APCS StudentRecord Problem

Files Needed: StudentRecord.java

Consider a grade-averaging scheme in which the final average of a student's scores is computed differently from the traditional average if the scores have "improved." Scores have improved if each score is greater than or equal to the previous score. The final average of the scores is computed as follows.

A student has `n` scores indexed from 0 to `n-1`. If the scores have improved, only those scores with indexes greater than or equal to `n/2` are averaged. If the scores have not improved, all the scores are averaged.

The following table shows several lists of scores and how they would be averaged using the scheme described above.

Student Scores	Improved?	Final Average
50, 50, 20, 80, 53	No	(50 + 50 + 20 + 80 + 53) / 5.0 = 50.6
20, 50, 50, 53, 80	Yes	(50 + 53 + 80) / 3.0 = 61.0
20, 50, 50, 80	Yes	(50 + 80) / 2.0 = 65.0

Consider the following incomplete `StudentRecord` class declaration. Each `StudentRecord` object stores a list of that student's scores and contains methods to compute that student's final average.

```
public class StudentRecord
{
  private int[] scores; // contains scores.length values
                        // scores.length > 1

  // constructors and other data fields not shown


  // returns the average (arithmetic mean) of the values in scores
  // whose subscripts are between first and last, inclusive
  // precondition:  0 <= first <= last < scores.length
  private double average(int first, int last)
  {  /* to be implemented in part (a) */  }


  // returns true if each successive value in scores is greater
  // than or equal to the previous value;
  // otherwise, returns false
  private boolean hasImproved()
  {  /* to be implemented in part (b) */  }


  // if the values in scores have improved, returns the average
  // of the elements in scores with indexes greater than or equal
  // to scores.length/2;
  // otherwise, returns the average of all of the values in scores
  public double finalAverage()
  {  /* to be implemented in part (c) */  }

}
```

Information

1. Fork the repository at jkimrusd/StudentRecord

2. The constructor and class variables have already been completed for you. You will be creating the methods for this class.

3. The Tester for the StudentRecord class is called StudentRecordTester.

4. In the `main()` function of StudentRecordTester, you have 4 arrays.

    a. An array of StudentRecord has been declared as `students`.

Directons

1. 3 `int` arrays called `a, b, c` contain the grades from the example in the previous page. Use this to create the 3 students in the `students` array.

2. **Part A** from the first page: Implement the method `average(int first, int last)` in the StudentRecord.java file. Test your `average()` method on each student. You will have to check the last two examples manually since the average for those are calculated in a different way. Make sure that you check the method with various inputs for first and last as well. Be aware that the `average()` method is a `private` method. Change it to `public` to test it, then change it back to `private` once you know that it works. You will also have to delete the method calls in the Tester once you change the method back to `private`.

3. **Part B** from the first page: Implement the method `hasImproved()` in the StudentRecord.java file.

4. Test your `hasImproved()` method on each one of the three `StudentRecord` objects. The first one should be `false`, while the last two should be `true`. Also, the `hasImproved()` method is a `private` method. Change it to `public` to test it, then change it back to `private` once you know that it works. You will also have to delete the method calls in the Tester once you change the method back to `private`.

5. Test your `finalAverage()` method on each one of the three `StudentRecord` objects. You should get the exact same final average for each example listed on page 1 of this document.