Capturas de JUnit tests

- ***Radix sort:***

RadixSortTest.java > RadixSortTest

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class RadixSortTest {
    @Test
    void testSort() {
        RadixSort<Integer> radixSort = new RadixSort<>();
        Integer[] array = {170, 45, 75, 90, 802, 24, 2, 66};
        Integer[] expected = {2, 24, 45, 66, 75, 90, 170, 802};

        assertArrayEquals(expected, radixSort.sort(array));
    }

    @Test
    void testEmptyArray() {
        RadixSort<Integer> radixSort = new RadixSort<>();
        Integer[] array = {};
        Integer[] expected = {};

        assertArrayEquals(expected, radixSort.sort(array));
    }

    @Test
    void testSingleElement() {
        RadixSort<Integer> radixSort = new RadixSort<>();
        Integer[] array = {42};
        Integer[] expected = {42};

        assertArrayEquals(expected, radixSort.sort(array));
    }
}
```

- ***Quick sort:***

QuickSortTest.java > ...

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class QuickSortTest {

    @Test
    public void testSort() {
        QuickSort<Integer> quickSort = new QuickSort<>();
        Integer[] array = {34, 7, 23, 32, 5, 62};
        Integer[] expected = {5, 7, 23, 32, 34, 62};
        assertArrayEquals(expected, quickSort.sort(array));
    }

    @Test
    public void testSortWithEmptyArray() {
        QuickSort<Integer> quickSort = new QuickSort<>();
        Integer[] array = {};
        Integer[] expected = {};
        assertArrayEquals(expected, quickSort.sort(array));
    }

    @Test
    public void testSortWithSingleElement() {
        QuickSort<Integer> quickSort = new QuickSort<>();
        Integer[] array = {42};
        Integer[] expected = {42};
        assertArrayEquals(expected, quickSort.sort(array));
    }
}
```

- ***Data Generator***

DataGeneratorTest.java > DataGeneratorTest > testGenerateData()

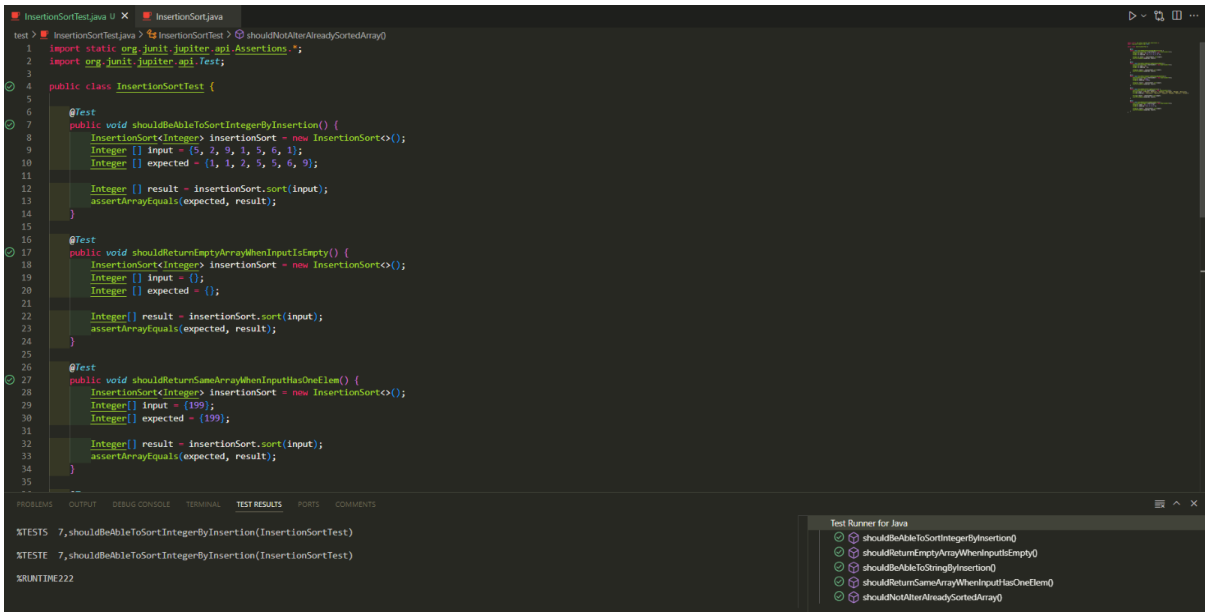
```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import java.io.File;

public class DataGeneratorTest {

    @Test
    public void testGenerateData() {
        // Crear una instancia de DataGenerator
        DataGenerator generator = new DataGenerator();
        String filename = "test_data.txt";


        generator.generateData(filename);


        File file = new File(filename);
        assertTrue(file.exists(), "El archivo no se creó correctamente");


        assertTrue(file.length() > 0, "El archivo está vacío");


        file.delete();
    }
}
```

- ***Data Reader:***

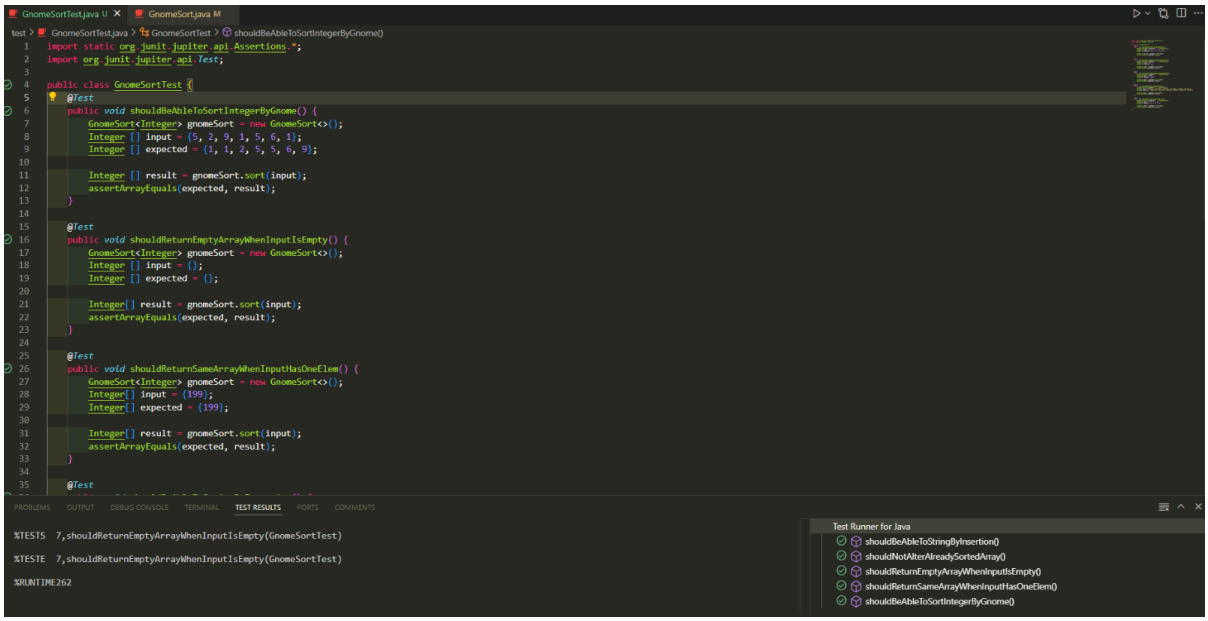
DataReaderTest.java > DataReaderTest > testReadData()

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import java.io.FileWriter;
import java.io.File;
import java.util.List;
import java.util.ArrayList;

@SuppressWarnings("unused")
public class DataReaderTest {
    @Test
    public void testReadData() {
        String filename = "test_read_data.txt";
        try (FileWriter writer = new FileWriter(filename)) {
            writer.write(str:"170\n45\n75\n90\n802\n24\n2\n66\n");
        } catch (Exception e) {
            e.printStackTrace();
        }


        List<List<Integer>> data = DataReader.readData(filename);

        assertEquals(1, data.size());
        assertEquals(8, data.get(index:0).size());

        // Verifica los valores
        List<Integer> dataset = data.get(index:0);
        assertArrayEquals(new Integer[]{170, 45, 75, 90, 802, 24, 2, 66}, dataset.toArray());

        new File(filename).delete();
    }

    @Test
    public void testReadDataWithMultipleDatasets() {
        String filename = "test_multiple_datasets.txt";
        try (FileWriter writer = new FileWriter(filename)) {
            writer.write(str:"Dataset size: 3\n170\n45\n75\nDataset size: 2\n90\n802\n");
        } catch (Exception e) {
            e.printStackTrace();
        }

        List<List<Integer>> data = DataReader.readData(filename);

        assertEquals(2, data.size());
        assertArrayEquals(new Integer[]{170, 45, 75}, data.get(index:0).toArray());
        assertArrayEquals(new Integer[]{90, 802}, data.get(index:1).toArray());
        new File(filename).delete();
    }
```

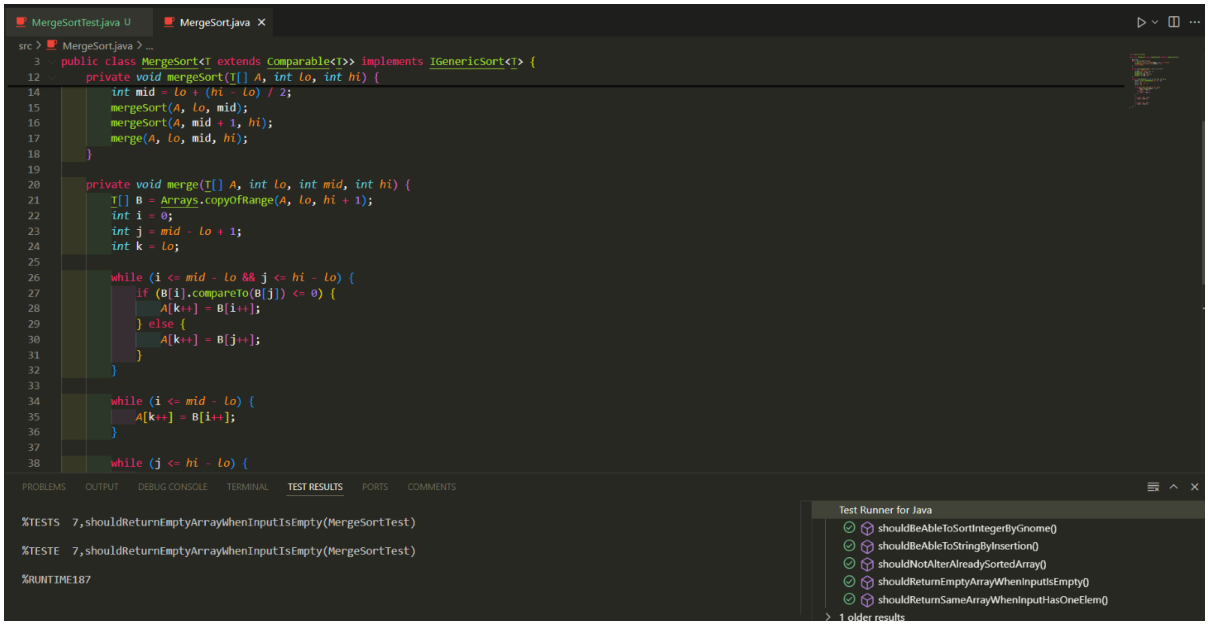
- ***Insertion Sort:***



- ***Gnome Sort:***



- ***Merge Sort:***


MergeSortTest.java U
MergeSort.java
src > MergeSort.java > ...
public class MergeSort<T extends Comparable<T>> implements IGenericSort<T> {
private void mergeSort(T[] A, int lo, int hi) {
int mid = lo + (hi - lo) / 2;
mergeSort(A, lo, mid);
mergeSort(A, mid + 1, hi);
merge(A, lo, mid, hi);
}
private void merge(T[] A, int lo, int mid, int hi) {
T[] B = Arrays.copyOfRange(A, lo, hi + 1);
int i = 0;
int j = mid - lo + 1;
int k = lo;
while (i <= mid - lo && j <= hi - lo) {
if (B[i].compareTo(B[j]) <= 0) {
A[k++] = B[i++];
} else {
A[k++] = B[j++];
}
}
while (i <= mid - lo) {
A[k++] = B[i++];
}
while (j <= hi - lo) {
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
TEST RESULTS
PORTS
COMMENTS
%TESTS 7,shouldReturnEmptyArrayWhenInputIsEmpty(MergeSortTest)
%TESTE 7,shouldReturnEmptyArrayWhenInputIsEmpty(MergeSortTest)
%RUNTIME187
Test Runner for Java
shouldBeAbleToSortIntegerByGnome()
shouldBeAbleToStringByInsertion()
shouldNotAlterAlreadySortedArray()
shouldReturnEmptyArrayWhenInputIsEmpty()
shouldReturnSameArrayWhenInputHasOneElem()
1 older results