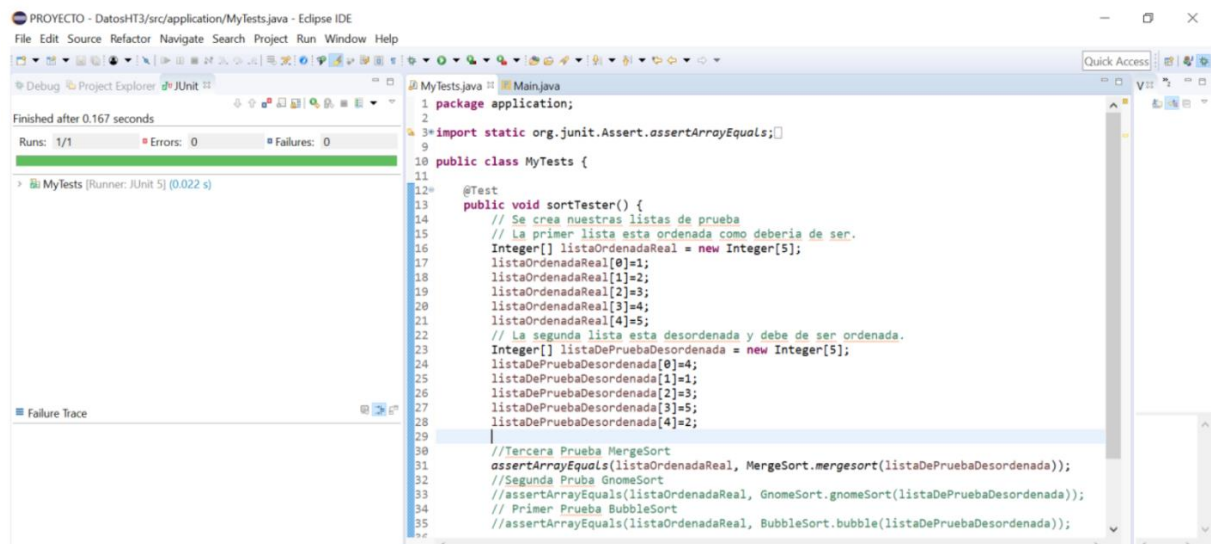
BUBBLESORT



```
1 package application;
2
3 import static org.junit.Assert.assertEquals;
4
5
6
7
8
9
10 public class MyTests {
11
12     @Test
13     public void sortTester() {
14         // Se crea nuestras listas de prueba
15         // La primer lista esta ordenada como deberia de ser.
16         Integer[] listaOrdenadaReal = new Integer[5];
17         listaOrdenadaReal[0]=1;
18         listaOrdenadaReal[1]=2;
19         listaOrdenadaReal[2]=3;
20         listaOrdenadaReal[3]=4;
21         listaOrdenadaReal[4]=5;
22         // La segunda lista esta desordenada y debe de ser ordenada.
23         Integer[] listaDePruebaDesordenada = new Integer[5];
24         listaDePruebaDesordenada[0]=4;
25         listaDePruebaDesordenada[1]=1;
26         listaDePruebaDesordenada[2]=3;
27         listaDePruebaDesordenada[3]=5;
28         listaDePruebaDesordenada[4]=2;
29         // Primer prueba BubbleSort
30         assertEquals(listaOrdenadaReal, BubbleSort.bubble(listaDePruebaDesordenada));
31     }
32 }
33
34
35
```

Finished after 0.129 seconds
Runs: 1/1 Errors: 0 Failures: 0
MyTests [Runner: JUnit 5] (0.010 s)

MERGESORT



```
1 package application;
2
3 import static org.junit.Assert.assertEquals;
4
5
6
7
8
9
10 public class MyTests {
11
12     @Test
13     public void sortTester() {
14         // Se crea nuestras listas de prueba
15         // La primer lista esta ordenada como deberia de ser.
16         Integer[] listaOrdenadaReal = new Integer[5];
17         listaOrdenadaReal[0]=1;
18         listaOrdenadaReal[1]=2;
19         listaOrdenadaReal[2]=3;
20         listaOrdenadaReal[3]=4;
21         listaOrdenadaReal[4]=5;
22         // La segunda lista esta desordenada y debe de ser ordenada.
23         Integer[] listaDePruebaDesordenada = new Integer[5];
24         listaDePruebaDesordenada[0]=4;
25         listaDePruebaDesordenada[1]=1;
26         listaDePruebaDesordenada[2]=3;
27         listaDePruebaDesordenada[3]=5;
28         listaDePruebaDesordenada[4]=2;
29
30         //Tercera Prueba MergeSort
31         assertEquals(listaOrdenadaReal, MergeSort.mergesort(listaDePruebaDesordenada));
32         //Segunda Pruba GnomeSort
33         //assertEquals(listaOrdenadaReal, GnomeSort.gnomeSort(listaDePruebaDesordenada));
34         // Primer Prueba BubbleSort
35         //assertEquals(listaOrdenadaReal, BubbleSort.bubble(listaDePruebaDesordenada));
36     }
37 }
38
39
40
```

Finished after 0.167 seconds
Runs: 1/1 Errors: 0 Failures: 0
MyTests [Runner: JUnit 5] (0.022 s)

QUICKSORT

```

1 package application;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class MyTests {
6
7     @Test
8     public void sortTester() {
9         // Se crea nuestras listas de prueba
10        // La primer lista esta ordenada como deberia de ser.
11        Integer[] listaOrdenadaReal = new Integer[5];
12        listaOrdenadaReal[0]=1;
13        listaOrdenadaReal[1]=2;
14        listaOrdenadaReal[2]=3;
15        listaOrdenadaReal[3]=4;
16        listaOrdenadaReal[4]=5;
17        // La segunda lista esta desordenada y debe de ser ordenada.
18        Integer[] listaDePruebaDesordenada = new Integer[5];
19        listaDePruebaDesordenada[0]=4;
20        listaDePruebaDesordenada[1]=1;
21        listaDePruebaDesordenada[2]=3;
22        listaDePruebaDesordenada[3]=5;
23        listaDePruebaDesordenada[4]=2;
24        //Cuarta Prueba QuickSort
25        assertEquals(listaOrdenadaReal, QuickSort.sort(listaDePruebaDesordenada));
26        //Tercera Prueba MergeSort
27        assertEquals(listaOrdenadaReal, MergeSort.mergesort(listaDePruebaDesordenada));
28        //Segunda Pruba GnomeSort
29        assertEquals(listaOrdenadaReal, GnomeSort.gnomeSort(listaDePruebaDesordenada));
30        //Primer Prueba BubbleSort
31        assertEquals(listaOrdenadaReal, BubbleSort.bubbleSort(listaDePruebaDesordenada));
32    }
33 }

```

SORTTEST

```

1 package application;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class MyTests {
6
7     @Test
8     public void sortTester() {
9         // Se crea nuestras listas de prueba
10        // La primer lista esta ordenada como deberia de ser.
11        Integer[] listaOrdenadaReal = new Integer[5];
12        listaOrdenadaReal[0]=1;
13        listaOrdenadaReal[1]=2;
14        listaOrdenadaReal[2]=3;
15        listaOrdenadaReal[3]=4;
16        listaOrdenadaReal[4]=5;
17        // La segunda lista esta desordenada y debe de ser ordenada.
18        Integer[] listaDePruebaDesordenada = new Integer[5];
19        listaDePruebaDesordenada[0]=4;
20        listaDePruebaDesordenada[1]=1;
21        listaDePruebaDesordenada[2]=3;
22        listaDePruebaDesordenada[3]=5;
23        listaDePruebaDesordenada[4]=2;
24        //Quinta Prueba RadixSort
25        assertEquals(listaOrdenadaReal, RadixSort.radixsort(listaDePruebaDesordenada, listaD
26        //Cuarta Prueba QuickSort
27        assertEquals(listaOrdenadaReal, QuickSort.sort(listaDePruebaDesordenada));
28        //Tercera Prueba MergeSort
29        assertEquals(listaOrdenadaReal, MergeSort.mergesort(listaDePruebaDesordenada));
30        //Segunda Pruba GnomeSort
31        assertEquals(listaOrdenadaReal, GnomeSort.gnomeSort(listaDePruebaDesordenada));
32    }
33 }

```